Energy Complexity of Regular Language Recognition

Öykü Yılmaz¹, Fırat Kıyak², Meriç Üngör¹, A. C. Cem Say¹

¹Department of Computer Engineering, Boğaziçi University, İstanbul, Turkey
²Department of Mathematics, Boğaziçi University, İstanbul, Turkey

Abstract. The erasure of each bit of information by a computing device has an intrinsic energy cost. Although any Turing machine can be rewritten to be thermodynamically reversible without changing the recognized language, finite automata that are restricted to scan their input once in “real-time” fashion can only recognize the members of a proper subset of the class of regular languages in this reversible manner. We use a general quantum finite automaton model to study the thermodynamic cost per step associated with the recognition of different regular languages. We show that zero-error quantum finite automata have no energy cost advantage over their classical deterministic counterparts, and prove an upper bound for the cost that holds for all regular languages. We also demonstrate languages for which “error can be traded for energy”, i.e. whose zero-error recognition is associated with provably bigger energy cost per step when compared to their bounded-error recognition by real-time finite-memory quantum devices.

Keywords: Quantum finite automata · Reversibility

1 Introduction

The discovery of the relationship between thermodynamics and computation, revealing the links between concepts of heat, entropy, and information, is a landmark scientific achievement [10]. As shown by Landauer [9], the erasure of each bit of information by a computing device necessitates the dissipation of an amount of heat proportional to the absolute temperature of the device, and therefore has an unavoidable minimum energy cost for any fixed temperature. Turing machine programs [2] (and even finite automata with two-way access to their input strings [7]) can be rewritten to be reversible, so that each one of their configurations has a single possible predecessor, and their computational steps can therefore in principle be executed using arbitrarily small amounts of energy, but things change when one limits attention to real-time finite automata.

It is known [13] that reversible real-time finite automata (where each state has at most one incoming transition with each possible symbol of the input alphabet) recognize only a proper subset of the class of regular languages, so some regular languages necessarily have automata with states receiving multiple transitions.
with the same symbol. Intuitively, it is impossible to “rewind” computations of such machines, since they “forget” which one of a set of possible predecessor states led them to their present state. It is natural to ask if this energy-related criterion could be used to define a hierarchy whose levels are associated with the minimum values of these “in-degrees” required to recognize the languages in question.

It was precisely because of the reversibility requirement inherent in unitary matrices that early definitions of real-time quantum finite automata (QFAs) \cite{12,3,7} were not able to capture all regular languages. Modern definitions of QFAs \cite{5,14}, which recognize all and only the regular languages with bounded error, are able to handle irreversible languages by using not one but many instances of an architectural component (called an operation element) that can be seen to correspond to the notion of “incoming transitions” discussed above, so the hierarchy question raised above is relevant for the study of bounded-error QFAs as well.

In this paper, we use the general QFA model of \cite{14} which allows us to model the information loss inherent in the computations of such machines, establishing a clear link with Landauer’s principle (Section 2) to study the thermodynamic cost per step associated with the recognition of different regular languages. In Section 3 we show that zero-error quantum finite automata have no energy cost advantage over their classical deterministic counterparts. That equivalence is used in Section 4 to establish an upper bound on the number of bits that have to be “forgotten” per computational step during the recognition of any regular language, namely, any such language on an alphabet with \( k \) symbols can be recognized by a zero-error quantum finite automaton that has at most \( k + 1 \) operation elements for each input symbol, and thus requires no more than \( \log_2(k + 1) \) bits to be erased per step. In Section 5 we demonstrate languages for which “error can be traded for energy”, i.e. whose zero-error recognition is associated with provably bigger energy cost per step when compared to their bounded-error recognition by real-time finite-memory quantum devices. Section 6 lists some open questions.

2 The general QFA framework and information erasure

Although classical physics, on which the intuition underlying deterministic computation models is based, is supposed to be subsumed by quantum physics, early definitions of quantum finite automata (e.g. \cite{12,7}) resulted in “weak” machines that could only capture a proper subset of the class of regular languages. The cause of this apparent contradiction was identified \cite{5} to be those early definitions’ imposition of unnecessarily strict limitations on the interaction of the automata with their environments. Classical finite automata, after all, are not “closed” systems, with loss of information about the preceding configuration and the ensuing transfer of heat to the environment implied by their logical structure. The modern definition of QFAs to be given below \cite{5,14} allows a sufficiently
A quantum finite automaton (QFA) is a 5-tuple $(Q, \Sigma, \{ E_\sigma | \sigma \in \Sigma_{[]} \}, q_1, F)$, where

1. $Q = \{ q_1, \ldots, q_n \}$ is the finite set of machine states,
2. $\Sigma$ is the finite input alphabet,
3. $q_1 \in Q$ is the initial state,
4. $F \subseteq Q$ is the set of accepting states, and
5. $\Sigma_{[]} = \Sigma \cup \{ \rceil \}$, where $\rceil \notin \Sigma$ is the left end-marker symbol, placed automatically before the input string, and for each $\sigma \in \Sigma_{[]}$, $E_\sigma$ is the superoperator describing the transformation on the current configuration of the machine associated with the consumption of the symbol $\sigma$. For some $l \geq 1$, each $E_\sigma$ consists of $l$ operation elements $\{ E_{\sigma, 1}, \ldots, E_{\sigma, l} \}$, where each operation element is a complex-valued $n \times n$ matrix.

