Technical Report No. 2007-537
Parallelism in quantum information processing defeats the Universal Computer

Marius Nagy and Selim G. Akl
School of Computing
Queen's University
Kingston, Ontario K7L 3N6
Canada
Email: {marius,akl}@cs.queensu.ca

Abstract

This paper is structured around the idea that a *finite* Universal Computer cannot be realized and presents in detail a series of unconventional computing paradigms supporting this idea from a quantum mechanical perspective.

1 Introduction

Some of the computations carried out today are qualitatively different from those performed more than half a century ago, when the age of computers was only just beginning. The traditional concept of computation is best captured by the functioning of the Turing machine. A sequence of operations (or transformations) forming the *algorithm* is applied to a set of *input* data to produce an *output* (or result). There are no space and time limitations, nor any restrictions imposed on the input or output data. The whole input is available at the outset (on the tape, in the case of the Turing machine) and the result is reported (placed on the tape) when the computation terminates (assuming it does). The majority of the computations performed today on various electronic computers and computing devices fit the above description. We refer to them as *conventional* computing paradigms and, given enough memory and time, they can be solved by any of today's computers. The Turing machine stands as a mathematical model, an abstract prototype for any of these computing devices.

However, in time, this rather simplistic view on computation has been challenged by increasingly demanding applications and real-world problems. For example, we need better solutions, faster, to problems whose input specifications may vary with time. Often, our results need to be obtained before certain deadlines, or else various penalties can be applied. The information processing tasks we face today or those we discover to take place in Nature often possess attributes which make them unsuitable for a Turing machine. These attributes usually describe the dynamic nature of a computation, from the way the input is presented, continuing with the characteristics of the algorithm operating on the input data and ending with the possible constraints that can be imposed on the output. We call such a computation *unconventional*, as opposed to the general pattern exhibited by Turing computations. In this paper we address five unconventional computing paradigms (enumerated below), sharing the generic property that the input variables are temporally and/or spatially interconnected. Each of these paradigms is exemplified through a concrete example provided by quantum mechanics.

With the advent of unconventional computing paradigms, the concept of *universality* too, needs to be revised, or at least clarified. To this end, *parallelism* may offer the means to show that a Universal Computer with fixed and finite physical characteristics (speed, number of processing units, etc.) cannot be built [1]. An infinite hierarchy of computing devices exists, with each machine capable of simulating any one below it in the hierarchy, but none above, because it lacks the required number of processing elements necessary for coping with the degree of parallelism inherent in certain applications.

In the general framework of *evolving* computations, whose characteristics vary during their execution, there are many paradigms for which a parallel computing approach is most appropriate, if not vital for the success of the computation [3]. Here are some examples:

- the computational complexity of a step in a certain computation may depend on the *time* when the step is executed or on the order of execution (*rank*) of that step within an algorithm that solves the problem at hand.
- the variables upon which the algorithm is supposed to act are affected by the passage of time.
- the input data are interconnected in such a way that operating on any one value inevitably disturbs the others.
- at each step of the computation, a global, mathematical constraint has to be obeyed.

In each of the above cases, a problem instance of size n can only be solved by a machine equipped with at least n processing units and the solution cannot possibly be simulated by another machine with fewer processors. This observation is at the heart of the impossibility of achieving universality in computing. In what follows, we show that quantum information processing provides excellent examples of evolving computing paradigms, and the need for parallelism in this newly emerged unconventional field transforms the Universal Computer into a myth.

The remainder of the paper is structured as follows. In the next section the reader is familiarized with the concept of evolving computations, and five examples of such computing paradigms are described therein. For each of these paradigms, a quantum mechanical instance is presented in the ensuing three sections. Section 3 shows how the procedures for computing the quantum Fourier transform and its inverse can be decomposed into steps of rank-varying computational complexity. In a practical setting, quantum decoherence places a hard deadline on when these computations have to be completed, offering a typical example of time-varying variables. Fortunately, the use of a parallel architecture can reduce the execution time and help complete the computation before the sensitive quantum information leaks into the surrounding environment.

Section 4 is concerned with quantum error-correction schemes from the viewpoint of time-varying computational complexity. In section 5, we focus on *entanglement* among qubits, first, as a quantum instance of the interacting variables paradigm. Since, technically, entanglement is a mathematical constraint imposed on the quantum state of the whole ensemble, we then assert that a quantum computation that has to maintain entanglement at all times belongs to the paradigm of computations obeying a global condition. Furthermore, such a computation can only be carried out by manipulating the whole ensemble as a single entity (in other words, acting in parallel on all components). Finally, section 6 draws some important conclusions based on the evidence presented in the paper.

2 Evolving computations

Evolution (or merely *change*) is a fundamental attribute of many systems that we observe and investigate, whether they are physical, biological, economic, social or of any other nature. Yet, until recently, computational systems whose characteristics change during the computational process itself did not receive much attention. In this section, we describe five computing paradigms, labeled *unconventional* precisely because of their dynamic nature.

At an abstract level, the following generic problem needs to be solved: a set of n input variables $x_0, x_1, \ldots, x_{n-1}$ have to be read and a certain function $\mathcal{F}(x_0, x_1, \ldots, x_{n-1})$ must be computed and the result reported. In the first two of the five cases to be described, the focus is on the algorithm employed to compute the function $\mathcal{F}$. What evolves during the computation is the complexity of each step in the algorithm.

2.1 Unconventional Computational Complexity

When analyzing the computational complexity of a given algorithm, we usually focus on how this quantity varies as a function of the problem size, without paying too much attention to how the complexity of each step in the algorithm varies throughout the computation. Though in many cases the complexity of each step is a constant, there are computations for which the cost of executing essentially similar steps is different from one step to another.

2.1.1 Rank-varying computational complexity

One factor that can dictate the complexity of a step is its *rank*, defined as the order of execution of that step. For instance, if the cost of executing the i^{th} step of an algorithm is $c(i) = 2^i$ elementary operations or time units, then the computational complexity of a step grows exponentially with its rank, for that respective algorithm. In other cases, it may be that the computational complexity of a step actually decreases with the rank. An algorithm made up of n steps for which $c(i) = n - i + 1$ for $i = 1, \ldots, n$ illustrates such a situation.

Examples of this kind are hardly new. Euclid's algorithm for computing the greatest common divisor of two numbers executes the same basic operation (a division) at each step, but the size of the operands (and implicitly the complexity of the operation) decreases continually. Algorithms for which an amortized analysis can be applied also make good examples of rank-varying computational complexity. Incrementing a binary counter [11] is a procedure in which the number of bit flips at each step is not constant, though it is neither strictly increasing nor strictly decreasing with the rank.

2.1.2 Time-varying computational complexity

Alternatively, the relentless passage of *time* can directly influence the computational complexity of a given step in the algorithm. The difference between a rank-driven and a time-driven computational complexity can probably be synthesized best in the following manner. If the cost of executing step S_j depends only on the state of the system after executing the previous $j-1$ steps, regardless of how much time was consumed to reach that state, then we clearly have an example of rank-varying computational complexity. But if the complexity of S_j is a function of the particular moment in time when that step is executed, then what we have is a procedure with steps of time-varying computational complexity. For example, if the computational complexity of S_j is described by the function $c(t) = 2^{2^t}$, then the computational resources required to complete that step are rapidly growing with the moment in time when S_j is actually executed.

2.2 Unconventional Computational Variables

For the three remaining examples of unconventional computing paradigms, the focus moves from the algorithm to the input variables $x_0, x_1, \ldots, x_{n-1}$, which now determine the dynamics of the system.

2.2.1 Time-varying variables

In the paradigm dealing with *time-varying variables*, time plays again the main role. Each argument of function $\mathcal{F}$ is itself a function of time: $x_0(t)$, $x_1(t)$, ..., $x_{n-1}(t)$. At each time unit, the values assumed by the input variables change in such a way that the new value cannot be predicted from the former, nor the former recovered from the latter. Certainly, this makes the computation of $\mathcal{F}(x_0(t_0), \ldots, x_{n-1}(t_0))$ at the precise moment $t = t_0$ a challenging task, in case we do not have the capability of reading all n input variables, in parallel, at the right moment.

2.2.2 Interacting variables

But even if the input variables are not affected by the passage of time, the computational environment may still change during the computation. In the next paradigm that we describe, it is the interactions among mutually dependent variables, caused by an interfering agent (performing the computation) that is the origin of the evolution of the system under consideration. Thus, a relationship exists between $x_0, x_1, \ldots, x_{n-1}$ that connects them together. Any attempt to read the value of any one variable will inevitably and unpredictably disturb the values of the remaining variables. More precisely, the act of reading x_i, for any $i \in \{0, 1, \ldots, n-1\}$, causes the system to make a transition from state $(x_0, x_1, \ldots, x_i, \ldots, x_{n-1})$ to $(x'_0, x'_1, \ldots, x'_i, \ldots, x'_{n-1})$. In this way, some of the values needed in the computation of $\mathcal{F}$ may be lost without possibility of recovery. This is the hallmark of the *interacting variables* paradigm.

