







Photonic realization of a quantum finite automaton

Carlo Mereghetti ¹, Beatrice Palano ², Simone Cialdi ^{1,3}, Valeria Vento ¹,
Matteo G. A. Paris ^{1,3} and Stefano Olivares ^{1,3,*}

¹Dipartimento di Fisica “Aldo Pontremoli,” Università degli Studi di Milano, via Celoria 16, 20133 Milano, Italy

²Dipartimento di Informatica “Giovanni Degli Antoni,” Università degli Studi di Milano, via Celoria 18, 20133 Milano, Italy

³INFN Sezione di Milano, via Celoria 16, 20133 Milano, Italy



(Received 23 July 2019; published 28 January 2020)

We describe a physical implementation of a quantum finite automaton that recognizes a well-known family of periodic languages. The realization exploits the polarization degree of freedom of single photons and their manipulation through linear optical elements. We use techniques of confidence amplification to reduce the acceptance error probability of the automaton. It is worth remarking that the quantum finite automaton we physically realize is not only interesting *per se* but it turns out to be a crucial building block in many quantum finite automaton design frameworks theoretically settled in the literature.

DOI: [10.1103/PhysRevResearch.2.013089](https://doi.org/10.1103/PhysRevResearch.2.013089)

I. INTRODUCTION

Quantum computing is a prolific research area, halfway between physics and computer science [1–5]. Most likely, its origins may be dated back to the 1970s, when some work on quantum information began to appear (see, e.g., Refs. [6,7]). In the early 1980s, Feynman suggested that the computational power of quantum mechanical processes might be beyond that of traditional computation models [8]. A similar idea was put forth by Manin [9]. Almost at the same time, Benioff proved that such processes are at least as powerful as Turing machines [10]. In 1985, Deutsch proposed the notion of a quantum Turing machine as a physically realizable model for a quantum computer [11].

The first impressive result witnessing “quantum power” was Shor’s algorithm for integer factorization, which could run in polynomial time on a quantum computer [12]. It should be stressed that no classical polynomial time factoring algorithm is currently known. On this fact, the security of many current cryptographic protocols, e.g., Rivest-Shamir-Adleman (RSA) and Diffie-Hellman, actually relies. Relevant progress was made by Grover, who proposed a quantum algorithm for searching an item in an unsorted database containing n items, which runs in time $O(\sqrt{n})$ [13].

These and other theoretical advances naturally drove much attention to efforts on *the physical realization* of quantum computational devices (see, e.g., Refs. [14–17]). While we can hardly expect to see a full-featured quantum computer in the near future, it might be reasonable to envision classical computing devices incorporating quantum components. Since the physical realization of quantum computational systems

has proved to be an extremely complex task, it is also reasonable to keep quantum components as small as possible. Small-size quantum devices are modeled by *quantum finite automata*, a theoretical model for quantum machines with finite memory.

Indeed, in current implementations of quantum computing, the preparation and initialization of qubits in superposition and/or entangled states is often challenging, making worthwhile the study of quantum computation with restricted memory, which requires less demanding resources, as in the case of the quantum finite automata.

The simplest and most promising from a physical realization viewpoint model of a quantum finite automaton is the so-called *measure-once quantum finite automaton* [18–21]. Such a model also served as a basis for defining several variants of quantum finite automata introduced and studied in plenty of contributions (see, e.g., Refs. [22–28]). Because it is the only model considered in the present paper, from now on for the sake of brevity we will simply write “quantum finite automaton” instead of “measure-once quantum finite automaton.”

The “hardware” of a (one-way) quantum finite automaton is that of a classical finite automaton. Thus, we have an input tape scanned by a one-way input head moving one position forward at each move, plus a finite basis state control. Some basis states are designated as *accepting* states. At any given time during the computation, the state of the quantum finite automaton is represented by a complex linear combination of classical basis states, called a superposition. At each step, a unitary transformation associated with the currently scanned input symbol makes the automaton evolve to the next superposition. Superposition dynamics can transfer the complexity of the problem from a large number of sequential steps to a large number of coherently superposed quantum states. At the end of input processing, the automaton is observed in its final superposition. This operation makes the superposition collapse to a particular (classical) basis state with a certain probability. The probability that the automaton accepts the input word is given by the probability of observing

*stefano.olivares@fisica.unimi.it

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article’s title, journal citation, and DOI.

(collapsing into) an accepting basis state. Quantum finite automata exhibit both advantages and disadvantages with respect to their classical (e.g., deterministic or probabilistic) counterparts. Basically, quantum superposition offers some computational advantages on probabilistic superposition. On the other hand, quantum dynamics must be reversible, and this requirement may impose severe computational limitations to finite memory devices. As a matter of fact, it is sometimes impossible to simulate classical finite automata by quantum finite automata. In fact, as we will discuss in Sec. II D, isolated cut point quantum finite automata recognize a proper subclass of regular languages [18,20,21].

Although weaker from a computational power point of view, quantum finite automata may greatly outperform classical ones when *descriptive power* is at stake. In the realm of descriptive complexity [29], models of computation are compared on the basis of their *size*. In the case of finite state machines, a commonly assumed size measure is the number of finite control states. Most likely, the first contribution explicitly studying the descriptive power of quantum versus classical finite automata is Ref. [30], where an extremely succinct quantum finite automaton is provided, accepting the unary language $L_m = \{a^k \mid k \in \mathbb{N} \text{ and } k \bmod m = 0\}$ for any given $m > 0$. The construction in Ref. [30] uses as a basic (and sole) module a quantum finite automaton \mathcal{A} for L_m with 2 basis states, whose acceptance reliability is then enhanced within a suitable modular building framework where traditional compositions (i.e., direct products and sums) of quantum systems are performed. Actually, many (if not all) contributions in the literature aiming to design small size quantum finite automata for several tasks (see, e.g., Refs. [25,31–38]) use the module \mathcal{A} as a crucial building block. In this sense, the language L_m and the module \mathcal{A} turn out to be “paradigmatic” as tools to build and test size-efficient quantum finite automata. Hence, a physical realization of the module \mathcal{A} might be well worth investigating.

In this paper, we put forward a physical implementation of quantum finite automata based on the polarization degree of freedom of single photons and able to recognize a family of periodic languages. More precisely, because of above stressed centrality in quantum finite automaton design frameworks, we focus on the physical implementation of the quantum finite automaton \mathcal{A} for the language L_m . We investigate the performance of our photonic automaton, taking into account the main sources of error and imperfections, e.g., in the preparation of the initial automaton state. We also use techniques of confidence amplification to reduce the acceptance error probability of the automaton.

The paper is structured as follows. In Sec. II, we provide an almost self-contained overview of the basic concepts underlying formal language theory and classical finite automata. Moreover, we quickly address practical impacts of finite automata and the importance of investigating their size in the light of possible physical implementations of such devices. Next, we present the notion of a quantum finite automaton together with some basic facts on its computational and descriptive power. We particularly focus on unary automata, i.e., automata with a single-letter input alphabet, and emphasize the notion of a language accepted with isolated cut point. In Sec. III, we introduce a simple unary language, as

a benchmark upon which to test the descriptive power of classical and quantum finite automata, namely the language $L_m = \{a^k \mid k \in \mathbb{N} \text{ and } k \bmod m = 0\}$ for any given $m > 0$. We provide a theoretical definition of a quantum finite automaton \mathcal{A} accepting L_m with isolated cut point and two basis states, whereas any classical automaton for L_m requires a number of states which grows with m .

The photonic implementation of the quantum finite automaton \mathcal{A} with two basis states is then discussed in Sec. IV. There, we start reviewing the standard quantum formalism used to describe the polarization state of the single photon, its dynamics, and the link with the formalism used in the previous sections. Then, we explain the working principle of the photonic implementation of the quantum finite automaton and propose a discrimination strategy to reduce the acceptance error probability. Section V describes the experimental apparatus and reports the results we obtained. Finally, we close the paper with Sec. VI, where we draw some concluding remarks and the outlook for our work.

II. PRELIMINARIES

A. Formal languages and classical finite automata

Formal language theory studies languages from a mathematical point of view, providing formal tools and methods to analyze language properties. Strictly connected with *automata theory*, the discipline dates back to the 1950s, and it was originally developed to provide a theoretical basis for natural language processing. It was soon realized that this theory was relevant to the artificial languages (e.g., programming languages) that had originated in computer science. Since its birth, formal language theory has become established as one of the most prominent areas in theoretical computer science. Its results have huge impacts in numerous fields, including practical computer science, cryptography and security, discrete mathematics and combinatorics, graph theory, mathematical logic, nature-inspired (e.g., quantum, biological, genetic) computational models, physics, and system theory.

The reader may find a lot of excellent textbooks where thoughtful presentations of formal language and automata theory and their applications are presented (see, e.g., Refs. [39,40]). In order to keep this paper as self-contained as possible, we present basic concepts and notations of formal language and automata theory and briefly emphasize those aspects which are relevant to the present work, i.e., regular languages and finite automata.

An alphabet is any finite set Σ of elements called symbols. A word on Σ is a sequence $\omega = \sigma_1\sigma_2 \dots \sigma_n$ with $\sigma_i \in \Sigma$ being its i th symbol. The length of ω , i.e., the number of symbols ω consists of, is denoted by $|\omega|$. We let ε be the empty word satisfying $|\varepsilon| = 0$. The set of all words (including the empty word) on Σ is denoted by Σ^* , and we let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. A language L on Σ is any subset of Σ^* , i.e., $L \subseteq \Sigma^*$. If $|\Sigma| = 1$, we say that Σ is a *unary* alphabet, and languages on unary alphabets are called unary languages. In the case of unary alphabets, we customarily let $\Sigma = \{a\}$ so that a unary language is any set $L \subseteq a^*$. The concatenation of the word $x \in \Sigma^*$ with the word $y \in \Sigma^*$ is the word xy consisting of the sequence of symbols of x immediately followed by the

sequence of symbols of y . For any $\sigma \in \Sigma$ and any positive integer k , we let σ^k be the word obtained by concatenating k times the symbol σ . We stipulate that $\sigma^0 = \varepsilon$.

Several formal tools have been introduced to rigorously express languages. *Formal grammars* are the main *generative systems* for languages. A formal grammar is a quadruple $G = (\Sigma, Q, P, S)$ where Σ and Q are two disjoint finite alphabets of, respectively, terminal and nonterminal symbols, $S \in Q$ is the start symbol, and P is the finite set of production rules, or simply, productions. Productions can be regarded as rewriting rules, typically expressed in the form $\alpha \rightarrow \beta$ with $\alpha \in (\Sigma \cup Q)^+$ and $\beta \in (\Sigma \cup Q)^*$. Given $w, z \in (\Sigma \cup Q)^*$, we say that z is derived in one step from w in G whenever $w = x\alpha y$, $z = x\beta y$, and $\alpha \rightarrow \beta$ is a production rule in P . Formally, we write $w \Rightarrow_G z$. More generally, z is derived from w in G whenever there is a sequence $w_0, w_1, \dots, w_{n-1}, w_n \in (\Sigma \cup Q)^*$ such that $w = w_0 \Rightarrow_G w_1 \Rightarrow_G \dots \Rightarrow_G w_{n-1} \Rightarrow_G w_n = z$. Formally, we write $w \Rightarrow_G^* z$. The language generated by the grammar $G = (\Sigma, Q, P, S)$ is the set $L(G) \subseteq \Sigma^*$ defined as $L(G) = \{\omega \in \Sigma^* \mid S \Rightarrow_G^* \omega\}$. Two grammars G, G' are equivalent whenever $L(G) = L(G')$.