Although it is customary in the literature to analyze these machines using density matrices [5], we take the alternative (but equivalent) approach of [14], which makes the thermodynamic cost of computational steps explicit by representing the “periphery” that will support intermediate measurements during the execution of our QFA. For this purpose, consider an auxiliary system with the state set $\Omega = \{ \omega_1, \ldots, \omega_l \}$, and an additional set of classical states $\{ s_1, \ldots, s_l \}$ that will mirror the members of $\Omega$ during computation, as will be described below.

Considered together, the auxiliary system and the “main system” of our machine defined above have the state set $\Omega \times Q$. The quantum state space of the overall system is $H_l \otimes H_n$, the composite of the corresponding finite-dimensional Hilbert spaces. Initially, this composite system is in the quantum state $| \omega_1 \rangle \otimes | q_1 \rangle$, and the classical state is $s_1$. At the beginning of every computational step, it will be ensured that the auxiliary system is again at one of its computational basis states, i.e. $| \omega_j \rangle$ for some $j$, and the classical state will be $s_j$.

Let $| \psi_x \rangle = \alpha_1 | q_1 \rangle + \ldots + \alpha_n | q_n \rangle$ denote any vector in $H_n$ that is attained by our QFA with nonzero probability after it has consumed the string $x \in \Sigma^*$. We will examine the evolution of the overall system for a single step starting at a state $| \omega_j \rangle \otimes | \psi_x \rangle$. If the symbol $\sigma$ is consumed from the input, the composed system first undergoes the unitary operation described by the product $U_\sigma U_{s_j}$, as described below.

$U_{s_j}$ is designed so that its application rotates the auxiliary state from $\omega_j$ to $\omega_1$, so that $U_{s_j}$ will act on

$$| \Psi_x \rangle = | \omega_1 \rangle \otimes | \psi_x \rangle = \left( \begin{array}{c} \alpha_1, \alpha_2, \ldots, \alpha_n, 0, 0, \ldots, 0 \end{array} \right)^T.$$  

Only the leftmost $n$ columns of the matrix $U_{s_j}$ are significant for our purposes, and the remaining ones can be filled in to ensure unitarity. Those first $n$ columns

References [1] and [14] provide a more comprehensive introduction to the quantum computation notation and concepts discussed here.
will be provided by the operation elements \( E_{\sigma,1}, \ldots, E_{\sigma,l} \), as indicated by the following partitioning of \( U_\sigma \) into \( n \times n \) blocks:

\[
U_\sigma = \begin{bmatrix}
E_{\sigma,1} & \cdots & E_{\sigma,l} \\
\vdots & \ddots & \vdots \\
E_{\sigma,1} & \cdots & E_{\sigma,l}
\end{bmatrix}
\]

(Since \( U_\sigma \) is unitary, one sees that the operation elements should satisfy the constraint \( \sum_{j=1}^l E_{\sigma,j}^\dagger E_{\sigma,j} = I \).)

Consider the \( n \)-dimensional vectors \( \widetilde{|\psi_{x\sigma,i}\rangle} = E_{\sigma,i} |\psi_x\rangle \) for \( i \in \{1, \ldots, l\} \).

Clearly, the vector \( \widetilde{|\Psi_{x\sigma}\rangle} = U_\sigma |\Psi_x\rangle \) that represents the overall system state obtained after the unitary transformation described above can be written by “stacking” these vectors, each of which corresponds to a different auxiliary state, on top of each other, as seen in Equation 1.

\[
\widetilde{|\Psi_{x\sigma}\rangle} = \begin{bmatrix}
|\psi_{x\sigma,1}\rangle \\
|\psi_{x\sigma,2}\rangle \\
\vdots \\
|\psi_{x\sigma,l}\rangle
\end{bmatrix} = |\omega_1\rangle \otimes \widetilde{|\psi_{x\sigma,1}\rangle} + |\omega_2\rangle \otimes \widetilde{|\psi_{x\sigma,2}\rangle} + \cdots + |\omega_l\rangle \otimes \widetilde{|\psi_{x\sigma,l}\rangle} \tag{1}
\]