2.2.3 Computations obeying a global condition

Finally, the relationship among the input variables may take the form of a global property $\mathcal{P}(x_0, x_1, \ldots, x_{n-1})$ that characterizes the initial state of the system and which must be maintained throughout the computation. In particular, if the effect of the computation is to change x_i to x'_i at some point, then $\mathcal{P}(x_0, x_1, \ldots, x'_i, \ldots, x_{n-1})$ must be true for the new state of the system. If the property $\mathcal{P}$ is not satisfied at a given moment of the computation, the latter is considered to have failed.

As the following sections prove it, each of these five unconventional paradigms of computation admits a quantum mechanical instance that requires a parallel approach for a successful outcome.

3 Quantum Fourier Transform

The Fourier transform is a very useful tool in computer science and it proved of crucial importance for quantum computation as well. Since it can be computed much faster on a quantum computer than on a classical one, the discrete Fourier transform allows for the construction of a whole class of fast quantum algorithms. Shor's quantum algorithms for factoring integers and computing discrete logarithms [26] are the most famous examples in this category.

The quantum Fourier transform is a linear operator whose action on any of the computational basis vectors $|0\rangle, |1\rangle, \cdots, |2^n - 1\rangle$ associated with an n-qubit register is described by the following transformation:

$$|j\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i jk/2^n} |k\rangle, \; 0 \leq j \leq 2^n - 1. \tag{1}$$

However, the essential advantage of quantum computation over classical computation is that the quantum mechanical principle of superposition of states allows all possible inputs to be processed at the same time. Consequently, if the quantum register is in an arbitrary superposition of the basis vectors

$$\sum_{j=0}^{2^n-1} x_j |j\rangle,$$

then the quantum Fourier transform will rotate this state into another superposition of the basis vectors

$$\sum_{k=0}^{2^n-1} y_k |k\rangle,$$

in which the output amplitudes y_k represent the discrete Fourier transform of the input amplitudes x_j. Classically, we can compute the numbers y_k from x_j using $\Theta(2^{2n})$ elementary arithmetic operations in a straightforward manner and in $\Theta(n2^n)$ operations by using the Fast Fourier Transform algorithm.

In contrast, a circuit implementing the quantum Fourier transform requires only $\Theta(n^2)$ elementary quantum gates. Such a circuit can be easily derived if equation (1) is rewritten as a tensor product of the n qubits involved:

$$|j_1 j_2 \cdots j_n\rangle \longrightarrow \frac{(|0\rangle + e^{2\pi i 0.j_n}|1\rangle) \otimes (|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle)}{2^{n/2}}. \tag{2}$$

using the binary representation $j_1 j_2 \cdots j_n$ of j and binary fractions in the exponents (for full details see [20]).

Note that each Fourier transformed qubit is in a balanced superposition of $|0\rangle$ and $|1\rangle$. These qubits differ from one another only in the relative phase between the $|0\rangle$ and the $|1\rangle$ components. For the first qubit in the tensor product, j_n will introduce a phase shift of 0 or π, depending on whether its value is 0 or 1, respectively. The phase of the second qubit is determined (controlled) by both j_n and j_{n-1}. It can amount to $\pi + \pi/2$, provided j_{n-1} and j_n are both 1. This dependency on the values of all the previous qubits continues up to (and including) the last term in the tensor product. When $|j_1\rangle$ gets Fourier transformed, the coefficient of $|1\rangle$ in the superposition involves all the digits in the binary expansion of j.

In the case of each qubit, the 0 or π phase induced by its own binary value is implemented through a Hadamard gate. The dependency on the previous qubits is reflected in the use of controlled phase shifts, as depicted in Figure 1. In the figure, H denotes the Hadamard transformation

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

while the gate R_k implements a $\pi/2^{k-1}$ phase shift of the $|1\rangle$ component, according to the unitary transformation

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}.$$

3.1 Rank-varying complexity

Computing the quantum Fourier transform and its inverse can also be seen as examples of algorithms with rank-varying complexity. According to the quantum circuit above, we need n Hadamard gates and $(n-1)+(n-2)+\cdots+1$ conditional rotations, for a total of $n(n+1)/2$ gates required to compute the Fourier transform on n qubits. But this total amount of work

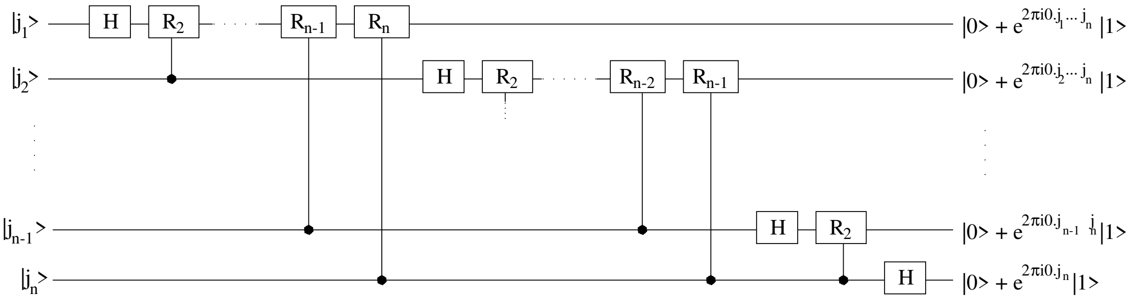

Figure 1: Quantum circuit performing the discrete Fourier transform.

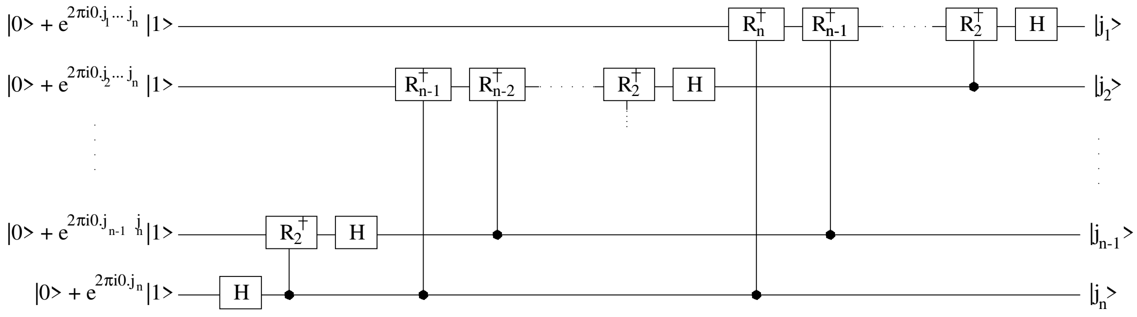

Figure 2: Quantum circuit performing the inverse Fourier transform.

is not evenly distributed over the n qubits. The number of gates a qubit needs to be passed through is in inverse relation with its *rank*. Thus, $|j_1\rangle$ is subjected to n elementary quantum gates, $n-1$ elementary unitary transformations are applied to $|j_2\rangle$, and so on, until $|j_n\rangle$, which needs only one basic operation.

If we break down the quantum Fourier transform algorithm into n steps (one for each qubit involved), then its complexity varies with each step. Starting with $|j_1\rangle$, the time needed to complete each step decreases over time. Since the rank of each step dictates its complexity, the circuit implementing the quantum Fourier transform is an example of a *rank-varying complexity* algorithm.

Naturally, the computation of the inverse quantum Fourier transform can also be decomposed into steps of varying complexity. Reversing each gate in Figure 1 gives us an efficient quantum circuit (depicted in Figure 2) for performing the inverse Fourier transform. Note that the Hadamard gate is its own inverse and $R_k^\dagger$ denotes the conjugate transpose of R_k:

$$R_k^\dagger \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{-2\pi i/2^k} \end{bmatrix}.$$

Getting back to the original $|j_1 j_2 \cdots j_n\rangle$ from its Fourier transformed expression has a certain particularity however. Because of the interdependencies introduced by the controlled rotations, the procedure must start by computing $|j_n\rangle$ and then work its way up to $|j_1\rangle$. The value of $|j_n\rangle$ is needed in the computation of $|j_{n-1}\rangle$. Both $|j_n\rangle$ and $|j_{n-1}\rangle$ are required in order to obtain $|j_{n-2}\rangle$. Finally, the value of all the higher rank bits are used to determine $|j_1\rangle$ precisely. Thus, computing the inverse Fourier transform by the quantum circuit illustrated in Figure 2 is a procedure the complexity of whose steps increases with their rank.