The following example provides a grammar and establishes the corresponding generated language.

Example 1. When listing grammar production rules, we can write $\alpha \rightarrow \beta_1|\beta_2|\dots|\beta_{n-1}|\beta_n$ as a shortcut for expressing the set of productions $\alpha \rightarrow \beta_1, \alpha \rightarrow \beta_2, \dots, \alpha \rightarrow \beta_n$. So, consider the grammar $G = (\Sigma = \{a, b\}, Q = \{B_0, \dots, B_k\}, P, B_0)$ where the set P of productions is defined as

$$P = \{B_0 \rightarrow aB_0|bB_0|bB_1\} \\ \cup \{B_i \rightarrow aB_{i+1}|bB_{i+1} \text{ for } 1 \leq i \leq k-1, B_k \rightarrow \varepsilon\}.$$

Let us derive the generated language $L(G)$. By repeatedly applying the productions $B_0 \rightarrow aB_0|bB_0$, from the start symbol B_0 we can derive αB_0 , for any $\alpha \in \{a, b\}^*$. Formally, $B_0 \Rightarrow_G^* \alpha B_0$. At this point, in order to generate a word of terminal symbols only, we must apply the production $B_0 \rightarrow bB_1$, thus having $B_0 \Rightarrow_G^* \alpha B_0 \Rightarrow_G \alpha bB_1$. Then, we are left to sequentially apply the productions $B_i \rightarrow aB_{i+1}|bB_{i+1}$ for every $1 \leq i \leq k-1$. So, $B_0 \Rightarrow_G^* \alpha B_0 \Rightarrow_G \alpha bB_1 \Rightarrow_G^* \alpha b\beta B_k$, for any $\beta \in \{a, b\}^*$ and $|\beta| = k-1$. By applying the last production $B_k \rightarrow \varepsilon$, we get $B_0 \Rightarrow_G^* \alpha B_0 \Rightarrow_G \alpha bB_1 \Rightarrow_G^* \alpha b\beta B_k \Rightarrow_G \alpha b\beta$. Thus, the language generated by G writes as

$$L(G) = \{\omega \in \{a, b\}^* \mid \omega = \alpha b\beta \text{ and } |\beta| = k-1\}.$$

In words, $L(G)$ consists of those words on $\{a, b\}$ featuring a symbol b at the k th position from the right.

Originally, four types of grammars have been pointed out, depending on the form of productions. The corresponding four classes of generated languages turn out to be relevant both from practical and theoretical points of view. Precisely, $G = (\Sigma, Q, P, S)$ is a grammar of the following:

TYPE 0: whenever productions in P do not have any particular restriction. The class of languages generated by this type of grammar is the class of *recursively enumerable languages*.

TYPE 1 or context-sensitive: whenever every production $\alpha \rightarrow \beta \in P$ satisfies $|\alpha| \leq |\beta|$; the production $S \rightarrow \varepsilon$ is allowed provided S never occurs within the right part of any

production in P . The class of languages generated by this type of grammar is the class of *context-sensitive languages*.

TYPE 2 or context-free: whenever every production in P is of the form $A \rightarrow \beta$ with $A \in Q$. The class of languages generated by this type of grammar is the class of *context-free languages*.

TYPE 3 or regular: whenever every production is of the form $A \rightarrow \varepsilon, A \rightarrow \sigma, \text{ or } A \rightarrow \sigma B$ with $\sigma \in \Sigma$ and $A, B \in Q$. The class of languages generated by this type of grammar is the class of *regular languages*. The reader may easily verify that the grammar proposed in Exercise 1 is a type 3 grammar, and hence the generated language is an example of regular language.

It can be shown that for any given type $i+1$ grammar, an equivalent type i grammar can be built. Hence, the class of regular languages is contained in the class of context-free languages, which is contained in the class of context-sensitive languages, which in turn is contained in the class of recursively enumerable languages. In addition, we have that such a language class hierarchy is proper. In fact, (i) there exist languages outside the class of recursively enumerable languages, (ii) there exist recursively enumerable languages that cannot be generated by any context-sensitive grammar, (iii) the ternary context-sensitive language $\{a^n b^n c^n \mid n \in \mathbf{N}\}$ cannot be generated by any context-free grammar, (iv) the binary context-free language $\{a^n b^n \mid n \in \mathbf{N}\}$ cannot be generated by any regular grammar. Beside the one in Example 1, further instances of regular languages will be provided below. This language class hierarchy is usually known as the *Chomsky hierarchy*, and the whole formal language and automata theory has been developing around it. Every level of the hierarchy has been deeply investigated, yielding profound results and widespread applications.

An alternative equivalent approach to define the Chomsky hierarchy uses language *accepting* systems, i.e., roughly speaking, formal computational devices which process input words and outcome an accept/reject final verdict. For one such device, the corresponding accepted (or recognized) language consists of those input words that are accepted. According to this point of view, (i) the class of recursively enumerable languages coincides with the class of languages accepted by *Turing machines*, (ii) the class of context-sensitive languages coincides with the class of languages accepted by *linear bounded automata*, (iii) the class of context-free languages coincides with the class of languages accepted by *nondeterministic pushdown automata*, and (iv) the class of regular languages coincides with the class of languages accepted by (several types of) *finite automata*.

In this paper, we will be concerned with the class of regular languages. In particular, we will focus on the computational model of *finite automata* defining them [see (iv) above]. For extensive and thoughtful surveys on classical finite automata theory, the reader is referred to, e.g., Refs. [39–41]. Several types of finite automata have been introduced and deeply investigated in the literature. Let us begin by the original and most basic version. In Fig. 1, the “hardware” of a *one-way deterministic finite automaton* (1dfa, for short [42]) A is depicted. We remark that the other versions of finite automata we are going to review share the same hardware but exhibit different dynamics.

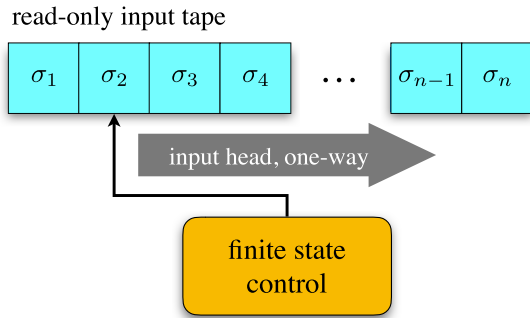


FIG. 1. Schematic diagram of the “hardware” of a one-way deterministic finite automaton (1dfa). The 1dfa is made of a read-only input tape consisting of a sequence of cells, each one capable of storing a symbol. The tape may be scanned by a “head”, which is moving one position right at each step. At each stage of the computation of A , a finite state control is in a state from a finite set Q .

We have a read-only input tape consisting of a sequence of cells, each one being able to store an input symbol. The tape is scanned by an input head always moving one position right at each step. This type of input head motion motivates the designation “one way.” At each time during the computation of A , a finite state control is in a state from a finite set Q . Some of the states in Q are designated as *accepting* states, while $q_0 \in Q$ is a designated initial state. The computation of A on a word ω from a given input alphabet Σ begins by having (i) ω stored symbol by symbol, left to right, in the cells of the input tape, (ii) the input head scanning the leftmost tape cell, and (iii) the finite state control being in the state q_0 . In a move, A reads the symbol below the input head and, depending on such a symbol and the state of the finite state control, it switches to the next state according to a fixed transition function and moves the input head one position forward. We say that A *accepts* ω if and only if it enters an accepting state after scanning the rightmost symbol of ω ; otherwise, A rejects ω . The language accepted by A is the set $L(A) \subseteq \Sigma^*$ consisting of all the input words accepted by A .

Formally, a 1dfa is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, with $q_0 \in Q$ being the initial state and $F \subseteq Q$ being the set of accepting states, Σ is the input alphabet, and $\delta : Q \times \Sigma \rightarrow Q$ is the transition function defining moves as follows: If A scans the input symbol σ by being in the state p and $\delta(p, \sigma) = q$ holds, then it enters the state q and shifts the input head one position forward. The transition function δ can be inductively extended from symbols in Σ to words in Σ^* as $\delta : Q \times \Sigma^* \rightarrow Q$. Namely, for any $q \in Q$ and $\omega \in \Sigma^*$, we let

$$\delta(q, \omega) = \begin{cases} q & \text{if } \omega = \varepsilon \\ \delta(\delta(q, \sigma), \alpha) & \text{if } \omega = \sigma\alpha. \end{cases}$$

Thus, the language accepted by A is the set $L(A) \subseteq \Sigma^*$ defined as $L(A) = \{\omega \in \Sigma^* \mid \delta(q_0, \omega) \in F\}$.

A nice pictorial representation of a 1dfa $A = (Q, \Sigma, \delta, q_0, F)$ is by its state (or transition) graph D_A . Basically, D_A is a labeled digraph having Q as the set of its vertexes and labeled directed edges representing moves. Precisely, there exists an edge from vertex p to vertex q with label σ if and only if $\delta(p, \sigma) = q$ holds true. Vertexes are

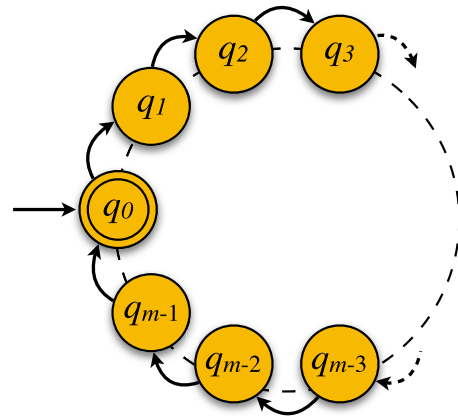


FIG. 2. The state graph for the 1dfa A accepting the language L_m .

usually drawn as circles on the plan with labels indicating the corresponding states, while labeled arrows join adjacent states. The vertex corresponding to the state q_0 has an incoming arrow, while vertexes associated with accepting states in F are double circled. It is easy to see that the computation of A on the input word ω can be tracked in D_A by following the unique directed path labeled ω from the vertex q_0 . So, A *accepts* ω if and only if such a path ends up in a double circled vertex.

To clarify the above notions, the next example displays a 1dfa accepting a simple unary language. We provide such a 1dfa both in its formal definition as a quintuple and as state graph.

Example 2. The following simple unary language will play an important role throughout the rest of the paper. For any given integer $m > 0$, let

$$L_m = \{a^k \mid k \in \mathbb{N} \text{ and } k \bmod m = 0\}. \tag{1}$$

Such a language can be accepted by the 1dfa

$$A = (Q = \{q_0, q_1, \dots, q_{m-1}\}, \Sigma = \{a\}, \delta, q_0, F = \{q_0\}),$$

where, for any $0 \leq i \leq m - 1$, we set $\delta(q_i, a) = q_{(i+1) \bmod m}$. It is easy to see that $\delta(q_0, a^k) = q_{k \bmod m}$ which is q_0 if and only if $k \bmod m = 0$ if and only if $a^k \in L_m$. Hence, $L(A) = L_m$. The state graph for the 1dfa A is depicted in Fig. 2. Because of unary input alphabet, all edges would have the same label a , which can then be safely omitted.