At this point in the execution of our QFA, the auxiliary system is measured in its computational basis. The probability \( p_k \) of observing outcome “\( \omega_k \)” out of the \( l \) different possibilities is the square of the length of \( \widetilde{|\psi_{x\sigma,k}\rangle} \). As a result of this probabilistic branching, the main system collapses to the state \( |\psi_{x\sigma,k}\rangle = \frac{\widetilde{|\psi_{x\sigma,k}\rangle}}{\sqrt{p_k}} \) with probability \( p_k \) (for \( k \) such that \( p_k > 0 \)), and the fact that this observation result is recorded for usage in the next step is represented by setting the classical state to \( s_k \), overwriting its present value. It is this final action of “forgetting” the previous value of the classical state that causes the energy cost associated per step of a QFA: \( \log_2 l \) classical bits are required to hold this information, and one needs to expend a minimum of \( k_B T \ln 2 \) joules to erase each bit, where \( k_B \) is Boltzmann’s constant, and \( T \) is the ambient temperature in kelvins [9]. A machine with \( l > 1 \) operating elements in its superoperators is therefore faced with an energy cost proportional to \( \log_2 l \).

After processing the entire input string symbol by symbol in this manner, the main system, described by some \( n \)-dimensional vector \( |\psi\rangle \), is measured in its computational basis. The probability of acceptance at this point is the sum of the squares of the lengths of the amplitudes of the accepting states in \( |\psi\rangle \). Rejection is similarly defined in terms of the non-accepting states. A language \( L \) is said to be recognized by a QFA with \textit{bounded error} if there exists a number \( \epsilon < \frac{1}{2} \) such that every string in \( L \) is accepted and every string not in \( L \) is rejected by that QFA with probability at least \( 1 - \epsilon \). If \( \epsilon = 0 \), i.e. the QFA has the property that
it accepts every input string with either probability 0 or 1, it is said to recognize the set of strings that it does accept with zero error.

It is known [11] that “modern” QFAs defined in this manner can recognize all and only the regular languages with bounded error. Given any deterministic finite automaton (DFA) with \( n \) states, it is straightforward to build a QFA with \( n \) machine states that recognizes the same language \( M \) with zero error. An examination of this construction is useful for understanding the nature of the information lost when the classical state is overwritten during a computational step of a QFA.

Consider the DFA whose state diagram is shown in Fig. 1a. Figures 1b and 1c depict the operation elements associated with symbols \( a \) and \( b \) in the QFA implementation of this machine. In each square matrix, both the rows and columns correspond to the states of the QFA, which in turn correspond to the states of the DFA of Fig. 1a. The entry at row \( i \), column \( j \) of the \( k \)’th operation element for symbol \( a \) represents the transition that the QFA would perform.

\( E_{a,1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \)

\( E_{b,1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \)

\( E_{a,2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \)

\( E_{b,2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \)

\( E_{a,3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \)

\( E_{b,3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \)

Fig. 1: A DFA and superoperators for its QFA implementation

\(^2\) “Zero-energy” QFAs with a single operation element in each superoperator correspond to the earliest definition in [12], and can recognize all and only the group languages (a proper subclass of the class of regular languages, whose DFAs have the property that one again obtains a DFA if one reverses all their transitions) with bounded error [11].

\(^3\) The left end-marker is inconsequential in DFA simulations, and its superoperator is not shown.
from its $j$'th machine state to the combination of its $i$'th machine state and $k$'th auxiliary state. Starting with the vector $(1,0,0,0)^T$ representing the machine being at the initial state with probability 1, the QFA would trace every step in the execution of the DFA on any input string, and recognize the same language with zero error.

The reader will note in Fig. 1 that the superoperators, which are just adjacency matrices for the DFA, have not one, but three operation elements precisely because state 3 has three incoming transitions labeled with the same symbol in Fig. 1a. We cannot have two 1's in the same row of two different columns of the matrices in Figures 1b and 1c since they must be orthonormal. We use the additional operation elements to represent the additional ways in which the machine can switch to state 3 with input $b$. Intuitively, the auxiliary state records which of the three transitions was used to enter state 3, and it is not possible to “trace the computation backwards” from that state when one has forgotten that information. This is why the language recognized by these machines is not “reversible” [8].

We have seen how any DFA with $n$ states and at most $l$ incoming transitions to the same state with the same symbol can be simulated by a zero-error QFA with $n$ machine states and $l$ operation elements (that only contain 0’s and 1’s) per superoperator. Note that the QFAs that are constructed to imitate DFAs in the fashion exemplified above do not use any “quantumness”: At all times, the state vector of the QFA never represents a superposition of more than one classical state, and just tracks the execution of the DFA faithfully. There is no probabilistic “branching” (since only one auxiliary state has nonzero amplitude at any step), and no constructive or destructive interference among amplitudes. It is natural to ask if QFAs with other complex-valued entries in their operation elements can utilize the famous non-classical capabilities of quantum systems to perform the same task in a more energy-efficient manner, i.e. with fewer operation elements. We turn to this question in the next section.