Can a parallel approach be employed in order to counter this variation in complexity and make all steps take a constant amount of time to execute? In the case of the quantum Fourier transform, it is interesting to note that strict sequentiality is enforced by the laws of quantum mechanics, but we still have a chance to speed up the computation, provided we restrict either the input or the output to be classical. No parallel algorithm exists in the general case, when an arbitrary superposition of the basis vectors is Fourier transformed and we are not allowed to measure the output. The reason for this impossibility is the quantum mechanical nature of the qubits controlling the phase shifts in Figures 1 and 2. Such a controlled rotation corresponds to a two-qubit gate and we need to apply, in parallel, a number of two-qubit gates, where the control qubit is the *same* in all gates. Since we cannot gain knowledge of the control qubit's state through measurement and cloning an unknown quantum bit is forbidden by the laws of quantum mechanics, any attempt to parallelize the procedure in the general case is doomed to failure.

However, if the Fourier transform step comes right before measuring in a quantum algorithm, then a parallel solution that can reduce the total running time can be devised. This is not too much of a constraint though, since virtually all quantum algorithms using some form of Fourier transform to interfere the multiple computational paths are following it with a measurement of the quantum register. In particular, this is also true in the case of Shor's quantum algorithms for factoring integers and computing discrete logarithms. Before presenting the details of the parallel architecture designed to compute the quantum Fourier transform, we first describe the most efficient sequential way of performing the same computation, under the assumption that the output is measured (and is, therefore, classical).

3.2 Semiclassical solution

Although the circuits for computing the quantum Fourier transform and its inverse are efficient in terms of the total number of gates employed, the majority of these gates operate on two qubits. This makes a practical implementation difficult, since arranging for one qubit to influence another in a desired way is far greater a challenge than evolving a single-qubit closed quantum system in accordance with any unitary transformation.

A method to replace all the two-qubit gates in the circuit performing the quantum Fourier transform by a smaller number of one-qubit gates controlled by classical signals has been developed by Griffiths and Niu [14]. Their approach takes advantage of the fact that the roles of the control and target qubits in any of the two-qubit gates required to carry on the computation of the quantum Fourier transform are interchangeable. Consequently, the quantum circuit in Figure 1 is equivalent to the one depicted in Figure 3 (for inputs restricted to four qubits).

Note that, from this new perspective, the computation of the quantum Fourier transform appears to be a procedure whose steps are of increasing complexity. However, under the assumption that the Fourier transform is immediately followed by a quantum measurement, the complexity of each step in the computation can be made constant. Since a control qubit enters and leaves a two-qubit gate unchanged, it follows that the top qubit in Figure 3 yields the same result regardless of whether it is measured as it exits the circuit or immediately after undergoing the Hadamard transform. In the latter case, the result of the measurement

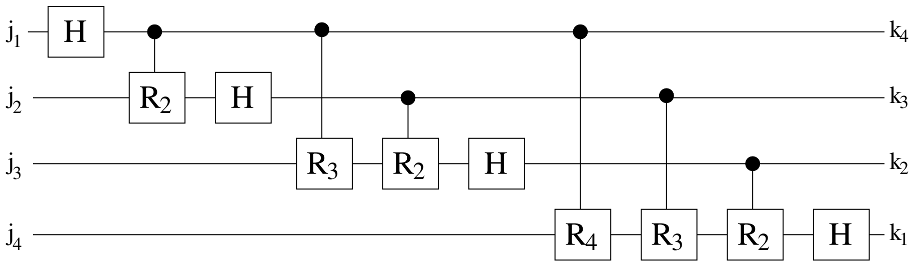

Figure 3: Alternative arrangement of gates in the circuit performing the quantum Fourier transform.

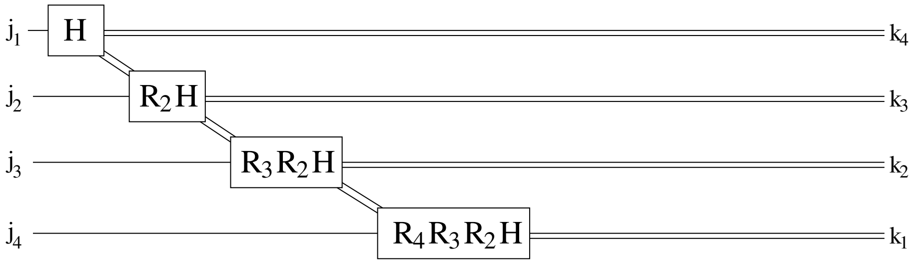

Figure 4: Semiclassical circuit for computing the quantum Fourier transform.

can be used to determine the phase shift that needs to be applied on the second qubit, before it too is subjected to a Hadamard transform and then measured. The phase computed for the second qubit together with the result of the second measurement are passed down as classical inputs for the rotation applied to the third qubit.

The computation proceeds in this manner all the way down to the last qubit, with a phase rotation, a Hadamard gate and a measurement being performed at each step. The process is illustrated in Figure 4, where double lines have been used to denote a classical signal, according to the usual convention. Although the phase shift applied to each qubit is considered a single operation, conceptually, it is a combination of the gates depicted in the corresponding box, with each component being applied only if the controlling qubit was measured as 1.

This semiclassical approach to computing the quantum Fourier transform achieves optimality in terms of the number of elementary unitary transformations that have to be applied. It also has the important advantage of employing only quantum transformations acting on a single qubit at a time. However, there is still room for improvement, as the total time needed to complete the computation can be further squeezed down if parallelism is brought into play. In what follows, we show how a quantum pipeline architecture is able to speed up the computation of the Fourier transform [17].

3.3 Parallel approach

The solution developed in [14] to reduce the complexity of the quantum Fourier transform envisages a purely sequential approach, which is motivated by the same data dependency that causes the complexity of a step to vary with its rank. Nevertheless, there is a certain

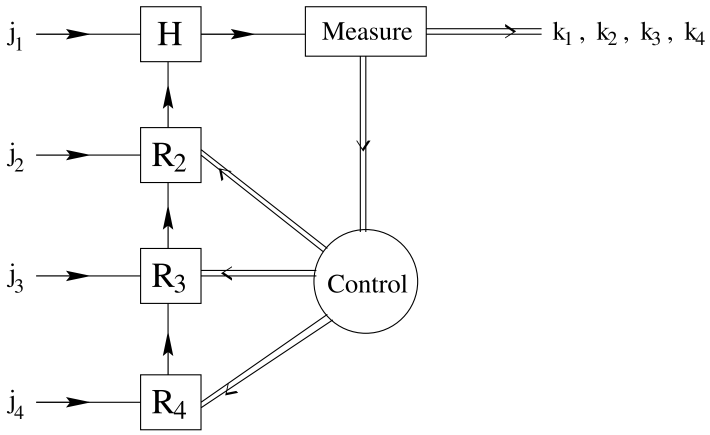

Figure 5: Quantum pipeline array for computing the Fourier transform.

degree of parallelism that is worth exploiting in the computation of the quantum Fourier transform (or its inverse) in order to minimize the overall running time.

Our parallel approach is based on the observation that once a qubit has been measured, all phase shift gates classically controlled by the outcome of that measurement can be applied in parallel. The arrangement, again for just four qubits, is shown in Figure 5. The one-qubit gates are ordered into a linear array having a Hadamard transform at the top and followed by a $\pi/2$ phase shift gate. The phase shift induced by any other gate down the array is just half the rotation performed by the immediately preceding gate.

This architecture allows R_2, R_3 and R_4 to be performed in parallel during the first cycle. Since each phase shift gate acts on a different qubit, they can all be applied simultaneously, if the top qubit yielded a 1 upon measurement. In the second cycle, each qubit in the array travels up one position, except of course for the top one, which has already been measured. Now, depending on the outcome of the second measurement, R_2 and R_3 can be simultaneously effected on the corresponding qubits. In the third cycle, only R_2 is needed and only if the control is 1. The computation ends with the last qubit reaching the Hadamard gate and being measured afterwards. A formal description of the procedure, in the general case, is given below.

Procedure *Parallel_Quantum_Fourier_Transform*

Input: $|j_1 j_2 \cdots j_n\rangle$
Output: $k_1 k_2 \cdots k_n$

for $i = 1$ **to** n **do**
    $|j_i\rangle \longleftarrow H|j_i\rangle$;
    Measure $|j_i\rangle$ as k_{n-i+1};
    if $k_{n-i+1} = 1$ **then**
        for $l = 2$ **to** $n - i + 1$ **do in parallel**
            $|j_{i+l-1}\rangle \longleftarrow R_l|j_{i+l-1}\rangle$;
            $|j_{i+l-1}\rangle$ moves one position up in the array

endfor
endif
endfor

In the worst case, when all qubits are measured as 1, there is no difference between the parallel algorithm outlined above and the sequential solution envisaged by Griffiths and Niu [14] with respect to the overall running time. Assuming, for analysis purposes, that measuring a qubit, applying a phase shift, and performing a Hadamard transformation, each takes one time unit, then the total time necessary to complete the Fourier transform on a quantum register with n qubits is $3n-1$, as the top qubit in both the sequential circuit of Figure 4 and the parallel circuit of Figure 5 does not require a phase shift. This analysis only considers the *quantum* operations that need to be performed. The sequential method also requires some *classical* computation, when the phase shift that is to be applied to each qubit is calculated.