Let us now turn to the model of a *one-way nondeterministic finite automaton* (1nfa, for short [42]). Formally, a 1nfa is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$ in which every component is defined as in 1dfa’s but the transition function, which is now a mapping $\delta : Q \times \Sigma \rightarrow 2^Q$, where 2^Q denotes the powerset of Q , i.e., the set of all subsets of Q . Unlike the deterministic case, now at each move A has several candidates as possible next states. Precisely, if A scans the input symbol σ by being in the state p and $\delta(p, \sigma) = S$ holds, then it may enter one of the states in S and shift the input head one position forward. Thus, on any input word ω , more computation paths from q_0 exist; if at least one of such paths leads to an accepting state, then A *accepts* ω . More formally, we can inductively extend the transition function δ to subsets of states and words as $\delta : 2^Q \times \Sigma^* \rightarrow 2^Q$. First of all, we define the extension $\delta : 2^Q \times \Sigma \rightarrow 2^Q$ as $\delta(S, \sigma) = \cup_{q \in S} \delta(q, \sigma)$, for any $S \subseteq Q$ and $\sigma \in \Sigma$.

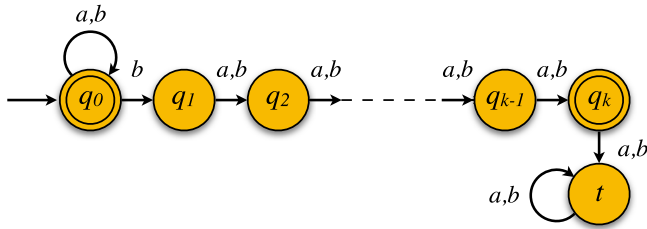


FIG. 3. The state graph of a 1nfa for the language E_k .

Then, for any $S \subseteq Q$ and $\omega \in \Sigma^*$, we let

$$\delta(S, \omega) = \begin{cases} S & \text{if } \omega = \varepsilon \\ \delta(\delta(S, \sigma), \alpha) & \text{if } \omega = \sigma\alpha. \end{cases}$$

Thus, the language accepted by A is the set $L(A) \subseteq \Sigma^*$ defined as $L(A) = \{\omega \in \Sigma^* \mid \delta(\{q_0\}, \omega) \cap F \neq \emptyset\}$. The reader may easily verify that a 1dfa can be seen as a 1nfa where, for any $q \in Q$ and $\sigma \in \Sigma$, we have that $\delta(q, \sigma)$ contains a single state.

The state graph D_A for the 1nfa $A = (Q, \Sigma, \delta, q_0, F)$ can be defined as above for the deterministic case, but now an edge from vertex p to vertex q with label σ exists if and only if $q \in \delta(p, \sigma)$ holds true. This means that, in general, a vertex may present more outgoing edges with the same label. Thus, A accepts an input word ω if and only if there exists a path in D_A labeled ω from q_0 to a double circled vertex.

The following example proposes a 1nfa expressed as state graph for a binary language.

Example 3. Consider the binary language in Example 1, for which a type 3 grammar was there provided. Here, we call that language E_k which was defined as

$$E_k = \{\omega \in \{a, b\}^* \mid \omega = \alpha b \beta \text{ and } |\beta| = k - 1\}. \quad (2)$$

Thus, a word on $\{a, b\}$ is in E_k if and only if its k th symbol from the right is b . In Fig. 3, the state graph of a 1nfa accepting E_k is depicted. The reader may easily verify that the accepted language is exactly E_k . Moreover, she may straightforwardly work out an equivalent formal definition of the 1nfa as a quintuple.

We complete our overview of classical models of finite automata by introducing the notion of a *one-way probabilistic finite automaton* (1pfa, for short [43]). Formally, a 1pfa is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$ in which every component is defined as usual, but now δ returns a probability distribution for the next state. More precisely, $\delta : Q \times \Sigma \times Q \rightarrow [0, 1]$ is defined such that $\delta(p, \sigma, q)$ is the probability that A , being in the state p , reaches the state q upon reading the symbol σ . As usual, the input head is shifted one position right at each move. Clearly, for any $p \in Q$ and $\sigma \in \Sigma$, we require that $\sum_{q \in Q} \delta(p, \sigma, q) = 1$. Inductively extending the transition function δ to words enables us to get $\delta : Q \times \Sigma^* \times Q \rightarrow [0, 1]$, where $\delta(p, \omega, q)$ yields the probability that A , being in the state p , reaches the state q upon reading the input word ω as

$$\begin{aligned} & \delta(p, \omega, q) \\ &= \begin{cases} 0 & \text{if } \omega = \varepsilon \text{ and } p \neq q \\ 1 & \text{if } \omega = \varepsilon \text{ and } p = q \\ \sum_{s \in Q} \delta(p, \sigma, s) \cdot \delta(s, \alpha, q) & \text{if } \omega = \sigma\alpha. \end{cases} \end{aligned}$$

Thus, the probability that A accepts the input word ω is written as $p_A(\omega) = \sum_{q \in F} \delta(q_0, \omega, q)$, i.e., the probability for A to reach an accepting state from the initial state q_0 after processing ω . Given a real number λ , we define the language accepted by A with cut point λ as the set $L_{A,\lambda} = \{\omega \in \Sigma^* \mid p_A(\omega) > \lambda\}$. A language $L \subseteq \Sigma^*$ is said to be accepted by A with isolated cut point λ whenever $L = L_{A,\lambda}$ and there exists $\rho > 0$ such that $|p_A(\omega) - \lambda| \geq \rho$ for every $\omega \in \Sigma^*$. The relevance of isolated cut point acceptance is due to the fact that, in this case, we can arbitrarily reduce the classification error probability of an input word by repeating a constant number of times (not depending on the length of the input word) its parsing and taking the majority of the answers [41,43]. In our experiment, we will use this fact to reduce the error probability. Notice that beside isolated cut point acceptance, other probabilistic acceptance modes are widely studied in the literature (see, e.g., Refs. [24,25,36,44]).

Without going into detail, even with a 1pfa A , a state graph D_A can be naturally associated. Now, edges in D_A are labeled by both a symbol and the corresponding transition probability.

Example 4. For two primes m, n , let the unary language

$$L_{m \cdot n} = \{a^k \mid k \in \mathbb{N} \text{ and } k \bmod (m \cdot n) = 0\}. \quad (3)$$

Notice that this is a particular instance of the unary language introduced in Example 2. We define the set of states $Q = \{s, p_0, \dots, p_{m-1}, q_0, \dots, q_{n-1}\}$ and construct the 1pfa $A = (Q, \Sigma = \{a\}, \delta, s, F = \{s, p_0, q_0\})$ where we set

$$\delta(s, a, p_1) = \frac{1}{2} = \delta(s, a, q_1),$$

$$\delta(p_i, a, p_{(i+1) \bmod m}) = 1 = \delta(q_j, a, q_{(j+1) \bmod n})$$

for $0 \leq i \leq m - 1$ and $0 \leq j \leq n - 1$,

and any other transition occurs with probability 0.

It is not hard to see that

$$p_A(a^k) = \begin{cases} 1 & \text{if } a^k \in L_{m \cdot n} \\ \leq \frac{1}{2} & \text{otherwise.} \end{cases}$$

Thus, the 1pfa A accepts $L_{m \cdot n}$ with cut point $\frac{3}{4}$ isolated by $\frac{1}{4}$. The state graph of the 1pfa A is sketched in Fig. 4. As usual, due to unary input alphabet, we omit the label a from every edge. Moreover, each edge without an associated probability defines a move occurring with certainty.

For the sake of completeness, we point out that *two-way finite automata* are also considered in the literature. Very roughly speaking, a two-way finite automaton has the same hardware as a one-way finite automaton, but its input head can move one position forward or backward, or stand still at each move. Two-way motion of the input head can be adopted by the three paradigms above recalled, thus leading to the models of 2dfa's, 2nfa's, and 2pfa's. Formal definitions and properties of two-way finite automata may be found, e.g., in Refs. [39–43,45,46].

The computational power of all these (and actually of many other) variants of finite automata has been well established in the literature over many years of research. As suggested in point (iv), in the automata-based characterization of Chomsky hierarchy above recalled, the following is true:

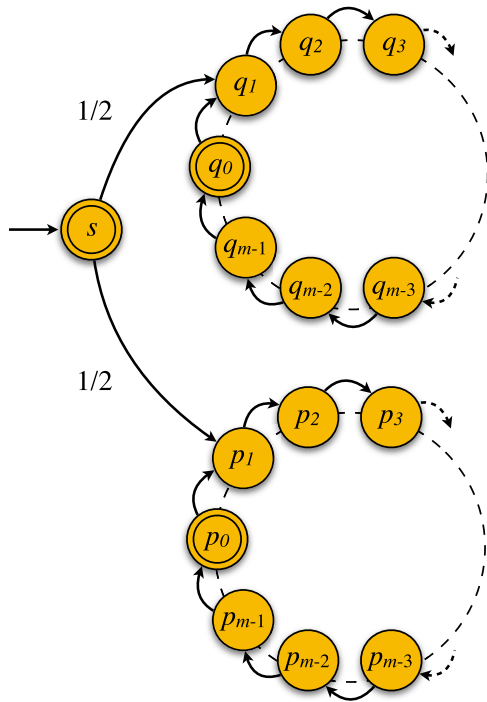


FIG. 4. The state graph of the 1pfa A for the language L_{m-n} .

Theorem 5. The class of languages accepted by $\{1, 2\}$ dfa’s, $\{1, 2\}$ nfa’s, or 1pfa’s with isolated cut point coincides with the class of regular languages.

The class of regular languages is properly contained in the class of languages accepted by isolated cut point 2pfa’s [45]. However, when restricted to unary alphabets, even isolated cut point 2pfa’s accept exactly unary regular languages [46].

Regular languages are of fundamental importance in many applications in computer science. Viewing regular languages throughout finite automata greatly improved compiler and interpreter design, parsing and pattern-matching algorithms, cryptography and security protocol testing, computer network protocol testing, model checking, and software validation. It is not an exaggeration to say that almost any task in computer science sooner or later leads to coping with some regular language which can be fruitfully managed via a suitable finite automaton.

However, beside being a valuable tool in language processing, finite automata represent a formidable theoretical model to deal with those physical systems which exhibit a predetermined sequence of actions depending on a sequence of events they are presented. Originally, finite automata have been introduced to describe the electric activity of brain neurons, but soon they have been extensively used in the design and analysis of several devices such as the control units for vending machines, elevators, traffic lights, combination locks, etc.

Particularly important is the use of finite automata in very large scale integration (VLSI) design, namely, in the project of sequential networks which are the building blocks of modern computers and digital systems. Very roughly speaking, a sequential network is a boolean circuit equipped with memory. Engineering a sequential network typically requires modeling its behavior with a finite automaton whose number of states

directly influences the amount of hardware (i.e., the number of logic gates) employed in the electronic realization of the sequential network. From this point of view, having fewer states in the modeling finite automaton directly results in employing smaller hardware which, in turn, means having less energy absorption and fewer cooling problems. These latter physical implementation aspects, as the reader may easily figure out, turn out to be of paramount importance given the current level of digital device miniaturization.

These “physical” (and other more theoretical) considerations have led to a trend in the literature in which, beside acceptance capabilities, the *descriptive* power of finite automata is deeply investigated. Within the realm of *descriptive complexity* [29], the *size* of finite automata is under consideration, and a common measure for finite automaton size is the *number of states*. In particular, reducing or increasing the number of states is studied, when using different computational paradigms (e.g., deterministic, nondeterministic, probabilistic, quantum, one-way, two-way) on a finite automaton to perform a given task. Let us quickly recall some very well-known results on the descriptive power of different types of finite automata. To this aim, we say that two finite automata A, A' are equivalent whenever $L(A) = L(A')$.