3 Zero-error QFAs have no energy advantage

For any language $L$ defined over alphabet $\Sigma$, the “indistinguishability” relation $\equiv_L$ on the set $\Sigma^*$ is defined as follows:

$$ (x \equiv_L y) \iff (\forall z \in \Sigma^* [xz \in L \iff yz \in L]). $$

Lemma 1. Let $M$ be a QFA recognizing a language $L$ with zero error. Let $x$ and $y$ be strings such that $x \not\equiv_L y$. If $|\psi_x\rangle, |\psi_y\rangle \in \mathcal{H}_n$ are any two vectors that are attained by $M$ with nonzero probability after it reads $x$ or $y$, respectively, then $\langle \psi_x | \psi_y \rangle = 0$.

---

4 Since none of the three states with $b$-transitions into state 3 is more likely to be the source than the others, this information amounts to $\log_2 3$ bits.
Proof. Let us say that \( x \) and \( y \) are distinguishable with respect to \( L \) in \( k \) steps if there exists a string \( z \) of length \( k \) that distinguishes them, i.e. \( xz \in L \) if and only if \( yz \notin L \). We will prove the statement by induction on the number of steps in which \( x \) and \( y \) are distinguishable with respect to \( L \).

**Basis:**
If \( x \) and \( y \) are distinguishable with respect to \( L \) in 0 steps, let us say without loss of generality that \( x \in L \) and \( y \notin L \). In this case, all entries of \( |\psi_x⟩ \) corresponding to the non-accepting states in \( Q \) must be zero, since \( M \) would otherwise reject \( x \) with nonzero probability. Similarly, all entries of \( |\psi_y⟩ \) corresponding to the accepting states in \( Q \) must be zero. But this means that

\[
\langle \psi_x | \psi_y \rangle = 0.
\]

**Induction step:**
Assume that the statement is true for all pairs of strings that are distinguishable with respect to \( L \) in \( k \) steps, where \( k \geq 0 \), and consider the case of any \( x \) and \( y \) that are distinguishable in \( k + 1 \) steps. In this context, we will further assume that \( \langle \psi_x | \psi_y \rangle \neq 0 \), and reach a contradiction.

Let \( σ \) be the leftmost symbol of the string \( z \) (of length \( k + 1 \)) that distinguishes \( x \) and \( y \). Consider two copies of \( M \) at states \( |\psi_x⟩ \) and \( |\psi_y⟩ \). When these two machines consume the input symbol \( σ \), the corresponding vectors representing the composite system of the machine and its environment are both multiplied by the unitary matrix \( U_σ \) to yield two \( nl \)-dimensional vectors, say,

\[
|\tilde{\Psi}_{xσ}\rangle = \begin{bmatrix}
|ψ_{xσ,1}\rangle \\
|ψ_{xσ,2}\rangle \\
\vdots \\
|ψ_{xσ,l}\rangle
\end{bmatrix}
\]

and

\[
|\tilde{\Psi}_{yσ}\rangle = \begin{bmatrix}
|ψ_{yσ,1}\rangle \\
|ψ_{yσ,2}\rangle \\
\vdots \\
|ψ_{yσ,l}\rangle
\end{bmatrix},
\]

where \( n \) and \( l \) are respectively the numbers of machine and auxiliary states, as we saw in Equation 1. Since \( U_σ \) preserves inner products and angles, these “tall” vectors are also not orthogonal by our assumption that \( \langle \psi_x | \psi_y \rangle \neq 0 \).

As discussed in Section 2, the state vectors that \( M \) can attain with nonzero probability after consuming this \( σ \) are the normalized versions of the (nonzero) \( n \)-dimensional “slices” of \( |\tilde{\Psi}_{xσ}\rangle \) and \( |\tilde{\Psi}_{yσ}\rangle \). Note in Equation 2 that if \( \langle ψ_{xσ,j} | ψ_{yσ,j} \rangle = 0 \) for all \( j \in \{1, \ldots, l\} \), then \( |\tilde{\Psi}_{xσ}\rangle \) and \( |\tilde{\Psi}_{yσ}\rangle \) must also be orthogonal. This means that \( \langle ψ_{xσ,j} | ψ_{yσ,j} \rangle \neq 0 \) for at least one \( j \), which is a contradiction, since \( xσ \) and \( yσ \) are distinguishable in \( k \) steps.

It follows that the subspace generated by the vectors attainable by \( M \) through reading strings in a particular equivalence class of \( \equiv_L \) must be orthogonal to all the subspaces corresponding to the other classes. This provides a new proof of the (already known) fact that zero-error QFAs can only recognize regular languages.

**Corollary 1.** If a language \( L \) is recognized by a zero-error QFA \( M \) with \( n \) machine states, each equivalence class \( C \) of \( \equiv_L \) corresponds to a subspace \( S_C \) of \( \mathcal{H}_n \), and any two subspaces corresponding to different classes are orthogonal to
each other. Since the sum of the dimensions of these subspaces is at most \( n \equiv_{L} \), can have at most \( n \) equivalence classes, and \( L \) is regular by the Myhill-Nerode theorem.