However, in the average case, some of the classical signals controlling the array of phase shift gates in Figure 5 will have been observed as 0, meaning that no phase shifts have to be performed during those respective cycles. In contrast, the sequential solution depicted in Figure 4 requires the application of a phase shift at every step following the first measurement with outcome 1. If the expected probability of a measurement yielding 0 equals the expected probability to observe a 1 following a measurement, then the running time of the parallel solution is shorter than the sequential running time by a difference proportional to the time it takes to effect a number of $O(n)$ phase shift gates, where n is the size of the input register.

The difference between the sequential running time and the parallel running time is maximum when $|j_1\rangle$ is measured as 1 and all the other qubits are observed in the state 0. In this case, the circuit in Figure 4 still performs $n-1$ phase shifts, for a total running time of $3n-1$ time units, while the circuit in Figure 5 executes all $n-1$ phase shifts in parallel during the first cycle, thus completing the computation in $2n+1$ time units.

The second advantage of the parallel approach is that the phase shift gates that need to be applied during the computation are known at the outset, making it easy to set them up beforehand in order to form the required linear array architecture. The systolic mode of operation of the quantum array compensates for the fixed characteristics of each gate, the qubits traversing the array to undergo a specific quantum evolution at each node. In the sequential approach, the phase shift applied to each qubit is not known at the outset, as it is computed on the fly based on the information about the measurements performed so far and transmitted as classical signals. This means that the gates effecting the necessary phase shifts in the semiclassical approach of Griffiths and Niu [14] have to be "programmed" or adjusted during the computation, in order to accommodate a discrete set of possible values for the phase shift.

The semiclassical Fourier transform and its parallelization are applicable to those quantum computations in which the Fourier transform immediately precedes a measurement of the qubits involved in the computation. Furthermore, the quantum systolic array architecture works equally fine if the input is already classical, in which case the restriction to measure the qubits after applying the Fourier transform can be lifted altogether.

When $j_1, j_2, \cdots, j_n$ are classical bits, the topology of the circuit in Figure 5 remains unchanged, except that no measurements are performed and the flow of data through the

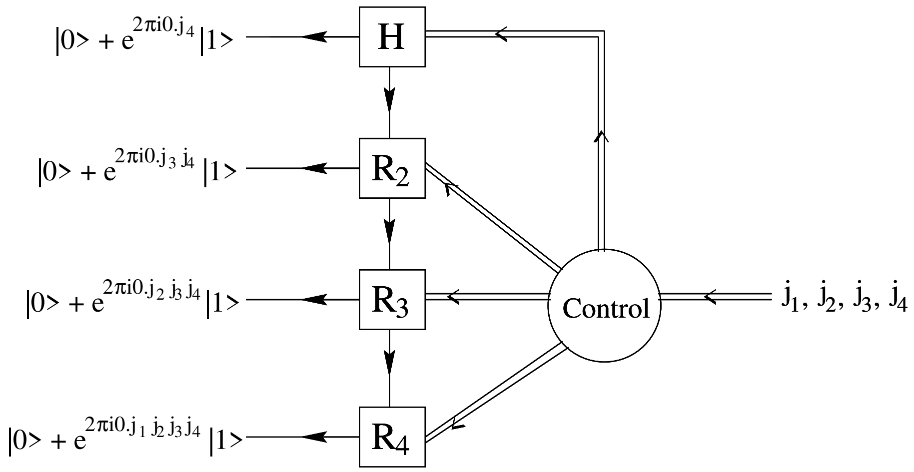

Figure 6: Quantum pipeline array for computing the Fourier transform on classical inputs.

linear array is reversed, as shown in Figure 6. As more data are fed into the linear array through the Hadamard gate, after having "controlled" the parallel execution of a set of phase shifts, the computational complexity of each step increases with its rank. When j_1 enters the array, only the Hadamard gate is active, but with each consecutive step, a new gate down the array joins the ones above it to operate on the qubits traversing the array. Because these gates operate in parallel, the execution time of each step is maintained constant. Also note that, in this case, all outputs are simultaneously obtained during the last step of the computation.

The overall parallel running time, in the worst case, is therefore $2n - 1$ time units, as there are no measurements to perform. In most cases, however, the parallel running time is smaller than the time needed to complete the computation in a purely sequential manner, where each qubit is dealt with one after the other, in decreasing order of their ranks.

Although applying the quantum Fourier transform on a classical input is of little value for quantum computing, the situation is different for quantum cryptography. Distributing classical keys through quantum means is a procedure that may use the quantum Fourier transform and its inverse as encoding and decoding algorithms to protect vital information while in transit [18]. Naturally, the parallel approach employed for the computation of the direct Fourier transform is also applicable, with the same results, to the circuit in Figure 2, performing the inverse Fourier transform.

The difficulty of devising a parallel algorithm for computing the quantum Fourier transform comes from the data dependency between the different steps of the procedure. In most cases, it is exactly this precedence among the steps composing an algorithm that determines the variation in complexity. As a consequence, it is not easy, in general, to design a parallel solution to a problem whose steps are characterized by rank-varying complexity. The data dependency may impose a strict order of execution, making the resulting algorithm inherently sequential (think about Euclid's algorithm again). But, there is also a positive aspect of the data dependency characterizing the quantum Fourier transform. It may be exploited in cryptographic applications, for example to increase the security and intrusion detection rate in quantum key distribution protocols [18].

On the other hand, perhaps there exist computations made up of steps of various rank-dependent complexities, for which the order of execution is of no consequence to the correctness of the computation. Imagine, for instance, a task made up of n steps: $S_1, S_2, \ldots, S_n$, where the steps can be executed in any order, but the more steps we execute before a certain step S_j $(1 \leq j \leq n)$, the more time it will take to complete S_j. For example, S_j may require i elementary operations (time units) if executed i^{th}, for $i = 1, 2, \ldots, n$. This rank-driven increase in complexity may be due to how many pieces of data have to be taken into consideration at each consecutive step, how the data were affected by executing the previous steps, or it may be justified by the size of the partial solution that has to be constructed at each step.

In any case, the problem of coping with steps of ever increasing complexity is avoided altogether by a parallel machine endowed with n processing units. All steps would then be executed simultaneously, and since each step has the rank 1 in such a parallel approach, the computational complexity is kept constant (one time unit, in our example) for all steps. The difference between a sequential and a parallel approach is even more dramatic if the complexity of a step grows faster with its rank. In the case where step S_j $(1 \leq j \leq n)$ needs 2^i elementary operations (time units) to be completed, if executed i^{th}, $i = 1, 2, \ldots, n$, the benefits of using a parallel approach are much higher.

¿From the research viewpoint adopted in this paper, it remains an open problem to investigate if there are quantum instances belonging to the rank-varying computational complexity paradigm for which there is no
pre-determined order of execution of the steps composing the algorithm. Perhaps the reversible nature of quantum evolutions may play some role, in the sense that when a step is executed, it must first undo the transformations performed by all previously executed steps.

The difference in time complexity between the sequential approach and the parallel one, in the computation of the direct or inverse quantum Fourier transform, may seem insignificant from a theoretical perspective, but it proves essential under practical considerations, as we show next.

3.4 Quantum decoherence

Qubits are fragile entities and one of the major challenges in building a practical quantum computer is to find a physical realization that would allow us to complete a computation before the quantum states we are working with become seriously affected by quantum errors. In an ideal setting, we evolve our qubits in perfect isolation from the outside world. But any practical implementation of a quantum computation will be affected by the interactions taking place between our system and the environment. These interactions cause quantum information to leak out into the environment, leading to errors in our qubits. Different types of errors may affect an ongoing computation in different ways, but *quantum decoherence*, as defined below, usually occurs extremely rapidly and can seriously interfere with computing the quantum Fourier transform and its inverse.