It is well known that any n -state 1nfa can be converted into an equivalent 2^n -state 1dfa [42], and that in general such an exponential size blowup is unavoidable. In fact, consider the language E_k in Example 3. There, a k -state 1nfa accepting E_k is sketched, but it can be shown that any 1dfa for E_k cannot have fewer than 2^k states. A similar exponential gap exists for 1dfa’s versus 1pfa’s: Any n -state 1pfa accepting a language with cut point isolated by ρ can be turned into an equivalent 1dfa with $(1 + 1/\rho)^n$ states [43]. Even in this case, the exponential blowup is in general “almost unavoidable” (stating the exact size gap between determinism and probabilism is an open problem). This can be proved by elaborating on the language L_{m-n} provided in Example 4. Equivalent 1dfa’s for n -state 2dfa’s and 2nfa’s can be obtained, paying by not less than n^n and 2^{n^2} states, respectively [42,47].

Following this line of research on the succinctness of different computational paradigms, we are going to investigate whether and how adopting the quantum paradigm of computation may reduce the number of states on finite state automata, thus providing theoretical foundations for the realization of more succinct devices with all potential benefits in terms of miniaturization and energy consumption above addressed.

To this aim, we will be particularly interested in *unary one-way finite automata*, i.e., automata having a unary input alphabet consisting of the sole symbol a . Clearly, unary one-way finite automata accept unary languages $L \subseteq a^*$. Here, we choose to provide a nice and compact matrix presentation of unary one-way finite automata that will naturally lead to formalizing the notion of a unary one-way quantum finite automaton. We recall that a matrix is said to be *boolean* whenever its entries are either 0 or 1 and *stochastic* whenever its entries are real numbers from the interval $[0,1]$ and each row sums to 1.

Let A be a unary one-way finite automaton with $\{q_1, q_2, \dots, q_n\}$ being the set of its states; some of these states are accepting. Then, A can be formally written as a triple $A = (\zeta, U, \eta)$, where $\eta \in \{0, 1\}^{n \times 1}$ is the characteristic

column vector of the accepting states, i.e., $\eta_i = 1$ if and only if q_i is an accepting state, while ζ and U have different forms depending on the nature of A . Precisely, A is as follows:

1dfa: $\zeta \in \{0, 1\}^n$ is the characteristic row vector of the initial state, U is an $n \times n$ boolean stochastic transition matrix, and hence U has exactly a single 1 per row, with $U_{ij} = 1$ if and only if and only if A moves from the state q_i to the state q_j upon reading a ; i.e., $U_{ij} = 1$ if and only if $\delta(q_i, a) = q_j$.

1nfa: as above, except that U is boolean with $U_{ij} = 1$ if and only if $q_j \in \delta(q_i, a)$.

1pfa: $\zeta \in [0, 1]^n$ is a stochastic row vector representing the *initial probability distribution* of the states,¹ and U is an $n \times n$ stochastic transition matrix with U_{ij} being the *probability* that A moves from the state q_i to the state q_j upon reading a , i.e., $U_{ij} = \delta(q_i, a, q_j)$.

The reader may easily work out the matrix presentation for the unary 1dfa and the unary 1pfa defined, respectively, in Examples 2 and 4.

Let us see how to express the notion of accepted language in this matrix presentation. The situation of the unary one-way finite automata A at the end its the computation on the input word a^k is described by the vector ζU^k having the following meaning (recall that η is the characteristic vector of the final states of A):

A is a 1dfa: ζU^k is the characteristic vector of the state reached by A at the end of the computation on a^k . Thus, the product $\zeta U^k \eta$ returns 1 if the reached state is accepting, and 0 otherwise. We say that A *accepts* a^k whenever $\zeta U^k \eta = 1$.

A is a 1nfa: ζU^k is the characteristic vector of the set of states reached by A at the end of the computation on a^k . Thus, the product $\zeta U^k \eta$ returns the number of reached accepting states. We say that A *accepts* a^k whenever $\zeta U^k \eta \geq 1$.

A is a 1pfa: ζU^k is a stochastic vector whose i th component represents the probability that A reaches the state q_i at the end of the computation on a^k . Thus, the product $p_A(a^k) = \zeta U^k \eta$ returns the probability for A to reach an accepting state at the end of the computation on a^k , i.e., *the probability that A accepts a^k* .

If A is a unary 1dfa or 1nfa, then *the accepted language* is defined as

$$L_A = \{a^k \mid k \in \mathbb{N} \text{ and } \zeta U^k \eta \geq 1\}. \tag{4}$$

Let A be a unary 1pfa. *The language accepted by A with cut point λ* is defined as

$$L_{A,\lambda} = \{a^k \mid k \in \mathbb{N} \text{ and } p_A(a^k) > \lambda\}. \tag{5}$$

As above recalled, the unary 1pfa A accepts a unary language $L \subseteq a^*$ with isolated cut point λ whenever $L = L_{A,\lambda}$ and there exists $\rho > 0$ such that $|p_A(a^k) - \lambda| \geq \rho$ for every $k \in \mathbb{N}$.

For the sake of completeness, we point out that when investigating the descriptive power of unary finite automata, we get size estimations which are slightly different than those

above quoted for finite automata working on general input alphabets. Thus, e.g., it is known that $e^{\Theta(\sqrt{n \log n})}$ states are necessary and sufficient for 1dfa's to simulate unary 1nfa's [48]. The same exponential blowup is proved in Refs. [49,50] for simulating unary 2dfa's and 2nfa's by 1dfa's. A "similar" exponential gap is also proved for simulating unary 1pfa's by 1dfa's; however, for this latter simulation the question should be stated more carefully, and we refer the reader to Refs. [51,52] for complete details. Finally, as previously recalled, we have that isolated cut point unary 2pfa's accept all and only regular languages, but their exact descriptive power is still an open question.

B. Basics of linear algebra

We briefly recall some basic notions of linear algebra (see, e.g., Ref. [53]) that are useful in the quantum picture and, in particular, to define the model of quantum finite automata. We denote by \mathbb{C} the field of complex numbers. Given a complex number $z \in \mathbb{C}$, its conjugate is denoted by \bar{z} and its modulus by $|z| = \sqrt{z\bar{z}}$. The set of $n \times m$ matrices having entries in \mathbb{C} is denoted by $\mathbb{C}^{n \times m}$. For matrices $C \in \mathbb{C}^{n \times m}$ and $D \in \mathbb{C}^{m \times r}$, their product is the matrix $(CD)_{ij} = \sum_{k=1}^m C_{ik}D_{kj}$ in $\mathbb{C}^{n \times r}$. The *adjoint* of a matrix $M \in \mathbb{C}^{n \times m}$ is the matrix $M^\dagger \in \mathbb{C}^{m \times n}$ with $M_{ij}^\dagger = \overline{M_{ji}}$. An Hilbert space of dimension n is the linear space $\mathbb{C}^{1 \times n}$ —in what follows denoted by \mathbb{C}^n for short—equipped with sum and product by elements in \mathbb{C} , in which, for any vectors $\zeta, \xi \in \mathbb{C}^n$, the *inner product* $\langle \zeta, \xi \rangle = \zeta \xi^\dagger$ is defined. If $\langle \zeta, \xi \rangle = 0$, we say that ζ and ξ are *orthogonal*. If ζ and ξ are orthogonal and $\|\zeta\| = 1 = \|\xi\|$, then ζ and ξ are said to be *orthonormal*. The *norm* of vector ζ is defined as $\|\zeta\| = \sqrt{\langle \zeta, \zeta \rangle}$. Two subspaces X, Y in \mathbb{C}^n are orthogonal if every vector in X is orthogonal to every vector in Y ; in this case, the linear space generated by $X \cup Y$ is denoted by $X \dot{+} Y$.

A matrix $M \in \mathbb{C}^{n \times n}$ is said to be *unitary* whenever $MM^\dagger = I = M^\dagger M$, where $I \in \mathbb{C}^{n \times n}$ is the identity matrix. Equivalently, M is unitary if and only if it preserves the norm, i.e., $\|\zeta M\| = \|\zeta\|$ for every $\zeta \in \mathbb{C}^n$. The eigenvalues of unitary matrices are complex numbers of modulus 1, i.e., they are in the form $e^{i\vartheta}$, for some real ϑ . A matrix $\mathcal{O} \in \mathbb{C}^{n \times n}$ is said to be *Hermitian* whenever $\mathcal{O} = \mathcal{O}^\dagger$. Let c_1, \dots, c_s be the eigenvalues of the Hermitian matrix \mathcal{O} and E_1, \dots, E_s be the corresponding eigenspaces. It is well known that (i) each eigenvalue c_k is real, (ii) E_i is orthogonal to E_j for every $i \neq j$, and (iii) $E_1 \dot{+} \dots \dot{+} E_s = \mathbb{C}^n$. Each vector $\zeta \in \mathbb{C}^n$ can be uniquely decomposed as $\zeta = \zeta_1 + \dots + \zeta_s$, where $\zeta_j \in E_j$. The linear transformation $\zeta \mapsto \zeta_j$ is the *projector* $P_j \in \mathbb{C}^{n \times n}$ on the subspace E_j . It is easy to see that $\sum_{j=1}^s P_j = I$. An Hermitian matrix \mathcal{O} is one-to-one determined by its eigenvalues and its eigenspaces (or, equivalently, by its projectors). In fact, we have $\mathcal{O} = c_1 P_1 + \dots + c_s P_s$.

C. Axiomatic for quantum mechanics in short

Here, we use the elements of linear algebra discussed so far to describe quantum systems (see, e.g., Refs. [54,55] for detailed expositions). Given a set $Q = \{q_1, \dots, q_m\}$ of *basis states*, every q_i can be represented by its characteristic vector $e_i \in \{0, 1\}^m$ having 1 at i th position and 0 elsewhere. A *quantum state* on Q is a superposition $\zeta \in \mathbb{C}^m$ of basis

¹The definition of a 1pfa previously given admits a single initial state q_0 instead of assigning to each control state the probability of being initial. It can be shown that the two definitions of a 1pfa are actually equivalent from both a computational and a descriptive point of view.

states of the form $\zeta = \sum_{k=1}^m \alpha_k e_k$, with coefficients α_k being complex *amplitudes* satisfying $\|\zeta\| = 1$. Given an alphabet $\Sigma = \{a_1, \dots, a_l\}$ of events, with every event symbol a_i we associate a unitary transformation $U(a_k) : \mathbf{C}^m \rightarrow \mathbf{C}^m$. An *observable* is described by an Hermitian matrix $\mathcal{O} = c_1 P_1 + \dots + c_s P_s$. Suppose that at a given instant a quantum system is described by the quantum state ζ . Then, we can operate the following:

(1) *Evolution by the event a_j* . The new state $\xi = \zeta U(a_j)$ is reached. This dynamics is *reversible*, meaning that $\zeta = \xi U^\dagger(a_j)$.

(2) *Measurement of \mathcal{O}* . Every outcome in $\{c_1, \dots, c_s\}$ can be obtained. The outcome c_j is obtained with probability $\|\zeta P_j\|^2 = \langle \zeta P_j, \zeta P_j \rangle$, and the state of the quantum system after observing such a measurement collapses to the superposition $\zeta P_j / \|\zeta P_j\|$. The state transformation induced by a measurement is typically *irreversible*.

D. One-way unary quantum finite automata

Several models of one-way (fully) quantum finite automata are proposed in the literature. Basically, they differ in measurement policy [22,24,25,44]. In this paper, we consider the simplest model of one-way quantum automata called *measure once* [18–21]. We focus on the unary case, i.e., automata having a single-letter input alphabet $\Sigma = \{a\}$. Indeed, the definition of a one-way quantum automata on a general alphabet comes straightforwardly. As done in Sec. II A for classical models of unary one-way finite automata, we are going to provide a matrix presentation of unary one-way quantum finite automata.