We can now demonstrate that every zero-error QFA \( M \) has a corresponding DFA \( M' \) which recognizes the same language, and is as efficient as \( M \) in terms of both memory (number of states) and energy requirement per computational step.\(^5\)

Theorem 1. For any \( n, l > 0 \), if a language is recognized with zero error by a QFA with \( n \) machine states and \( l \) operation elements per superoperator, then the same language is also recognized by a DFA with \( n \) states and at most \( l \) incoming transitions to the same state with the same symbol.

Proof. Let \( M \) be a zero-error QFA with \( n \) machine states and \( l \) operation elements per superoperator. By Corollary \(^4\) \( M \) recognizes a regular language \( L \). Let \( k \) be the number of states of the minimal DFA \( D \) recognizing \( L \). Each input string \( x \) that carries \( D \) to state \( i \in \{1, \ldots, k\} \) of \( D \) will carry \( M \) to a state vector in a corresponding subspace \( S_i \) of \( H_n \). Consider the DFA \( M' \) that is described by the 5-tuple \( (Q, \Sigma, \delta, q_1, F) \), where

1. \( Q \) is the finite set of states, containing \( \sum_i \dim(S_i) \) elements, with \( \dim(S_i) \) equivalent states corresponding to \( S_i \) for each \( i \in \{1, \ldots, k\} \),
2. \( \Sigma \) is the same as the input alphabet of \( M \),
3. \( q_1 \) is the initial state, selected arbitrarily from among the elements of \( Q \) that correspond to the subspace containing the vector attained by \( M \) after consuming the empty input string,
4. \( F \) is the set of accepting states, designated to contain all and only the elements of \( Q \) that correspond to any subspace containing vectors attained by \( M \) after consuming members of \( L \), and
5. \( \delta \) is the transition function, which mimics \( M \)'s action on its state vector, as follows: For each \( i \in \{1, \ldots, k\} \), call the subset of \( \dim(S_i) \) states corresponding to \( S_i \) “the \( i \)'th bag”. If \( M \) maps vectors in \( S_i \) to vectors in \( S_j \) upon reading a symbol \( \sigma \), all states in the \( i \)'th bag of \( M' \) will transition to states in the \( j \)'th bag with the symbol \( \sigma \). For each bag, incoming transitions will be distributed as “evenly” as possible among the members of that bag, so that if \( M' \) has a total of \( T_j \) incoming \( \sigma \)-transitions to its \( j \)'th bag, no state in that bag will have more than \( \lceil T_j / \dim(S_j) \rceil \) incoming \( \sigma \)-transitions.

\(^5\) The fact that zero-error QFAs have no advantage over equivalent DFAs in terms of the number of machine states was first proven by Klauck \(^6\), using Holevo’s theorem and communication complexity arguments.

\(^6\) At this point, one may be tempted to declare that the set of subspaces already provides the state set of the DFA we wish to construct. After all, each matrix of the form \( U_{\sigma} \) that we saw in Section \(^2\) “maps” a vector in \( S_i \) to one or more vectors in \( S_j \) if and only if \( D \) switches from state \( i \) to state \( j \) upon consuming \( \sigma \). However, this simple construction does not guarantee our aim of keeping the maximum number of incoming transitions with the same label to any state in the machine to a minimum.
Let us calculate the maximum possible number of incoming \( \sigma \)-transitions that can be received by a state in \( M' \). Let \( j \) be some state of \( D \) with \( p \) incoming \( \sigma \)-transitions from states \( \{i_1, i_2, ..., i_p\} \). For any \( r \in \{1, 2, ..., p\} \), let \( x \) be some string which carries \( D \) to state \( i_r \) and \( M \) to some vector \( |\psi_x\rangle \in \mathcal{H}_n \) with nonzero probability. We know that the processing of \( \sigma \) corresponds to the action of the matrix we called \( U_\sigma \) in Section 2. Recall from Equation 1 that
\[
U_\sigma(|\omega_1\rangle \otimes |\psi_x\rangle) = |\omega_1\rangle \otimes |\bar{\psi}_{x\sigma,1}\rangle + |\omega_2\rangle \otimes |\bar{\psi}_{x\sigma,2}\rangle + \ldots + |\omega_l\rangle \otimes |\bar{\psi}_{x\sigma,l}\rangle.
\]
Since \( M \) transitions to a vector in \( S_j \) with probability 1 upon receiving \( \sigma \), all the \( |\bar{\psi}_{x\sigma,k}\rangle \) must lie in \( S_j \) (for \( 1 \leq k \leq l \)). We therefore have \( U_\sigma(|\omega_1\rangle \otimes |\psi_x\rangle) \subseteq \mathcal{H}_l \otimes S_j \).
This is true for all \( x \) and \( |\psi_x\rangle \), and \( S_i \) is, by definition, generated by such vectors; therefore, \( U_\sigma(C|\omega_1\rangle \otimes S_{i_1}) \subseteq \mathcal{H}_l \otimes S_j \).