In the context of a quantum key distribution protocol [18], consider the task of recovering the original (classical) bit string $j = j_1 j_2 \cdots j_n$ from its quantum Fourier transformed form. The circuit performing this computation (see Figure 2) takes as input n qubits. The state of each qubit can be described by the following general equation:

$$|\psi_k\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{e^{i\theta_k}}{\sqrt{2}}|1\rangle,\ 1 \leq k \leq n \tag{3}$$

where the relative phase θ_k, characterizing the qubit of rank k, depends on the values of bits $j_k, j_{k+1}, \cdots, j_n$. The corresponding density operator is given by

$$\rho_k = |\psi_k\rangle\langle\psi_k| = \frac{1}{2}|0\rangle\langle 0| + \frac{e^{-i\theta_k}}{2}|0\rangle\langle 1| + \frac{e^{i\theta_k}}{2}|1\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|, \tag{4}$$

or in matrix form

$$\rho_k = \frac{1}{2}\begin{bmatrix} 1 & e^{-i\theta_k} \\ e^{i\theta_k} & 1 \end{bmatrix}. \tag{5}$$

The diagonal elements (or the *populations*) measure the probabilities that the qubit is in state $|0\rangle$ or $|1\rangle$, while the off-diagonal components (the *coherences*) measure the amount of interference between $|0\rangle$ and $|1\rangle$ [10]. Decoherence then, resulting from interactions with the environment, causes the off-diagonal elements to disappear. Since that is where the whole information carried by a qubit is stored, the input qubits for computing the inverse Fourier transform are very sensitive to decoherence. When they become entangled with the environment, the interference brought about by the Hadamard gate is no longer possible, as the system becomes effectively a statistical mixture. In other words, decoherence makes a quantum system behave like a classical one.

Naturally, this process is not instantaneous, but it usually occurs extremely rapidly, subject to how well a qubit can be isolated from its environment in a particular physical realization. Because of decoherence, we must obtain the values of $j_1, j_2, \cdots, j_n$ before time limit δ, after which the errors introduced by the coupling with the environment are too serious to still allow the recovery of the binary digits of j.

The precise value of δ will certainly depend on the particular way chosen to embody quantum information, but if δ lies between the parallel completion time and the sequential completion time, then the quantum pipeline array may be the only architecture capable to precisely recover all digits in the binary expansion of j. From a different perspective, the parallel solution allows for longer bit strings to be transmitted between the communicating parties, thus achieving better scalability over the purely sequential approach.

Griffiths and Niu [14] also point to decoherence as a possible problem when discussing their semiclassical solution to computing the quantum Fourier transform, and suggest to counter it by arranging the computation in such a way that the more significant bits of the input register are produced earlier than the less significant ones. This may or may not be possible, depending on the particular characteristics of a certain application. In the case of the inverse Fourier transform used as a decoding method in a quantum key distribution protocol [18], starting to work early on higher rank qubits is not possible because the rank of each qubit is not disclosed until the second stage of the protocol, when all qubits are available to the receiving party. In such a situation, the parallel approach previously described can make the difference between success and failure when computing the quantum Fourier transform or its inverse in a practical setting. Alternatively, since the overall running time scales up with the number of input qubits, parallelism may be a way to improve scalability and still complete the computation before decoherence effects take hold. In this context, we emphasize that scalability and decoherence are the two most important issues in designing a practical quantum computer.

3.5 Time-varying variables

In this section we have seen that the computation of the Fourier transform by quantum means belongs to the class of computations in which the complexity of each step depends on its rank. In addition, if we also take into consideration the properties of the computational environment, we are faced with the negative effects caused by quantum decoherence. Formally, the data stored in the quantum register before time limit δ is significantly different from what the same qubits encode after the decoherence threshold δ. The coupling between our qubits and their surrounding environment effectively places a hard deadline on the computation. After this deadline, the input data (variables) will have changed and if the computation is not yet complete, it has inevitably failed. From this perspective, the computation of the quantum Fourier transform (whether direct or inverse) in the presence of decoherence is an example of the paradigm dealing with *time-varying variables*.

As we have demonstrated above, parallelism can help us cope with variables whose values change over time. The use of a parallel approach becomes critical when the solution to a certain problem must accommodate a deadline. In our case, quantum decoherence places an upper bound on the scalability of computing the quantum Fourier transform or its inverse, and the only chance to reach beyond that limit is through a parallel solution.

4 Quantum error-correction

In the examples presented in the previous section, the complexity of each step evolves with its rank. The more steps are executed before the current one, the higher the computational resources required to complete it. In this section, we still focus on steps of variable complexity, but in this case the variation is *time driven* rather than *rank driven*. In other words, we can have a high computational complexity even for the first step, if we allow some time to pass before starting the computation. The amount of computational resources required to successfully carry out a certain step are directly proportional with the amount of time elapsed since the beginning of the computation. We illustrate this paradigm through the use of error-correcting codes employed to maintain a quantum computation error-free.

The laws of quantum mechanics prevent, in general, a direct application of the classical error-correction techniques. We cannot inspect (measure) at leisure the state of a quantum memory register to check whether an ongoing computation is not off track without the risk of altering the intended course of the computation. Moreover, because of the no-cloning theorem, quantum information cannot be amplified in the same way digital signals can. Correcting quantum errors certainly requires much more ingenuity than fixing classical bits, but the basic idea of using redundancy is still useful.

Like in the classical case, the information contained in a qubit is spread out over several qubits so that damage to any one of them will not influence the outcome of the computation.

In the quantum case, though, the encoding of the logical qubit is achieved through the use of specific resources, by entangling the logical qubit with several ancilla qubits. In this way, the information in the state of the qubit to be protected is spread among the correlations characterizing an entangled state. Paradoxically enough, entanglement with the environment can be fought back using quantum error-correcting codes based on entanglement [23].

4.1 Quantum codes

The construction of all quantum error-correcting codes is based on the surprising, yet beautiful idea of *digitizing the errors*. How can quantum errors be digitized when, as the variables they affect, they form a continuum? The answer lies in the linear nature of quantum mechanics. Any possible error affecting a single qubit can be expressed as a linear combination of no errors (I), bit flip errors (X), phase errors (Z) and bit flip phase errors (Y), where I, X, Z and Y are the Pauli operators describing the effect of the respective errors. Generalizing to the case of a quantum register, an error can be written as $\sum_i e_i E_i$ for some error operators E_i and coefficients e_i. The error operators can be tensor products of the single-bit error transformations or more general multibit transformations. An error correcting code that can undo the effect of any error belonging to a set of correctable errors E_i will embed n data qubits (logical qubits) in $n+k$ code qubits (physical qubits). The joint state of the ensemble of code qubits is subject to an arbitrary error, mathematically expressed as a linear combination of the correctable error operators E_i.

To recover the original encoded state, a syndrome extraction operator has to be applied that uses some ancilla qubits to create a superposition of the error indices i corresponding to those correctable error operators E_i that have transformed the encoded state. Measuring only the ancilla qubits will collapse the superposition of errors, yielding only one index k. But because the ancilla qubits were entangled with the code qubits through the application of the syndrome extraction operator, the side effect of the measurement is that the corruption caused by all error transformations will be undone, save for the one corresponding to index k. Consequently, only one inverse error transformation is required in order to complete the recovery process. In essence, knowing how to deal with a set of fundamental error transformations allows us to tackle any linear combination of them by projecting it to one of the basis components. This process is referred to as *digitizing* or *discretizing* the errors.

Peter Shor's second major contribution to the advancement of quantum computation was the creation in 1995 of an algorithm that could correct any kind of error (amplitude and/or phase errors) affecting a single qubit in a 9-qubit code [25]. In a different approach, Steane studied the interference properties of multiple particle entangled states and managed to devise a shorter, 7-qubit code [27]. The number of qubits necessary for a perfect recovery from a single error was later squeezed down to a minimum of five [6, 15].

Naturally, in order to cope with more than one error at a time, it is necessary to use larger and more elaborate codes. The book of Nielsen and Chuang [20] offers a detailed treatment of quantum codes, explaining how ideas from classical linear codes can be used to construct large classes of quantum codes, as the Calderbank-Shor-Steane (CSS) codes [9, 28], or the stabilizer codes (also known as additive quantum codes), which are even more general than the CSS codes and are based on the stabilizer formalism developed by Gottesman [13].

The major drawback in using large and intricate quantum codes is that the corrective circuit itself is as much prone to errors as the quantum circuit responsible for the main computation. The more errors we are attempting to rectify, the more the complexity and length of the recovery procedure will increase (see [12] for some theoretical bounds on the relationship between the number of data qubits, the total number of entangled qubits and the maximal number of errors that can be tolerated). Thus, we can only increase the size of the error correction codes up to a certain cutoff point, past which no further gains in accuracy can be made.

One attempt to overcome this limitation are the *concatenated* codes. If a certain code uses n physical qubits to encode one logical qubit, a concatenated version of that code is obtained by further encoding each of the n qubits in another block of n. This hierarchical structure (tree) can be further expanded to accommodate as many levels as desired. By adding more levels of concatenation, the overall chance for an error can be made arbitrarily small, provided that the probability of an individual error is kept below a certain critical threshold [24]. Of course, the high cost of using concatenated codes lies in the exponential increase in the number of qubits with the number of levels added.