A unary *measure-once one-way quantum finite automaton* (1qfa, for short) with n basis states, some of which are designated as *accepting* states, is formally defined by the triple $A = (\zeta, U, P)$, where the following hold:

(i) $\zeta \in \mathbf{C}^n$, with $\|\zeta\| = 1$, is the initial superposition of basis states.

(ii) $U \in \mathbf{C}^{n \times n}$ is a unitary transition matrix with U_{ij} being the *amplitude* that A moves from the basis state q_i to the basis state q_j upon reading a , so that $|U_{ij}|^2$ is the *probability* of such a transition.

(iii) $P \in \mathbf{C}^{n \times n}$ is the projector onto the *accepting* subspace, i.e., the subspace of \mathbf{C}^n spanned by the accepting basis states. The projector P represents the observable $\mathcal{O} = 1 \cdot P + 0 \cdot (I - P)$.

At the end of the computation on the input word a^k , the state of A is described by the final superposition ζU^k . At this point, the observable \mathcal{O} is measured, and A is observed in an accepting basis state with probability $p_A(a^k) = \|\zeta U^k P\|^2$. This is the *probability that A accepts a^k* .

The definition of the unary language $L_{A,\lambda}$ accepted by A with cut point λ and the notion of a unary language accepted by A with isolated cut point are identical to those provided in Sec. II A for the model of unary 1pfa's.

The designation “measure once” given to the model of 1qfa above introduced is due to the fact the observation for acceptance is performed only once, at the end of input processing. Throughout the rest of the paper, for the sake of

brevity, by 1qfa we will mean “measure-once 1qfa,” unless otherwise stated.

Several contributions in the literature show that, surprisingly enough, isolated cut point 1qfa's are less powerful than classical models of one-way finite automata. In fact, Refs. [18,20,21] prove the following:

Theorem 6. The class of languages on general alphabets accepted by isolated cut point 1qfa's coincides with the class of group languages [56], a *proper* subclass of regular languages.

This limitation still remains for more general variants of (fully) quantum finite automata [22,24,25,57]. To overcome this computational weakness and exactly reach classical acceptance capability, hybrid models are proposed in the literature, consisting of classical finite automata “embedding” small quantum finite memory components (see, e.g., [24,28,35,58–60]).

By restricting to *unary* alphabets, the computational power of isolated cut point 1qfa's still remains strictly lower than that of classical devices. On the other hand, it is proved in Ref. [61] that the class of unary languages accepted by “measure-many” isolated cut point 1qfa's coincides with the class of unary regular languages. Roughly speaking, a measure-many 1qfa [24,25,57] is defined as a measure-once 1qfa, but the observation for acceptance is performed at each step along the computation.

III. THEORETICAL DESIGN OF A SMALL QUANTUM FINITE AUTOMATON

Although being computationally weaker, 1qfa's may greatly outperform classical devices when *size*—customarily measured by the *number of basis states*—is considered (see, e.g., Refs. [18,30,33,34,36,38,62–66]). To prove this fact, we test the descriptive power of several models of classical and quantum one-way finite automata on the very simple benchmark language introduced in Example 2: For any given integer $m > 0$, we let the unary language

$$L_m = \{a^k \mid k \in \mathbf{N} \text{ and } k \bmod m = 0\}. \quad (6)$$

Despite its simplicity, this language proves to be particularly size consuming on the classical model of one-way finite automata, as shown in the following:

Theorem 7. For any integer $m > 0$, let $m = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_s^{\alpha_s}$ be its integer factorization, for primes p_i and positive integers α_i . To accept the language L_m , the following number of states are necessary and sufficient:

- (i) m states on $1\{d,n\}$ fa's, and
- (ii) $p_1^{\alpha_1} + p_2^{\alpha_2} + \dots + p_s^{\alpha_s}$ states on $2\{d,n\}$ fa's and isolated cut point 1pfa's.

Proof. (i) In Example 2, an m -state 1dfa (which is clearly a particular 1nfa) for L_m is provided. The fact that m states are necessary for any $1\{d,n\}$ fa to accept L_m can be easily obtained by using the pumping lemma for regular languages [39,40].

(ii) For $2\{d,n\}$ fa's, the result is proved in [49]. For 1pfa's, the result is proved in Ref. [65]. ■

By adopting the quantum paradigm, we can obtain isolated cut point 1qfa's for L_m of incredibly small size.

Theorem 8. For any integer $m > 0$, the language L_m can be accepted by an isolated cut point 1qfa with *two* basis states.

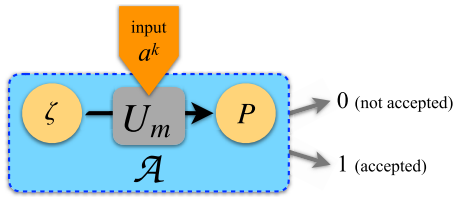


FIG. 5. Scheme of the 1qfa \mathcal{A} accepting the language L_m : Given the initial automaton state ζ and the input word a^k the automaton outputs “1” (accepted) or “0” (not accepted). See the text for details.

Proof. We define the 1qfa \mathcal{A} with 2 basis states as

$$\begin{aligned} \mathcal{A} &= \left(\zeta = (1, 0), \right. \\ U_m &= \begin{pmatrix} \cos(\pi/m) & \sin(\pi/m) \\ -\sin(\pi/m) & \cos(\pi/m) \end{pmatrix}, \\ P &= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \left. \right). \end{aligned} \quad (7)$$

One may easily verify that U is a unitary matrix and that

$$(U_m)^k = \begin{pmatrix} \cos(\pi k/m) & \sin(\pi k/m) \\ -\sin(\pi k/m) & \cos(\pi k/m) \end{pmatrix}. \quad (8)$$

Straightforward calculations show that the probability that \mathcal{A} accepts the word a^k amounts to

$$\begin{aligned} p_{\mathcal{A}}(a^k) &= \|\zeta(U_m)^k P\|^2 = \cos^2\left(\frac{\pi k}{m}\right) \\ &= \begin{cases} 1 & \text{if } k \bmod m = 0 \\ < \cos^2(\pi/m) & \text{otherwise.} \end{cases} \end{aligned} \quad (9)$$

In other words, our 1qfa \mathcal{A} accepts with certainty the words in L_m , while the acceptance probability for the words not in L_m is bounded above by $\cos^2(\pi/m) < 1$.

So, we can set the cut point $\lambda = [1 + \cos^2(\pi/m)]/2$ and isolation $\rho = [1 - \cos^2(\pi/m)]/2$, and conclude that L_m is accepted by the 1qfa \mathcal{A} with two basis states and cut point λ isolated by ρ . ■

In Fig. 5, we depict the 1qfa \mathcal{A} of Eq. (7) in order to highlight the input word a^k , the initial automaton state ζ , the unitary operator U_m , and the measurement described by the projector P .

It is worth noting that the isolation $\rho = [1 - \cos^2(\pi/m)]/2$ around the cut point of the 1qfa \mathcal{A} of Eq. (7) tends to 0 for $m \rightarrow +\infty$. Hence, as m grows, so does the error probability, i.e., with high probability \mathcal{A} may erroneously accept (reject) words not in L_m (words in L_m). To overcome this lack of precision, several modular design frameworks have been settled in the literature, aiming at enlarging cut point isolation paying by increasing the number of basis states [25,31–38]. Within these frameworks, for *any desired* isolation $\rho > 0$, a 1qfa can be theoretically defined, which accepts L_m with cut point isolated by ρ and featuring $O(\frac{\log m}{\rho})$ basis states. Although the number of basis states now depends on m , still it remains exponentially lower than the number of states of equivalent classical one-way finite automata displayed in Theorem 7. In addition, the proposed

$O(\frac{\log m}{\rho})$ -state 1qfa turns out to be the *smallest possible*. In fact, in Ref. [34] it is proved that any 1qfa accepting L_m with cut point isolation ρ must have at least $\frac{\log m}{\log[1+2/\rho]}$ basis states.

It should be stressed that all the design frameworks proposed in the literature, aiming to build extremely succinct 1qfa’s not only for L_m but also for more general families of languages, use the simple 1qfa \mathcal{A} of Eq. (7) as a crucial *building block*. Within these frameworks, the 1qfa \mathcal{A} is suitably composed in a modular pattern by using traditional compositions (i.e., direct product and sum of quantum systems), in order to enhance *precision* in language recognition. In particular, from this perspective, a physical realization of the 1qfa \mathcal{A} is not only interesting *per se* but it may provide a concrete computational component upon which to physically project more sophisticated and precise 1qfa’s by traditional compositions of quantum systems.

IV. PHOTONIC IMPLEMENTATION OF THE QUANTUM FINITE AUTOMATON

In this section, we describe the physical implementation of the 1qfa \mathcal{A} of Eq. (7). The experimental realization is based on the polarization degree of freedom of single photons and their manipulation through suitable rotators of polarization. For the sake of clarity, before discussing the physical implementation, we will summarize in the following the basic formalism used to describe this kind of quantum system.

A. The Dirac formalism

In order to describe the physical implementation of the 1qfa \mathcal{A} of Eq. (7) accepting the language L_m , it is useful to review the standard notation for quantum mechanics introduced by Dirac [54]. This will help the reader to easily pass from the notation used in the previous sections to the one we will use in the following. In this notation, the state “ ψ ” of a quantum system is described by the symbol $|\psi\rangle$ which is, in general, a complex column vector in a Hilbert space. In the present work, we are interested in the (linear) polarization state of a single photon; therefore, only the two basis states $|H\rangle$ and $|V\rangle$, referring to the horizontal (H) and vertical (V) polarization, respectively, are needed. Indeed, because of the very law of quantum mechanics, any normalized linear combination of these two vectors represents a quantum state. For instance, a single photon polarized at an angle θ with respect to the horizontal is described by the state vector

$$|\theta\rangle = \cos\theta |H\rangle + \sin\theta |V\rangle. \quad (10)$$

Since we are in the presence of only two basis states, we can give a geometrical representation of them and of the corresponding spanned space, as shown in Fig. 6(a).

In this formalism, it is clear the correspondence

$$|H\rangle = \zeta^\dagger = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \text{and } \langle H| = (|H\rangle)^\dagger = \zeta = (1, 0), \quad (11)$$

where ζ is the same state introduced in Eq. (7). Analogously, we have

$$|V\rangle = \xi^\dagger = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{and} \quad |\theta\rangle = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \quad (12)$$

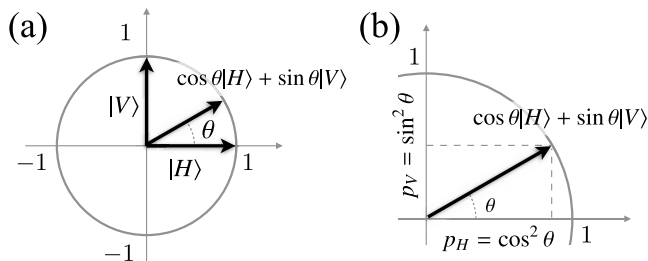


FIG. 6. (a) Two-dimensional representation of the polarization states $|H\rangle$ (horizontal polarization) and $|V\rangle$ (vertical polarization) of a single photon. We also report the representation of the single-photon state with (linear) polarization at the angle θ . (b) The square of the projections along the horizontal and the vertical axes correspond to the probability of finding the photon with horizontal and vertical polarization, respectively.