By Corollary 1 the spaces \( C|\omega_1\rangle \otimes S_{i_r} \) are disjoint for all \( r \in \{1, 2, ..., p\} \).
We have
\[
\dim(C|\omega_1\rangle \otimes S_{i_1}) + \dim(C|\omega_1\rangle \otimes S_{i_2}) + \ldots + \dim(C|\omega_1\rangle \otimes S_{i_p}) \leq \dim(\mathcal{H}_l \otimes S_j),
\]
since \( U_\sigma \) is an injective linear map. In other words,
\[
T_j = \dim(S_{i_1}) + \dim(S_{i_2}) + \ldots + \dim(S_{i_p}) \leq l \dim(S_j).
\]
Therefore,
\[
\frac{T_j}{\dim(S_j)} \leq l, \text{ and no state receives more than } l \text{ incoming } \sigma\text{-transitions.}
\]

Having seen that the erasure costs associated with zero-error QFAs are precisely representable by DFAs, we will use this link to establish an upper bound for the energy requirement of regular language recognition in the next section.

### 4 An upper bound for information erasure per step

It is natural to ask if there exists a universal bound on the number of bits that have to be “forgotten” per computational step of any finite automaton. In this section, we provide an answer to this question.

**Theorem 2.** Every language on an alphabet \( \Sigma \) can be recognized by a DFA that has at most \( |\Sigma|+1 \) incoming transitions labeled with the same symbol to any of its states.

**Proof.** For unary input alphabets, any minimal machine is already in the required form. For \( k > 1 \), let \( M \) be the minimal DFA recognizing some language \( L \) on the alphabet \( \Sigma = \{\sigma_1, \ldots, \sigma_k\} \). If \( M \) is in the required form, we are done. Otherwise, let \( Q = \{q_1, q_2, ..., q_n\} \) be the state set of \( M \). We will add some new states (each of which will be equivalent to some \( q_i \in Q \)) to \( M \) to obtain a larger machine recognizing \( L \). We will use the sets \( C_1, C_2, ..., C_n \) so that each \( C_i \) will contain the states that are equivalent to \( q_i \) in our machine at every step of
our construction. Let \( \mu_i \) denote the number of states in \( C_i \) at any point in the process. Originally, each \( C_i = \{q_i\} \), and each \( \mu_i = 1 \). Let \( d_{\sigma,i} \) denote the total number of incoming transitions to states in \( C_i \) with the symbol \( \sigma \). Note that \( d_{\sigma,i} \) is equal to the sum of \( \mu_j \) over the \( C_j \) containing states that transition to states in \( C_i \) with the symbol \( \sigma \), and it is independent of the details of which particular state in \( C_i \) receives a particular \( \sigma \)-transition. Define

\[
u_i = \mu_i(k + 1) - \max_{\sigma \in \Sigma} d_{\sigma,i}.
\]

Note that if any state in any \( C_i \) receives more than \( k + 1 \) transitions with the same symbol, then the DFA does not satisfy the requirement of the theorem. The first term \( \mu_i(k + 1) \) is the maximum allowed number of incoming transitions with the same symbol to \( C_i \) for the machine to satisfy the requirement of the theorem. If each \( \nu_i \) is nonnegative, then it is possible to distribute the incoming transitions among the states in \( C_i \) such that no state receives more than \( k + 1 \) transitions with the same symbol. The rest of the proof outlines an algorithm to find \((\mu_i)_{i \in \{1,2,\ldots,n\}}\) making all \( \nu_i \) nonnegative.

**Algorithm**

- Repeat until all \((\nu_i)_{1 \leq i \leq n}\) are nonnegative:
  - Find an index \( i \) with the smallest \( \nu_i \) value
  - Add a state to \( C_i \), increasing \( \mu_i \) by 1
- End

At each iteration of the loop, \( \sum_{i \in \{1,\ldots,n\}} \mu_i(k + 1) \) increases by \( k + 1 \). On the other hand, \( \sum_{i \in \{1,\ldots,n\}} \max_{\sigma \in \Sigma} d_{\sigma,i} \) also increases, but it increases by at most \( k \), since the added state has only \( k \) outgoing transition arrows. Therefore, by Equation 3, the sum of all the \( \nu_i \) increases at least by one. Note that \( \max_{i \in \{1,2,\ldots,n\}} \nu_i \) is bounded above by a constant, because only those \( \nu_i \) that are less than 0 are increased (by at most \( k + 1 \)) during the execution of the algorithm. The algorithm terminates at some point because the sum of finitely many bounded variables cannot increase by one forever.

The required DFA is then constructed by distributing each \( C_i \)'s incoming transitions evenly between its states, which works because each \( \nu_i \) is now nonnegative.

We now show that the bound shown in Theorem 2 is tight.

**Theorem 3.** For every \( j \geq 1 \), there exists a language \( L_j \) on a \( j \)-symbol alphabet with the following property: All DFAs recognizing \( L_j \) have a state \( q \) such that at least \( j + 1 \) states transition to \( q \) upon receiving the same symbol.