4.2 Time-varying complexity

This short exposition of the various quantum error-correcting codes devised to maintain the coherence of fragile quantum states and to protect them from dissipative errors caused by spontaneous emissions, for example, clearly shows one thing. The more time it takes to complete a quantum computation, the more errors are introduced in the process, and consequently, the more time, number of ancilla qubits and higher complexity error-correcting schemes that need to be employed. Correcting quantum errors is an important task executed alongside the mainstream computation and its complexity is heavily dependent on time. Steps executed soon after the initialization of the quantum register will require none or low complexity recovery techniques, while steps executed long after the initialization time may require complicated schemes and heavy resources allocated to deal with quantum errors.

As with the other paradigms investigated in this paper, here too parallelism can help avoid this increase in the complexity of the recovery procedure and ultimately ensure the success of the computation. If the steps of the algorithm are independent of one another and can be executed in any order, then the most straightforward application of parallelism is to execute all steps simultaneously and thus complete the computation before any serious errors can accumulate over time. In this way we try to avoid or elude quantum errors rather than deal with them. But parallelism, in the form of redundancy, can also be used to correct quantum errors.

4.3 Error correction via symmetrization

The technique called *error correction via symmetrization* [8, 4] is yet another example of how the duality of quantum-mechanical laws can be exploited for the benefit of quantum computation. Although the measurement postulate severely restricts us in recycling techniques from classical error correction, it can still offer conceptually new ways of achieving error correction that are simply unavailable to classical computers. Error correction via symmetrization relies on the projective effect of measurements to do the job. The technique uses n quantum computers, each performing the same computation. Provided no errors occur, the joint state of the n computers is a symmetric one, lying somewhere in the small symmetric subspace of the entire possible Hilbert space. Devising a clever measurement that projects the joint state back into the symmetric subspace should be able to undo possible errors, without even knowing what the error is.

To achieve this, the n quantum computers need to be carefully entangled with a set of ancilla qubits placed in a superposition representing all possible permutations of n objects. In this way, the computation can be performed over all permutations of the computers simultaneously. Then, by measuring the ancilla qubits, the joint state of the n computers can be projected back into just the symmetric computational subspace, without the errors being measured explicitly. Peres has shown that this technique is most appropriate for correcting several qubits that are slightly wrong, rather than correcting a single qubit that is terribly wrong [22]. Error correction via symmetrization can be applied repeatedly, at regular time intervals, to avoid the accumulation of large errors and continually project the computation back into its symmetric subspace.

No matter which parallel approach is employed, if the required number of quantum processing units is provided, then the algorithm is successful. Simulating the same solution on an insufficient number of quantum computers will lead to a gradual accumulation of the quantum errors up to the point where the results of the computation are compromised.

5 Entanglement

In this section, we focus on the most counterintuitive property exhibited by quantum particles, namely *entanglement*. The components of a quantum system are said to be entangled, if the state of the ensemble cannot be broken down or decomposed into the states of the constituents. Although the state of the system as a whole is well-defined, neither of its components is in a well-defined state.

Entanglement is responsible for the strong correlations exhibited by two or more particles when they are measured, and which cannot be explained by classical means. At an abstract level, entanglement among qubits can be described as the behavior exhibited by a set of interacting variables. When such a variable is subjected to a measurement, the process has consequences on the other variables in the set, as well.

5.1 Interacting variables

Formally, suppose there are n variables $x_0, x_1, \cdots, x_{n-1}$. Although these variables may represent the parameters of a physical or biological system, the following formalism is abstracted away from any particular realization and does not necessarily describe the dynamics of a quantum system. The dependence of each variable on all others induces the system to continually evolve until a state of equilibrium may eventually be reached. In the absence of any external perturbations, the system can remain in a stable state indefinitely. We can model the interdependence between the n variables through a set of functions, as follows:

$$
\begin{aligned}
x_0(t+1) &= f_0(x_0(t), x_1(t), \ldots, x_{n-1}(t)) \\
x_1(t+1) &= f_1(x_0(t), x_1(t), \ldots, x_{n-1}(t)) \\
&\vdots \\
x_{n-1}(t+1) &= f_{n-1}(x_0(t), x_1(t), \ldots, x_{n-1}(t))
\end{aligned}
\tag{6}
$$

This system of equations describes the evolution of the system from state $(x_0(t), x_1(t), \ldots, x_{n-1}(t))$ to state $(x_0(t+1), x_1(t+1), \ldots, x_{n-1}(t+1))$, one time unit later. In the case where the system has reached equilibrium, its parameters will not change over time. It is important to emphasize that, in most cases, the dynamics of the system are very complex, so the mathematical description of functions $f_0, f_1, \ldots, f_{n-1}$ is either not known to us or we only have rough approximations for them.

Assuming the system is in an equilibrium state, our task is to measure its parameters in order to compute a function $\mathcal{F}$, possibly a global property of the system at equilibrium. In other words, we need the values of $x_0(\tau), x_1(\tau), \ldots, x_{n-1}(\tau)$ at moment τ, when the system is in a stable state, in order to compute

$$\mathcal{F}(x_0(\tau), x_1(\tau), \ldots, x_{n-1}(\tau)).$$

Without loss of generality, we can try to estimate the value of $x_0(\tau)$, for instance, by measuring the respective parameter at time τ. Although, for some systems, we can acquire the value of $x_0(\tau)$ easily in this way, the consequences for the entire system can be dramatic. Unfortunately, any measurement is an external perturbation for the system, and in the process, the parameter subjected to measurement may be affected unpredictably.

Thus, the measurement operation will change the state of the system from $(x_0(\tau), x_1(\tau), \ldots, x_{n-1}(\tau))$ to $(x'_0(\tau), x_1(\tau), \ldots, x_{n-1}(\tau))$, where $x'_0(\tau)$ denotes the value of variable x_0 after measurement. In those cases where the measurement process has a non-deterministic effect upon the variable being measured, we cannot estimate $x'_0(\tau)$ in any way. But, regardless of the particular instance of the model, the transition from $(x_0(\tau), x_1(\tau), \ldots, x_{n-1}(\tau))$ (that is, the state before measurement) to $(x'_0(\tau), x_1(\tau), \ldots, x_{n-1}(\tau))$ (that is, the state after measurement) does not correspond to the normal evolution of the system according to its dynamics described by functions f_i, $0 \leq i < n$.

However, because the equilibrium state was perturbed by the measurement operation, the system will react with a series of state transformations, governed by equations (6). Thus, at each time step after τ, the parameters of the system will evolve either towards a new equilibrium state or maybe fall into a chaotic behavior. In any case, at time $\tau + 1$, all n variables have acquired new values, according to the expressions of functions f_i:

$$x_0(\tau+1) = f_0(x'_0(\tau), x_1(\tau), \ldots, x_{n-1}(\tau))$$

$$
\begin{aligned}
x_1(\tau+1) &= f_1(x_0'(\tau), x_1(\tau), \ldots, x_{n-1}(\tau)) \\
&\vdots \\
x_{n-1}(\tau+1) &= f_{n-1}(x_0'(\tau), x_1(\tau), \ldots, x_{n-1}(\tau))
\end{aligned}
\tag{7}
$$

Consequently, unless we are able to measure all n variables, in parallel, at time τ, some of the values composing the equilibrium state

$$(x_0(\tau), x_1(\tau), \ldots, x_{n-1}(\tau))$$

will be lost without any possibility of recovery.

It is important to emphasize that the computational paradigm to which the above setting belongs is not a conventional one. The input data necessary to compute $\mathcal{F}$ is not available at the outset and have to be acquired through measurement operations. Perhaps some readers may object to labeling the process of obtaining the necessary information as *computation*. They may be accustomed to seeing computation from the conventional point of view (like, for example, performing a basic arithmetic operation on a pair of numbers). However, the qualitatively new ways of manipulating information nowadays is forcing us to challenge the limitations of the classical computational paradigm and adopt a broader perspective (often called *unconventional* or *non-classical*) on computation [29].

¿From this new perspective, a computing machine is seen as an open system whose output depends on the interaction with its environment, a system capable of taking on new information (either communicated to it by an external agent or acquired directly through measurements). The emergence of this new model of computation is motivated by applications as diverse as data acquisition in signal processing [21] and the control of nuclear power plants [5]. Furthermore, such a computational paradigm can be realized through various physical means including, of course, a quantum mechanical one [19].