In Sec. II B, we introduced the inner product $\langle \zeta, \xi \rangle = \zeta \xi^\dagger$ between the states ζ and ξ . Using the Dirac formalism, we have

$$\zeta \xi^\dagger = \langle H|V \rangle = 0, \tag{13}$$

where we also used the orthonormality of the involved states. If we now introduce the projectors

$$\Pi_H = |H\rangle\langle H| = P = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \tag{14}$$

where P is the same as in Eq. (7), and

$$\Pi_V = |V\rangle\langle V| = Q = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \tag{15}$$

given the state $|\theta\rangle$, with $(|\theta\rangle)^\dagger = \vartheta$, we have

$$p_H = \langle \vartheta P, \vartheta P \rangle = \langle \vartheta | \Pi_H | \theta \rangle = |\langle H | \theta \rangle|^2 = \cos^2 \theta, \tag{16a}$$

$$p_V = \langle \vartheta Q, \vartheta Q \rangle = \langle \vartheta | \Pi_V | \theta \rangle = |\langle V | \theta \rangle|^2 = \sin^2 \theta, \tag{16b}$$

where we used $\Pi_J^2 = \Pi_J$, with $J \in \{H, V\}$ and $\langle a|b \rangle = \overline{\langle b|a \rangle}$. The geometrical meanings of p_H and p_V are reported in Fig. 6(b), where, from the physical point of view, they correspond to the probability of finding the photon with horizontal or vertical polarization, respectively.

In the context of the polarization of single photons, the analog of the unitary operator U_m defined in Eq. (7) is the operator $R(\pi/m)$ which corresponds to a rotator of polarization, which rotates the polarization of the photons by an amount π/m . We can write $R(\pi/m) = U_m^\dagger$. Thereafter, the one-step evolution of the state $|H\rangle = \zeta^\dagger$ reads

$$R(\pi/m)|H\rangle = \zeta U_m. \tag{17}$$

B. Photonic quantum automaton

In Fig. 7, we depicted the basic elements of the photonic quantum automaton implementing the 1qfa \mathcal{A} of Eq. (7) accepting the language L_m . Given the input word a^k (see also Fig. 5), a single photon, generated in the state $|H\rangle$ is sent through k rotators of polarization, where each rotator applies a rotation of a fixed amount π/m . It is worth noting that in order to actually reproduce the computation of a 1qfa, a single rotation should be applied step by step upon reading

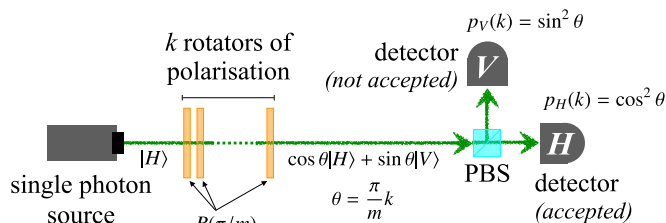


FIG. 7. Sketch of the photonic implementation of the 1qfa \mathcal{A} accepting the language L_m . Single photons are generated in the polarization state $|H\rangle$ and then they pass through k polarization rotators, k being the length of the input word a^k . Each rotator implements the operator $R(\pi/m)$ rotating the polarization by the amount π/m : The overall polarization rotation is $\theta = \pi k/m$. Finally, the photons are addressed to two photodetectors by means of a polarizing beam splitters (PBS) according to their horizontal (H) or vertical (V) polarization.

each input symbol, since the input word length is not known in advance. After the rotators, the single photon is sent to a polarizing beam splitter (PBS), a device which transmits (reflects) the horizontal (vertical) polarization component of the input state. Since after the rotators the state of the photon is $|\theta\rangle$, given in Eq. (10), it is detected by the H or V detector (see Fig. 7) with the probabilities given in Eqs. (16). It is worth noting that, as expected, $p_H(k)$ is equal to the automaton acceptance probability $p_{\mathcal{A}}(a^k)$; see Eq. (9). As mentioned in Theorem 8, this kind of automaton accepts *with certainty* the word a^k if $k \bmod m = 0$, but it has also a high error probability to accept the word if $k \bmod m = 1$. In fact, in this case, $p_H(k)$ attains its maximum $\cos^2(\pi/m)$.

To reduce the error probability, one can send $M = N_c(m)$ copies of the same input word a^k , collect the number $N_c(k)$ of counts at the detector H , and evaluate the ratio

$$f_k = \frac{N_c(k)}{N_c(m)} \xrightarrow{M \gg 1} p_H(k). \tag{18}$$

In this scenario, we let $f_1 = f_{(k \bmod m)=1}$ be the highest frequency less than $f_0 = f_{(k \bmod m)=0} = 1$. That is, f_1 is the highest frequency for words that are erroneously accepted (those words a^k for which $k \bmod m = 1$), and f_0 is the frequency of those words that are correctly accepted (those words a^k for which $k \bmod m = 0$). Thus, we can define the threshold frequency

$$f_{\text{th}} = \frac{f_0 + f_1}{2} = \frac{1 + f_1}{2}, \tag{19}$$

and we use the following strategy:

- if $f_k > f_{\text{th}} \Rightarrow a^k$ is accepted by \mathcal{A} ,
- if $f_k < f_{\text{th}} \Rightarrow a^k$ is rejected. (20)

It is clear that such a strategy leads to a zero error probability; namely, all and only the words in L_m can have $f_k > f_{\text{th}}$. However, in a realistic scenario the number of detected photons is subjected to Poisson statistical fluctuations, due to the very nature of the detection process [67]. So, given the word a^k , the number of detected counts $N_c(k)$ fluctuates according to a Poisson distribution with mean $\mu_k = \langle N_c \rangle \cos^2(\pi k/m)$, where $\langle N_c \rangle$ is the *average* number of detected photons obtained for

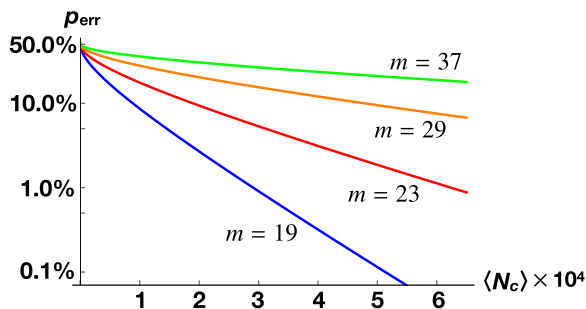


FIG. 8. Error probability (wrong acceptance probability) of the 1qfa \mathcal{A} accepting the language L_m , for different values of m , as a function of the average number of counts $\langle N_c \rangle$.

$k \bmod m = 0$. Thus, it is possible to have a *detected frequency* $\tilde{f}_k = N_c(k)/\langle N_c \rangle$ which incorrectly satisfies

$$\tilde{f}_k > \tilde{f}_{th} = (\tilde{f}_0 + \tilde{f}_1) / 2 \quad [\text{resp.}, \tilde{f}_k < \tilde{f}_{th} = (\tilde{f}_0 + \tilde{f}_1) / 2]$$

also for a word a^k not belonging (resp., belonging) to the language L_m , leading to a non-null experimental acceptance error probability p_{err} .

If we assume $\mu_1 = \langle N_c \rangle \cos^2(\pi/m) \gg 1$, the distribution of the detected number of counts for $k \bmod m = 1$ can be approximated by a Gaussian distribution function with mean and variance given by same value μ_1 . Analogously, for $k \bmod m = 0$ we have a Gaussian distribution with mean and variance equal to $\mu_0 = \langle N_c \rangle$. Now we can find a more suitable threshold N_{th} of the detected counts by considering the intersection between the two Gaussians, namely

$$N_{th} = \langle N_c \rangle \left| \cos(\pi/m) \right| \sqrt{1 - \frac{\ln[\cos^2(\pi/m)]}{\langle N_c \rangle \sin^2(\pi/m)}}, \quad (21)$$

and the corresponding discrimination strategy reads

$$\begin{aligned} \text{if } N_c(k) \geq N_{th} &\Rightarrow a^k \text{ is accepted by } \mathcal{A}, \\ \text{if } N_c(k) < N_{th} &\Rightarrow a^k \text{ is rejected.} \end{aligned} \quad (22)$$

The experimental error probability is thus given by (we consider only the two relevant contributions)

$$\begin{aligned} p_{\text{err}} &= \int_{-\infty}^{N_{th}} \frac{dx}{\sqrt{2\pi\mu_0}} \exp\left[-\frac{(x-\mu_0)^2}{2\mu_0}\right] \\ &+ \int_{N_{th}}^{\infty} \frac{dx}{\sqrt{2\pi\mu_1}} \exp\left[-\frac{(x-\mu_1)^2}{2\mu_1}\right], \end{aligned} \quad (23)$$

where $1 \ll \langle N_c \rangle \cos^2(\pi/m) = \mu_1 < N_{th} < \mu_0 = \langle N_c \rangle$. We note that p_{err} corresponds to the probability of accepting (resp., rejecting) the word a^k whenever it should be rejected (resp., accepted). In Fig. 8, we plot the error probability for different values of m : As one may expect, as m increases so should the average number of counts $\langle N_c \rangle$ in order to have a small error probability.

V. EXPERIMENTAL RESULTS

The main elements of our physical implementation of the 1qfa \mathcal{A} accepting the language L_m are sketched in Fig. 7. However, in order to reduce the losses and other sources of

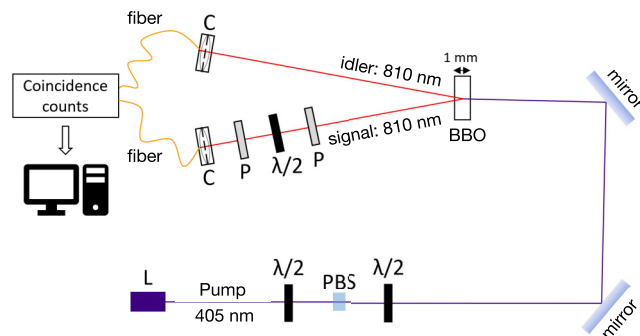


FIG. 9. Schematic diagram of the experimental setup. A 405-nm continuous wave (cw) laser diode (L) generates a pump beam which passes through an amplitude modulator, composed of a half-wave plate ($\lambda/2$) and a polarizing beamsplitter cube (PBS), and through another half-wave plate to set the polarization. The beam interacts with a 1-mm-long barium borate (BBO) crystal generating photons at 810 nm via parametric down conversion (PDC). The two beams separated by the horizontal plane are called signal and idler: On the signal's branch there are two polarizers (P) separated by a half-wave plate. Photons are finally focused into two multimode fibers through two couplers (C), and sent to homemade single-photon counting modules.

noise, in the actual setup we replace the action of the k polarization rotators on the input word a^k by using a single rotator applying an overall rotation of $\theta = \pi k/m$, which “simulates” the whole computation of the 1qfa: For this reason, we will refer to our system as a *photonic quantum simulator* [68] of the quantum automaton. As mentioned in the previous section, an actual 1qfa does not have an *a priori* knowledge about the length k of the input word. In fact, it reads the input word symbol by symbol while applying a rotation π/m per each scanned input symbol a . Practically, this can be implemented, for instance, by a motorized rotator of polarization, but this is beyond the scope of the present work. Nevertheless, it is worth noting that a more advanced technology, e.g., based on integrated optics or optoelectronics, can be used to realize the very setup of Fig. 7.

The experimental setup is shown in Fig. 9.