**Proof.** For the unary alphabet, it is easy to see that the language \( L_1 \) containing all strings except the empty string must have the property. For \( j > 1 \), define the “successor” function \( F \) on \( \{1,\ldots,j\} \) by \( F(i) = (i \mod j) + 1 \), and let \( B \) be \( F \)'s inverse. On the alphabet \( \Sigma_j = \{\sigma_1,\ldots,\sigma_j\} \), define

\[
L_j = \{w \mid w \text{ ends with } \sigma_i \sigma_{F(i)} \text{ for some } 1 \leq i \leq j\}.
\]
Let $M$ be a DFA recognizing $L_j$. Assume, without loss of generality, that $M$ does not have unreachable states.

Similarly to the proofs of Theorems 1 and 2, we will be talking about "bags" into which the states of $M$ are partitioned. Each bag contains states that are equivalent to the ones in the same bag, and distinguishable from all states in the other bags. $S$ is the bag that contains the initial state. For each $k$, $A_k$ is the bag containing the state reached by the input $\sigma_{B(k)} \sigma_k$, and $R_k$ is the bag containing any state reached by inputs of the form $\tau \sigma_k$, where $\tau$ is any substring not ending with $\sigma_{B(k)}$. Note that $A_i$ and $R_k$ are distinct bags for any $i, k \leq j$, because all states in $A_i$ are accepting states and those in $R_k$ are not. For $X \in \{A, R\}$, $A_k$ and $X_i$ are also distinct when $k \neq i$, since $M$ would reach an accepting state if it consumes the symbol $\sigma_{F(k)}$ when in a member of $A_i$, whereas it would reach a rejecting state with that symbol from a state in $X_i$. $S$ is distinct from all the $A_i$ and $R_i$, because it contains the only state which is two steps away from any accept state. The bags $(A_k)_k$, $(R_k)_k$ and $S$ partition the entire state set.

The definition of $L_j$ dictates that all incoming transitions to states in $A_k$ or $R_k$ are labeled with the symbol $\sigma_k$. Let $i$ ($1 \leq i \leq j$) be the index minimizing $|A_i| + |R_i|$, i.e. the sum of states in $A_i$ and $R_i$. Note that all states in all bags $(A_k)_k$, $(R_k)_k$ and $S$ transition to either $A_i$ or $R_i$ upon reading the symbol $\sigma_i$, so there are

$$\left( \sum_{1 \leq k \leq j} |A_k| \right) + \left( \sum_{1 \leq k \leq j} |R_k| \right) + |S|$$

transitions with the symbol $\sigma_i$. Since $|A_i| + |R_i|$ is minimal and $|S| > 0$, this number is strictly larger than $j(|A_i| + |R_i|)$. At least one state in $A_i$ or $R_i$ should thus have at least $j + 1$ incoming $\sigma_i$-transitions by the pigeonhole principle. $\square$

Theorems 2 and 3 imply that, for any particular temperature $T$, given any amount of energy, there exists a regular language (on a suitably large alphabet) whose recognition at $T$ requires a DFA with at least that much energy cost per computational step. When the alphabet is fixed, one can always rewrite any DFA on that alphabet to obtain an "energy-efficient" machine recognizing the same language with each step costing no more than the bound proven in Theorem 2. By Theorem 1, the same energy costs are associated with zero-error QFAs for that language.

Since smaller alphabets are associated with less energy cost per step, one may ask whether encoding a language on a bigger alphabet by replacing each symbol by a binary substring would decrease the overall energy consumption. For any $j > 1$, any machine recognizing language $L_{2j}$ as defined in the proof of Theorem 3 needs to forget $n \log_2 (2^j + 1)$ bits to process an input of length $n$. For a machine recognizing a version of $L_j$ encoded in binary, the cost per step would be less, but the encoded input string would be longer, with the total number of erased bits amounting to the greater value $nj \log_2 3$. 
5 Trading energy for error

It turns out that the minimum energy required for the recognition of some regular languages is reduced if one allows the finite automaton to give erroneous answers with probability not exceeding some bound less than $\frac{1}{2}$.

Recall the language family $\{L_j | j \geq 1\}$ defined in the proof of Theorem 3. Any zero-error QFA recognizing some $L_j$ must have at least $j + 1$ operating elements by Theorem 1. Since $L_1$ is not a group language, no QFA with a single operating element can recognize it, even with bounded error [4].

**Theorem 4.** There exists a QFA with two operating elements per superoperator that recognizes the language $L_2$ with bounded error.