5.2 Quantum distinguishability

The problem of distinguishing among entangled quantum states is a quantum mechanical instance of the formalism detailed above. Suppose we have a fixed set of quantum states described using the usual Dirac notation $|\Psi_i\rangle$ ($1 \le i \le n$) known to both Alice and Bob. Alice randomly chooses a state from the set and prepares a qubit (or set of qubits) in that particular state. She then gives the qubit(s) to Bob who is free to investigate them in any way he likes. To be more specific, Bob can apply any kind of measurement on the qubit(s) and possibly process and/or interpret the information acquired through measurement. In the end, his task is to identify the index i of the state characterizing the qubit(s) Alice has given him. The only case in which a set of quantum states can be reliably (that is, 100% of the time) distinguished from one another is if they are pairwise orthogonal.

Now consider the case in which we try to distinguish among the four Bell states $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, $\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$, $\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$, $\frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$.

by resorting only to direct quantum measurements (in other words, no quantum transformations are possible before a measurement). In these circumstances, any sequential approach (that is, measuring the qubits one after the other) will be of no help here, regardless of the basis in which the measurements are performed. By measuring the two qubits, in sequence, in the computational basis, Bob can distinguish the states $\frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle)$ from $\frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle)$. He does this by checking if the outcomes of the two measurements are the same or not. But this kind of measurement makes it impossible to differentiate between $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$, or between $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ and $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$.

Alternatively, Bob can decide to perform his measurements in a different basis, like $(|+\rangle, |-\rangle)$, where the basis vectors are

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Due to the fact that

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} = \frac{|++\rangle + |--\rangle}{\sqrt{2}}$$

and

$$\frac{|00\rangle - |11\rangle}{\sqrt{2}} = \frac{|+-\rangle + |-+\rangle}{\sqrt{2}},$$

Bob can now reliably distinguish the quantum state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ from $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$. Indeed, if the two qubits yield identical outcomes when measured in this new basis, then we can assert with certainty that the state was not $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$. Similarly, if the measurement outcomes for the qubits are different, the original state could not have been $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Unfortunately, in this new setup, the quantum states $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ become indistinguishable and the same is true about $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$ and $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$.

The computational bases $(|0\rangle, |1\rangle)$ and $(|+\rangle, |-\rangle)$ are, respectively, the two extremities of an (theoretically) infinite number of choices for the basis relative to which the quantum measurements are to be performed. But even though the separation line between the four Bell states will drift with the choice of the basis vectors, the two extreme cases discussed above offer the best possible distinguishability.

Intuitively, this is due to the entanglement exhibited between the two qubits in all four states. As soon as the first qubit is measured (regardless of the basis), the superposition describing the entangled state collapses to the specific state consistent with the measurement result. In this process, some of the information originally encapsulated in the entangled state is irremediably lost. Consequently, measuring the second qubit cannot give a complete separation of the four EPR states. But the Bell states do form an orthonormal basis, which means that (at least theoretically) they can be distinguished by an appropriate quantum measurement. However, this measurement must be a *joint* measurement of both qubits simultaneously, in order to achieve the desired distinguishability. Not surprisingly, this is very difficult to accomplish in practice.

The distinguishability of the four Bell (or EPR) states is the key feature in achieving superdense coding [7]. However, in the experimental demonstration of this protocol [16] two of the possibilities cannot be distinguished from one another, precisely because of the difficulties associated with implementing a joint measurement.

5.2.1 Generalization

A more compact representation of the Bell basis is through a square matrix where each column is a vector describing one of the Bell states:

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix}$$

The elements of each column are the amplitudes or proportions in which the computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ are present in the respective EPR state.

This scenario can be extended to ensembles of more than two qubits. The following matrix describes eight different entangled states that cannot be reliably distinguished unless a joint measurement of all three qubits involved is performed:

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

In general, for a quantum system composed of n qubits, one can define the following 2^n entangled states of the system:

$$\begin{aligned} &\frac{1}{\sqrt{2}}(|000\cdots 0\rangle \quad \pm \quad |111\cdots 1\rangle) \\ &\frac{1}{\sqrt{2}}(|000\cdots 1\rangle \quad \pm \quad |111\cdots 0\rangle) \\ &\qquad\qquad\vdots \\ &\frac{1}{\sqrt{2}}(|011\cdots 1\rangle \quad \pm \quad |100\cdots 0\rangle) \end{aligned} \tag{8}$$

These vectors form an orthonormal basis for the state space corresponding to the n-qubit system. The only chance to differentiate among these 2^n states using quantum measurement(s) is to observe the n qubits simultaneously, that is, perform a single joint measurement of the entire system. In the given context, *joint* is really just a synonym for *parallel*. Indeed, the device in charge of performing the joint measurement must posses the ability to "read" the information stored in each qubit, in parallel, in a perfectly synchronized manner. In this sense, at an abstract level, and just for the sake of offering a more intuitive understanding of the process, the measuring apparatus can be viewed as having n probes. With all probes operating in parallel, each probe can "peek" inside the state of one qubit, in a perfectly synchronous operation. The information gathered by the n probes is seen by the measuring device as a single, indivisible chunk of data, which is then interpreted to give one the 2^n entangled states as the measurement outcome.

¿From a mathematical (theoretical) point of view, such a measurement operator can be easily constructed by defining each of the 2^n states that are to be distinguished to be a projector associated with the measurement operation. We are well aware though, that a physical realization of this mathematical construction is extremely difficult, if not impossible to achieve in practice, with today's technology. The experimental demonstration of the superdense coding protocol mentioned at the end of previous section clearly shows this difficulty (for just two qubits!). Yet, if there is any hope to see a joint measurement performed in the future, then only a device operating in a parallel synchronous fashion on all n qubits (as explained above) would succeed.

It is perhaps worth emphasizing that if such a measurement cannot be applied then the desired distinguishability can no longer be achieved regardless of how many other measuring operations we are allowed to perform. In other words, even an infinite sequence of measurements touching at most $n-1$ qubits at the same time cannot equal a single joint measurement involving all n qubits.

Furthermore, with respect to the particular distinguishability problem that we have to solve, a single joint measurement capable of observing $n-1$ qubits simultaneously offers no advantage whatsoever over a sequence of $n-1$ consecutive *single* qubit measurements. This is due to the fact that an entangled state like

$$\frac{1}{\sqrt{2}}(|000\cdots 0\rangle + |111\cdots 1\rangle)$$

cannot be decomposed neither as a product of $n-1$ individual states nor as a product of two states (one describing a single qubit and the other describing the subsystem composed of the remaining $n-1$ qubits). Any other intermediate decomposition is also impossible.

Overall, our distinguishability problem can only be tackled successfully within a parallel approach, where we can measure all qubits simultaneously. In this sense, distinguishing among entangled quantum states can be viewed as a quantum variant of the measure-compute-set problem formulated in [2], which also admits only a parallel solution.

The inherent parallelism characterizing the task of distinguishing among entangled quantum states through measurements implies that a device capable of measuring at most n qubits simultaneously (where n is a fixed, finite number) will fail to solve the distinguishability problem for $n+1$ qubits. For this reason, our example joins the other paradigms illustrated in quantum mechanical terms throughout this paper, to support the idea advanced in [1] about the impossibility of realizing the concept of a Universal Computer. This result holds as long as the candidate Universal Computer cannot apply its (internal) set of basic processing operations ("gates") onto systems from the outside world. In other words, its processing capabilities can only be exercised on data already stored in its (internal) memory. In the case that we have described, all input data on which the machine can work must be acquired through measurement(s).

Conceptually, distinguishing among entangled quantum states is a quantum example of measuring interdependent variables. In this particular quantum instance, the interdependence between variables takes the form of entanglement between qubits, the phenomenon ultimately responsible for making a parallel approach imperative. But not only measuring entangled states requires a parallel solution, quantum evolutions that have to maintain a certain entangled state may also resort to parallelism in order to achieve their goal. In what follows, we investigate entanglement as a global mathematical constraint that has to be satisfied throughout a quantum computation.

5.3 Transformations obeying a global condition

Some computational problems require the transformation of a mathematical object in such a way that a property characterizing the original object is to be maintained at all times throughout the computation. This property is a global condition on the variables describing the input state and it must be obeyed at every intermediate step in the computation, as well as for the final state. Geometric flips, map recoloring and rewriting systems are three examples of transformations that can be constrained by a global mathematical condition [3].

Here, we show that some quantum transformations acting on entangled states may also be perceived as computations obeying a global mathematical constraint. Consider, for example, an ensemble of n qubits sharing the following entangled state:

$$\frac{1}{\sqrt{2}}|000\cdots 0\rangle + \frac{1}{\sqrt{2}}|111\cdots 1\rangle. \tag{9}$$

The entanglement characterizing the above state determines a strict correlation between the values observed in case of a measurement: either all qubits are detected in the state 0 or they are all seen as 1. Suppose that this correlation has to be maintained unaltered, regardless of the local transformations each of the qubits may undergo. Such a transformation may be the application of a NOT quantum gate to any of the qubits forming the ensemble. After such an event, the particular entangled state given in eq. (9) is no longer preserved and as a consequence, the correlation between the qubits will be altered. The qubit whose state was "flipped" will be observed in the complementary state, with respect to the other qubits. The global mathematical constraint is no longer satisfied.