(1) The pump derives from a 405-nm cw InGaN laser diode, which we chose in order to use detectors in silicon, the ones with the lowest noise on the market: Indeed, these work with maximum quantum efficiency at 810 nm, which is the same wavelength of the photons generated via parametric down conversion (PDC) from a 405-nm pump.

(2) The laser beam passes through an amplitude modulator composed by a half-wave plate and a polarizing beamsplitter cube (PBS), and then through another half-wave plate to set the polarization vertical with respect to the optical bench.

(3) The interaction between the pump and a 1-mm-long BBO crystal generates photons at 810 nm with horizontal polarization, along the surface of a cone, via type-I-*eo* PDC: For this purpose, the optical axis of the crystal is on the vertical plane at the phase-matching angle.

(4) The intersection of the cone with the horizontal plane distinguishes two beams (branches): the signal and the idler. It is possible to finely tune the angle of the outgoing photons by properly rotating the principal axis of the BBO.

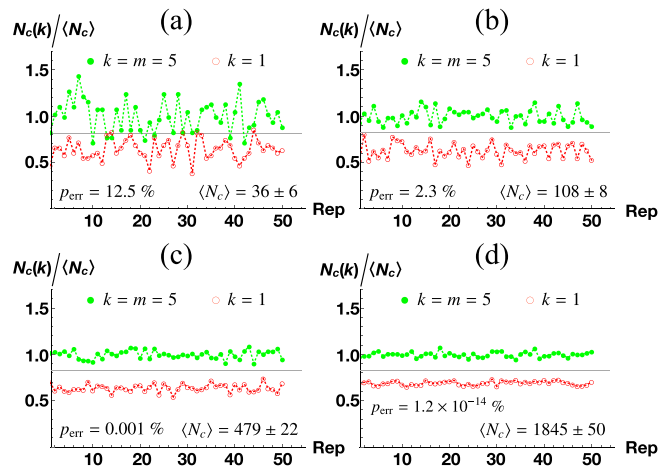


FIG. 10. Experimental ratio $N_c(k)/\langle N_c \rangle$ with $k = m = 5$ (green disks) and $k = 1$ (red circles) as a function of the experimental run number (Rep), k being the length of the input word a^k of the Iqfa \mathcal{A} accepting the language L_m . We considered different values of the overall average number of counts: (a) $\langle N_c \rangle = 36$, (b) $\langle N_c \rangle = 108$, (c) $\langle N_c \rangle = 479$, and (d) $\langle N_c \rangle = 1845$. The horizontal lines are the threshold values N_{th} given in Eq. (21). In the plots, we also report the theoretical error probability from Eq. (23). The reduction of the relative statistical fluctuations is evident.

(5) Along the signal branch, a polarizer ensures the transmission of the horizontally polarized photons, then a half-wave plate is used to simulate the k polarization rotators, and finally another horizontal polarizer transmits the photons to the detector. This last half-wave plate can be manually rotated and is equipped with graduations where a unit corresponds to 4° in polarization: By considering the working principle of the half-wave plate, this can be obtained by actually rotating the plate by 2° . Therefore, in general, in order to obtain a rotation in polarization of amount θ , one should rotate the plate by $\theta/2$.

(6) On each branch, photons are finally focused into a multimode fiber and sent to a homemade single-photon counting module, based on an avalanche photodiode operated in Geiger mode with passive quenching [69]. We chose to measure the coincidence counts in order to obtain a better signal-noise ratio: Indeed, the photodiodes produce a thermal background such that approximately 1% of the direct counts are dark counts, while the coincidence dark counts are only 0.001% of the coincidence counts.

In Fig. 10, we show typical experimental results from our photonic simulator of the Iqfa \mathcal{A} for the language L_m , with $m = 5$ (this choice allows us to put better in evidence the role of the statistical fluctuations of the detected number of photons). In this case, a single rotation of polarization (taking place, e.g., on the input word of length $k = 1$) has $\theta = 36^\circ$, which corresponds to rotating by 9 units the half-wave plate on signal's branch [see point (5) in the above description of our experimental setup].

Here we only show the interesting results for input words a^k of length $k = 5$ and $k = 1$. These two inputs, respectively representing a word in L_5 and one of the most prone to error classification words *not* in L_5 , turn out to be critical for testing the accuracy of the discrimination strategy we use. Further-

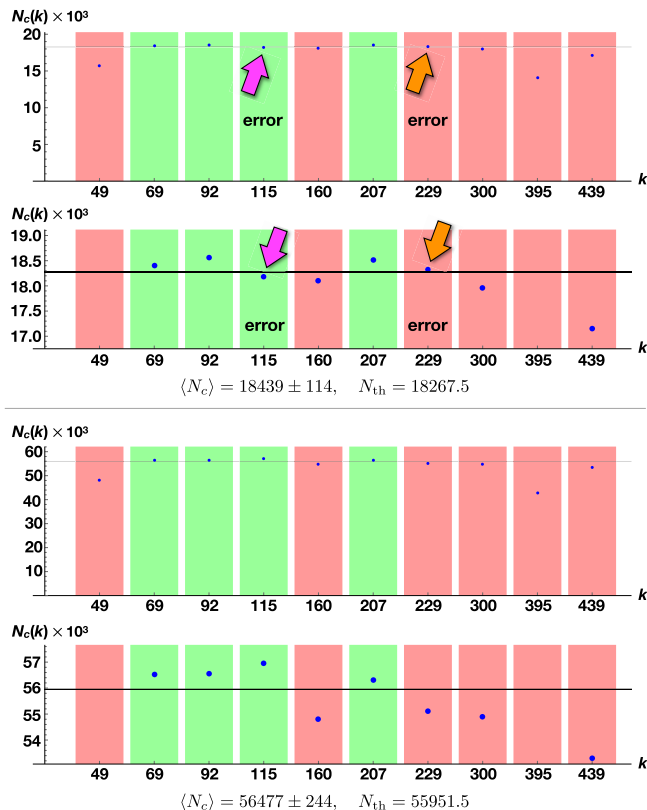


FIG. 11. Examples of the experimental number of counts $N_c(k)$ (dots) as a function of the length k of the input word a^k (we have chosen 10 values of k randomly in the interval $[1, 500]$). The horizontal lines refer to the threshold N_{th} on the number of counts for the discrimination strategy (see the text for details): If $N_c(k) \geq N_{th}$ the word a^k is accepted. The color of the vertical bars refers to the theoretical acceptance (green) or rejection (red) of the corresponding input a^k by the Iqfa \mathcal{A} accepting the language L_m , with $m = 23$. The average number of counts is $\langle N_c \rangle = 18439 \pm 114$ (top panel, corresponding to $N_{th} = 18267.5$) and $\langle N_c \rangle = 56477 \pm 244$ (bottom panel, leading to $N_{th} = 55951.5$). In both panels, the lower plots are a magnification of the region around the threshold N_{th} . (Top panel) In this case, the error probability evaluated from Eq. (23) is $p_{err} = 10.3\%$. The number 229 is accepted according to our strategy, since the number of counts (see the orange arrows) is larger than the threshold (horizontal solid line), but it should be rejected; on the other hand, the number 115 should be accepted but it is rejected since the number of counts (see the magenta arrows) is below the threshold. (Bottom panel) Here the error probability evaluated from Eq. (23) is $p_{err} = 1.3\%$. Now the number 229 is rejected and the number 115 is accepted, according to the definition of L_m . See the text for details.

more, in order to highlight the reduction of the statistical fluctuations, we plot the ratio $N_c(k)/\langle N_c \rangle$. Each point corresponds to the number of counts at the detector H when the average total number of counts is $\langle N_c \rangle = 36, 108, 479$, and 1845 , respectively, which can be obtained varying pump's power by rotating the half-wave plate of the amplitude modulator. We repeated the experiments 50 times with an acquisition time of 1s for each of the two values of k . It is clear that increasing $\langle N_c \rangle$ reduces the relative fluctuations and thus the error probability decreases accordingly.

To better appreciate the performance of our photonic simulator, we consider our 1qfa \mathcal{A} accepting the language L_m , with $m = 23$. In this case, a single rotation of polarization (taking place, e.g., on the input word of length $k = 1$) has (approximately) $\theta = 8^\circ$, which corresponds to 2 units in the half-wave plate's scale [see point (5) in the above description of our experimental setup]. In the top panel of Fig. 11, we report examples of the number of counts $N_c(k)$ for just one given experimental run as a function of different values k of the length of the input word a^k and for $\langle N_c \rangle = 18439$. The acquisition time of each point is 10s. We can see that, due to the statistical fluctuations mentioned above, sometimes the automaton fails to accept the word: This is the case for the input lengths 115 and 229, as discussed in the figure caption. We remark that in the latter cases we have chosen two particular experimental runs in which the automaton fails: If we had considered the average over many runs, we would have found that the automaton always succeeds *on average*, since the standard deviation, due to the statistical scaling, can be reduced at will. Of course, given a particular run, the error is independent of k , but depends only on the random, statistical fluctuations, which can be controlled by increasing $\langle N_c \rangle$, as we can see in the bottom panel of Fig. 11, where we report the results of the photonic simulator taking the same words of the top panel as inputs but with $\langle N_c \rangle = 56477$, obtained with an acquisition time of 30s. This can be also understood by considering Eq. (23): The error probability is reduced from $p_{\text{err}} = 10.3\%$ for $\langle N_c \rangle = 18439$ of the previous case to the current $p_{\text{err}} = 1.3\%$ for $\langle N_c \rangle = 56477$.

VI. CONCLUSIONS

We have suggested and demonstrated a photonic realization of quantum finite automata able to recognize a well-known family of unary periodic languages. Our device exploits the polarization degree of freedom of single photons and their manipulation through linear optical elements. In particular, we have designed and implemented a one-way quantum finite automaton \mathcal{A} accepting the unary language $L_m = \{a^k \mid k \in \mathbb{N} \text{ and } k \bmod m = 0\}$ with only two basis states and isolated cut point. Notice that any classical finite

automaton for L_m requires a number of states which grows with m . We have implemented the quantum finite automaton \mathcal{A} using the polarization degree of freedom of a single photon and have exploited a discrimination strategy to reduce the acceptance error probability.

It is worth noting that, for the particular one-way quantum finite automaton we considered, we exploited only the polarization degree of freedom of (quantized) optical fields and photodetection. Therefore, one can implement a similar automaton also exploiting polarization of a classical coherent field (a laser beam) and intensity measurements. Nevertheless, our experiment uses single photons that are intrinsically quantum objects, and thus it paves the way for more complex quantum finite automata we are planning to address and which exploit genuine quantum resources, such as entanglement. In fact, the quantum technology employed in our implementation is the same used in the current quantum information processing setups based on optical states.

Besides being interesting in itself for fundamental reasons, our physical realization of the one-way quantum finite automaton \mathcal{A} provides a concrete implementation of a small quantum computational component that can be used to physically build more sophisticated and precise quantum finite automata. Indeed, several modular design frameworks have been modeled and widely investigated from a theoretical point of view [24,25,30–38,65] to build succinct and precise quantum finite automata performing different tasks, where the module \mathcal{A} plays a crucial role. Within these frameworks, by suitably assembling a sufficient number of \mathcal{A} -like modules via traditional compositions of quantum systems (i.e., direct products and sums), the existence of succinct and precise quantum finite automata has been theoretically shown. From this perspective, our results are instrumental to a deeper understanding of possible physical implementations of these design frameworks by means of photonic technology and pave the way for the construction of other more powerful models of quantum finite automata.