**Proof.** Consider Fig. 2 which depicts the transitions of a QFA named $M_2$. (All arrows in the figure correspond to transitions with amplitude 1, unless otherwise indicated.) The superoperators for the left end-marker and the two input symbols are shown in Fig. 3. Upon reading the left end-marker, $M_2$ branches to three equal-amplitude submachines that never interfere in the remainder of the computation. For $i \in \{1, 2\}$, submachine $M_{2,i}$ accepts an input string if and only if it ends with $\sigma_i \sigma_{F(i)}$. Submachine $M_{2,3}$ accepts every input. Since any string is in $L_2$ if and only if it is accepted by one of $M_{2,1}$ and $M_{2,2}$, $M_2$ recognizes $L_2$ with error probability $\frac{1}{3}$.

**Theorem 5.** For all $j \geq 3$, there exists a QFA $M_j$ with three operating elements per superoperator that recognizes the language $L_j$ with error probability bounded by $\frac{j-1}{2j-1}$.
Fig. 3: Transition matrices for the QFA of Figure 2

(a) Superoperator for $\triangleright$  (b) Superoperator for $\sigma_1$  (c) Superoperator for $\sigma_2$

**Proof.** The argument is similar to the one in the proof of Theorem 4. Machine $M_j$ has $j + 1$ submachines. For $i \in \{1, \ldots, j\}$, submachine $M_{j,i}$ (depicted in Fig. 4) accepts its input if and only if it ends with $\sigma_i \sigma_F(i)$, whereas submachine $M_{j,j+1}$ accepts every input. (In Fig. 4 arrow labels of the form $\Sigma - \Gamma$ express that all symbols of the input alphabet except those in set $\Gamma$ effect a transition with amplitude 1 between the indicated states.) $M_j$ starts by branching with amplitude $\frac{1}{\sqrt{2^j - 1}}$ to each of $M_{j,i}$ for $i \in \{1, \ldots, j\}$, and with amplitude $\frac{1}{\sqrt{2^j - 1}}$ to $M_{j,j+1}$. Strings in $L_j$ must lead one of the first $j$ submachines to acceptance, and “tip the balance” for the overall machine to accept with probability at least $\frac{j}{2^j - 1}$. It is easy to see in Fig. 4 that the superoperators would have just three operating elements. \hfill \square

Fig. 4: Submachine $M_{j,i}$ in the construction of Theorem 5
6 Concluding remarks

The approach we present for the study of energy complexity can be extended to several other scenarios, like interactive proof systems, involving finite-memory machines. We end with a list of some open questions.

– Is there a regular language whose zero-error recognition requires a QFA with more than two operation elements, whereas no energy savings are possible when one allows some bounded error in the recognition process?
– Is there a regular language whose bounded-error recognition requires a QFA with more than three operation elements?
– Can the construction in the proof of Theorem 5 be improved to reduce the error bounds, the energy requirement, or both?

References

1. Ambainis, A., Yakaryılmaz, A.: Automata and quantum computing (2018), arXiv:1507.01988v2
2. Bennett, C.H.: Logical reversibility of computation. IBM Journal of Research and Development 17(6), 525–532 (1973)
3. Bertoni, A., Carpentieri, M.: Regular languages accepted by quantum automata. Information and Computation 165(2), 174–182 (2001)
4. Brodsky, A., Pippenger, N.: Characterizations of 1-way quantum finite automata. SIAM Journal on Computing 31, 1456–1478 (2002)
5. Hirvensalo, M.: Quantum automata with open time evolution. International Journal of Natural Computing 1, 70–85 (2010)
6. Klauck, H.: On quantum and probabilistic communication: Las Vegas and one-way protocols. In: 32th ACM Symp. on Theory of Comp. pp. 644–651 (2000)
7. Kondacs, A., Watrous, J.: On the power of quantum finite state automata. In: Proceedings 38th Symp. Found. Computer Science. pp. 66–75 (1997)
8. Kutrib, M.: Aspects of reversibility for classical automata. In: Calude, C.S., Freivalds, R., Kazuo, I. (eds.) Computing with New Resources: Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday, pp. 83–98. Springer International Publishing (2014)
9. Landauer, R.: Irreversibility and heat generation in the computing process. IBM Journal of Research and Development 5(3), 183–191 (1961)
10. Leff, H.S., Rex, A.F. (eds.): Maxwell’s Demon 2: Entropy, Classical and Quantum Information, Computing. CRC Press (2002)
11. Li, L., Qiu, D., Zou, X., Li, L., Wu, L., Mateus, P.: Characterizations of one-way general quantum finite automata. Theoretical Computer Science 419, 73–91 (2012)
12. Moore, C., Crutchfield, J.P.: Quantum automata and quantum grammars. Theoretical Computer Science 237(1), 275–306 (2000)
13. Pin, J.E.: On reversible automata. In: Simon, I. (ed.) Proceedings of the first LATIN conference. pp. 401–416. Lecture Notes in Computer Science 583, Springer, São-Paulo, Brazil (1992)
14. Say, A.C.C., Yakaryılmaz, A.: Quantum finite automata: A modern introduction. In: Calude, C.S., Freivalds, R., Kazuo, I. (eds.) Computing with New Resources: Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday, pp. 208–222. Springer International Publishing (2014)