Parallelism can once again make the difference and help maintain the required entangled state. If, at the same time one or more of the qubits are "flipped", we also apply a NOT gate to all remaining qubits, simultaneously, then the final state coincides with the initial one. In this way, although the value of each qubit has been switched, the correlation we were interested to maintain remains the same. Also note that any attempt to act on less than n

qubits simultaneously is doomed to failure.

The state given in eq. (9) is not the only one with this property. Any entangled state from the orthonormal basis set (8) could have been used in the example presented above. The correlation among the qubits would have been different, but the fact that applying a NOT gate, in parallel, to all qubits does not change the quantum state of the ensemble is true for each entangled state appearing in system (8). Perhaps the scenario described above can be extended to other quantum transformations beside the NOT gate. Another, perhaps more interesting generalization would be a quantum computation that has to maintain entanglement as a generic, global mathematical constraint and not a specific type of entanglement with a particular correlation among the qubits involved. Such a computation would allow entanglement to change form, but the mathematical definition of entanglement would still have to be obeyed at each step, with each transformation.

6 Conclusion

In this paper, we have analyzed a series of computational paradigms in which the information is encoded and processed using quantum mechanical means. In each particular case we have discussed, a parallel approach offers the best, if not the only way of seeing the task accomplished. This has two important consequences with respect to universality. In the first place, it proves that parallelism is a universal concept, transcending the boundaries imposed by a particular way of representing and transforming information. The computational problems addressed herein clearly demonstrate the value of a parallel solution for quantum computation and information, confirming the capital role played by parallelism in the theory of computation.

It is important to note that we refer here to the common understanding of the term *parallelism* and not to *quantum parallelism*. The latter syntagm is used to denote the ability to perform a certain computation simultaneously on all terms of a quantum superposition, regardless of the number of qubits composing the quantum register whose state is described by that superposition. As opposed to this interpretation, we refer to parallelism as the ability to act simultaneously on a certain number of qubits. Thus, we can rightfully assert that parallelism transcends the laws of physics and represents a fundamental aspect of computation, regardless of the particular physical way chosen to embody information.

Paradigm	**Description**	**Quantum Example**
1. Rank-varying complexity	The complexity of a computational step is a function of its rank.	Quantum Fourier Transform
2. Time-varying complexity	The complexity of a step depends on when it is executed.	Quantum error correction
3. Time-varying variables	Input variables change their values with time.	Quantum decoherence
4. Interacting variables	Input data are interconnected, affecting each other's behavior.	Measuring entangled states
5. Computations obeying a global condition	A certain global property has to be maintained throughout the computation.	Maintaining entanglement

A more subtle connection exists between parallelism and the hypothetical notion of a Universal Computer, a machine with fixed and finite characteristics, capable of simulating any other computing device. One thing common to all examples presented in this paper is that if the degree of parallelism required to solve a certain problem is not available, then no approach can be successful. To be more precise, if n processing units are needed to solve a problem, then a machine endowed with $n-1$ processors is not able to complete the task. In other words, such a machine is not capable of simulating the successful parallel algorithm running on the n-processor device (even if it is given an unbounded amount of time and memory to perform the simulation). And since the principle of simulation is the one supporting the myth of a Universal Computer, we must conclude that the existence of such a machine is impossible.

We also wish to draw attention on the unconventional aspect of the computing paradigms responsible for uncovering this result on universality. This further motivates the study of non-traditional computational environments and proves that sometimes the results can be surprising.

Acknowledgment This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

[1] Selim G. Akl. The myth of universal computation. In R. Trobec, P. Zinterhof, M. Vajteršic, and A. Uhl, editors, *Parallel Numerics, Part 2, Systems and Simulation*, pages 211–236. University of Salzburg, Austria and Jožef Stefan Institute, Ljubljana, Slovenia, 2005.

[2] Selim G. Akl. Coping with uncertainty and stress: A parallel computation approach. *International Journal of High Performance Computing and Networking*, 4(1/2):85–90, 2006.

[3] Selim G. Akl. Evolving computational systems. In Sanguthevar Rajasekaran and John H. Reif, editors, *Parallel Computing: Models, Algorithms, and Applications*. CRC Press, 2007.

[4] Adriano Barenco, André Berthiaume, David Deutsch, Artur Ekert, Richard Jozsa, and Chiara Macchiavello. Stabilization of quantum computations by symmetrization. `http://xxx.lanl.gov/abs/quant-ph/9604028`, April 1996.

[5] B. Barutçu, S. Şeka, E. Ayaz, and E. Türkcan. Real-time reactor noise diagnostics for the Borsele (PWR) nuclear power plant. *Progress in Nuclear Energy*, 43(1–4):137–143, 2003.

[6] Charles H. Bennett, David P. DiVincenzo, John A. Smolin, and William K. Wootters. Mixed state entanglement and quantum error correction. *Physical Review A*, 54:3824–3851, 1996. `http://arxiv.org/abs/quant-ph/9604024`.

[7] Charles H. Bennett and Stephen J. Wiesner. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69(20):2881–2884, 1992.

[8] André Berthiaume, David Deutsch, and Richard Jozsa. The stabilization of quantum computation. In *Proceedings of the Workshop on Physics and Computation: PhysComp '94*, pages 60–62, Los Alamitos, CA, 1994, 1994. IEEE Computer Society Press.

[9] A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098–1106, 1996. `http://arxiv.org/abs/quant-ph/9512032`.

[10] C. Cohen-Tannoudji, B. Diu, and F. Laloe. *Quantum Mechanics*, volume 1 and 2. Wiley, New York, 1977.

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 2001.

[12] Artur Ekert and Chiara Macchiavello. Quantum error correction for communication. *Physical Review Letters*, 77:2585–2588, 1996.

[13] Daniel Gottesman. Class of quantum error-correcting codes saturating the quantum hamming bound. *Physical Review A*, 54:1862–1868, 1996. `http://arxiv.org/abs/quant-ph/9604038`.

[14] Robert Griffiths and Chi-Sheng Niu. Semiclassical Fourier transform for quantum computation. *Physical Review Letters*, 76:3228–3231, 1996.

[15] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correction code. `http://arxiv.org/abs/quant-ph/9602019`, February 1996.

[16] Klaus Mattle, Harald Weinfurter, Paul G. Kwiat, and Anton Zeilinger. Dense coding in experimental quantum communication. *Physical Review Letters*, 76(25):4656–4659, 1996.

[17] Marius Nagy and Selim G. Akl. Coping with Decoherence: Parallelizing the Quantum Fourier Transform. In *19th International Conference on Parallel and Distributed Computing Systems*, pages 108–113, San Francisco, California, September 2006.

[18] Marius Nagy and Selim G. Akl. Quantum key distribution revisited. Technical Report 2006-516, School of Computing, Queen's University, Kingston, Ontario, June 2006.

[19] Marius Nagy and Selim G. Akl. Quantum measurements and universal computation. *International Journal of Unconventional Computing*, 2(1):73–88, 2006.

[20] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[21] G. Okša, M. Bečka, and M. Vajteršic. Parallel computation with structured matrices in linear modeling of multidimensional signals. *Parallel and Distributed Computing Practices*, 5(3):289–299, 2004.

[22] Asher Peres. Error symmetrization in quantum computers. `http://xxx.lanl.gov/abs/quant-ph/9605009`, May 1996.

[23] John Preskill. Fault-tolerant quantum computation. In Hoi-Kwong Lo, Sandu Popescu, and Tim Spiller, editors, *Introduction to quantum computation and information*, pages 213–269. World Scientific, 1998. `http://xxx.lanl.gov/abs/quant-ph/9712048`.

[24] John Preskill. Reliable quantum computers. *Proceedings of the Royal Society of London A*, 454:385–410, 1998. `http://xxx.lanl.gov/abs/quant-ph/9705031`.

[25] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52:2493–2496, October 1995.

[26] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *Special issue on Quantum Computation of the SIAM Journal on Computing*, 26(5):1484–1509, October 1997.

[27] Andrew M. Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77(5):793–797, July 29, 1996.

[28] Andrew M. Steane. Multiple particle interference and quantum error correction. *Proceedings of the Royal Society of London A*, 452:2551–2576, 1996.

[29] Susan Stepney, Samuel L. Braunstein, John A. Clark, Andy Tyrrell, Andrew Adamatzky, Robert E. Smith, Tom Addis, Colin Johnson, Jonathan Timmis, Peter Welch, Robin Milner, and Derek Partridge. Journeys in non-classical computation I: A grand challenge for computing research. *International Journal of Parallel, Emergent and Distributed Systems*, 20(1):5–19, March 2005.