ACKNOWLEDGMENT

M. G. A. Paris is member of GNFM-INdAM. C. Mereghetti and B. Palano are members of GNCS-INdAM.

-
- [1] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, Strength and weakness of quantum computing, *SIAM J. Comput.* **26**, 1510 (1997).
 - [2] E. Bernstein and U. Vazirani, Quantum complexity theory, *SIAM J. Comput.* **26**, 1411 (1997); A preliminary version appeared in *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)* (ACM Press, New York, NY, 1993).
 - [3] J. Gruska, *Quantum Computing* (McGraw-Hill, New York, 1999).
 - [4] M. Hirvensalo, *Quantum Computing* (Springer, Berlin, 2004).
 - [5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, UK, 2011).
 - [6] A. S. Holevo, Some estimates of the information transmitted by quantum communication channels, *Prob. Peredachi Inf.* **9**, 3 (1973) [*Prob. Inf. Transm.* **9**, 177 (1973)].
 - [7] R. S. Ingarden, Quantum information theory, *Rep. Math. Phys.* **10**, 43 (1976).
 - [8] R. Feynman, Simulating physics with computers, *Int. J. Theor. Phys.* **21**, 467 (1982).
 - [9] Y. I. Manin, *Vychislimoe i nevychislimoe (Computable and Non-computable)*, Kibernetika (Sovetskoe Radio, Moskwa, 1980) (in Russian).
 - [10] P. Benioff, Quantum mechanical Hamiltonian models of Turing machines, *J. Stat. Phys.* **29**, 515 (1982).
 - [11] D. Deutsch, Quantum theory, the Church-Turing principle, and the universal quantum computer, *Proc. R. Soc. London, Ser. A* **400**, 97 (1985).
 - [12] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26**, 1484 (1997); A preliminary version appeared in *Proceedings of the 35th IEEE Symposium on Foundations of Computer*

- Science (FOCS)* (IEEE Computer Society Press, Los Alamitos, CA, 1994).
- [13] L. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC)* (ACM Press, New York, NY, 1996), pp. 212–219.
- [14] I. L. Chuang and Y. Yamamoto, Simple quantum computer, *Phys. Rev. A* **52**, 3489 (1995).
- [15] D. P. DiVincenzo, The physical implementation of quantum computation, *Fortschr. Phys.* **48**, 771 (2000).
- [16] M. Nakahara, S. Kanemitsu, M. M. Salomaa, and S. Takagi, editors, *Physical Realizations of Quantum Computing* (World Scientific, Singapore, 2006).
- [17] S. Feld and C. Linnhoff-Popien (eds.), Quantum technology and optimization problems, in *Proceedings of the First International Workshop QTOP 2019* (Springer, Berlin, 2019).
- [18] A. Bertoni, and M. Carpentieri, Regular languages accepted by quantum automata, *Inf. Comput.* **165**, 174 (2001).
- [19] A. Bertoni and M. Carpentieri, Analogies and differences between quantum and stochastic automata, *Theor. Comput. Sci.* **262**, 69 (2001).
- [20] A. Brodsky and N. Pippenger, Characterizations of 1-way quantum finite automata, *SIAM J. Comput.* **31**, 1456 (2002).
- [21] C. Moore and J. P. Crutchfield, Quantum automata and quantum grammars, *Theor. Comput. Sci.* **237**, 275 (2000).
- [22] A. Ambainis, M. Beaudry, M. Golovkins, A. Kikusts, M. Mercer, and D. Thérien, Algebraic results on quantum automata, *Theory Comp. Syst.* **39**, 165 (2006).
- [23] A. Ambainis and J. Watrous, Two-way finite automata with quantum and classical states, *Theor. Comput. Sci.* **287**, 299 (2002).
- [24] A. Ambainis and A. Yakaryilmaz, Automata and quantum computing, [arXiv:1507.01988v2](https://arxiv.org/abs/1507.01988v2).
- [25] A. Bertoni, C. Mereghetti, and B. Palano, Quantum computing: One-way quantum automata, in *Proceedings of the Seventh Conference on Developments in Language Theory (DLT)* (Springer, Berlin, 2003), pp. 1–20.
- [26] A. Bertoni, C. Mereghetti, and B. Palano, Trace monoids with idempotent generators and measure-only quantum automata, *Nat. Comput.* **9**, 383 (2010).
- [27] C. Mereghetti and B. Palano, Upper bounds on the size of one-way quantum finite automata, in *Proceedings of the Seventh Italian Conference on Theoretical Computer Science (ICTCS)* (Springer, Berlin, 2001), pp. 123–135.
- [28] S. Zheng, D. Qiu, L. Li, and J. Gruska, One-way finite automata with quantum and classical states, in *Languages Alive: Essays Dedicated to Jürgen Dassow on the Occasion of His 65th Birthday*, edited by H. Bordihn, M. Kutrib, and B. Truthe (Springer, Berlin, 2012), pp. 273–290.
- [29] M. Holzer and M. Kutrib, Descriptive complexity—an introductory survey, in *Scientific Applications of Language Methods*, edited by C. Martín-Vide (Imperial College Press, London, UK, 2010), pp. 1–58.
- [30] A. Ambainis and R. Freivalds, One-way quantum finite automata: Strengths, weaknesses, and generalizations, in *Proceedings of the 39th Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society Press, Washington, DC, 1998), pp. 332–342.
- [31] A. Ambainis and N. Nahimovs, Improved constructions of quantum automata, *Theor. Comput. Sci.* **410**, 1916 (2009).
- [32] A. Bertoni, C. Mereghetti, and B. Palano, Approximating stochastic events by quantum automata, in *Proceedings of the ERATO Conference on Quantum Information Science (ERATO)*, Kyoto, Japan, 2003), pp. 43–44.
- [33] A. Bertoni, C. Mereghetti, and B. Palano, Small size quantum automata recognizing some regular languages, *Theor. Comput. Sci.* **340**, 394 (2005).
- [34] A. Bertoni, C. Mereghetti, and B. Palano, Some formal tools for analyzing quantum automata, *Theor. Comput. Sci.* **356**, 14 (2006).
- [35] M. P. Bianchi, C. Mereghetti, and B. Palano, Complexity of promise problems on classical and quantum automata, in *Computing with New Resources, Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday*, edited by C. S. Calude, R. Freivalds, and K. Iwama (Springer, Berlin, 2014), pp. 161–175.
- [36] M. P. Bianchi, C. Mereghetti, and B. Palano, Quantum finite automata: Advances on Bertoni’s ideas, *Theor. Comput. Sci.* **664**, 39 (2017).
- [37] C. Mereghetti and B. Palano, On the size of one-way quantum finite automata with periodic behaviors, *Theor. Inf. Appl.* **36**, 277 (2002).
- [38] C. Mereghetti and B. Palano, Quantum automata for some multiperiodic languages, *Theor. Comput. Sci.* **387**, 177 (2007).
- [39] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, New York, 2001).
- [40] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, New York, 1979).
- [41] A. Paz, *Introduction to Probabilistic Automata* (Academic Press, New York, 1971).
- [42] M. Rabin and D. Scott, Finite automata and their decision problems, *IBM J. Res. Develop.* **3**, 114 (1959).
- [43] M. O. Rabin, Probabilistic automata, *Inf. Control* **6**, 230 (1963).
- [44] J. Gruska, Descriptive complexity issues in quantum computing, *J. Auto., Lang. Comb.* **5**, 191 (2000).
- [45] C. Dwork and L. Stockmeyer, A time complexity gap for two-way probabilistic finite-state automata, *SIAM J. Comput.* **19**, 1011 (1990).
- [46] J. Kaneps, Regularity of one-letter languages acceptable by two-way finite probabilistic automata, in *Proceedings of the Eighth International Symposium on Fundamentals of Computation Theory (FCT)* (Springer, Berlin, 1991), pp. 287–296.
- [47] J. C. Shepherdson, The reduction of two-way automata to one-way automata, *IBM J. Res. Develop.* **3**, 198 (1959).
- [48] M. Chrobak, Finite automata and unary languages, *Theor. Comput. Sci.* **47**, 149 (1986); Corrigendum, **302**, 497 (2003).
- [49] C. Mereghetti and G. Pighizzini, Two-way automata simulations and unary languages, *J. Auto., Lang. Comb.* **5**, 287 (2000).
- [50] C. Mereghetti and G. Pighizzini, Optimal simulations between unary automata, *SIAM J. Comput.* **30**, 1976 (2001).
- [51] M. P. Bianchi and G. Pighizzini, Normal forms for unary probabilistic automata, *Theor. Inf. Appl.* **46**, 495 (2012).
- [52] M. Milani and G. Pighizzini, Tight bounds on the simulation of unary probabilistic automata by deterministic automata, *J. Auto., Lang. Comb.* **6**, 481 (2001).
- [53] M. Marcus and H. Minc, *Introduction to Linear Algebra* (Dover, New York, 1988).

- [54] P. A. M. Dirac, A new notation for quantum mechanics, *Math. Proc. Cambridge Philos. Soc.* **35**, 416 (1939).
- [55] R. I. G. Hughes, *The Structure and Interpretation of Quantum Mechanics* (Harvard University Press, Cambridge, MA, 1992).
- [56] J.-E. Pin, *Varieties of Formal Languages* (North Oxford Academic Press, London, UK, 1986).
- [57] A. Kondacs and J. Watrous, On the power of quantum finite state automata, in *Proceedings of the 38th Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society Press, Washington, DC, 1997), pp. 66–75.
- [58] M. P. Bianchi, C. Mereghetti, and B. Palano, On the power of one-way automata with quantum and classical states, *Int. J. Found. Comput. Sci.* **26**, 895 (2015).
- [59] M. Hirvensalo, Quantum automata with open time evolution, *Int. J. Natural Comput. Res.* **1**, 70 (2010).
- [60] C. Mereghetti, B. Palano, Quantum finite automata with control language, *Theor. Inf. Appl.* **40**, 315 (2006).
- [61] M. P. Bianchi and B. Palano, Behaviours of unary quantum automata, *Fundamenta Informaticae* **104**, 1 (2010).
- [62] F. Abelayev and A. Gainutdinova, On the lower bounds for one-way quantum automata, in *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS)* (Springer, Berlin, 2000), pp. 132–140.
- [63] A. Bertoni, C. Mereghetti, and B. Palano, Lower bounds on the size of quantum automata accepting unary languages, in *Proceedings of the Eighth Italian Conference on Theoretical Computer Science (ICTCS)* (Springer, Berlin, 2003), pp. 86–96.
- [64] M. P. Bianchi, C. Mereghetti, and B. Palano, Size lower bounds for quantum automata, *Theor. Comput. Sci.* **551**, 102 (2014).
- [65] C. Mereghetti, B. Palano, and G. Pighizzini, Note on the succinctness of deterministic, nondeterministic, probabilistic and quantum finite automata, *Theor. Inf. Appl.* **35**, 477 (2001).
- [66] A. Bertoni, C. Mereghetti, and B. Palano, Golomb rulers and difference sets for succinct quantum automata, *Int. J. Found. Comput. Sci.* **14**, 871 (2003).
- [67] R. Loudon, *The Quantum Theory of Light* (Oxford University Press, Oxford, UK, 2000).
- [68] S. Cialdi, M. A. C. Rossi, C. Benedetti, B. Vacchini, D. Tamascelli, S. Olivares, and M. G. A. Paris, All-optical quantum simulator of qubit noisy channels, *Appl. Phys. Lett.* **110**, 081107 (2017).
- [69] R. G. W. Brown, K. D. Ridley, and J. G. Rarity, Characterization of silicon avalanche photodiodes for photon correlation measurements, 1: Passive quenching, *Appl. Opt.* **25**, 4122 (1986).