Quantum Computing and Abstract State Machines

E. Grädel and A. Nowack

Mathematische Grundlagen der Informatik,
RWTH Aachen, D-52056 Aachen, Germany,
{graedel, nowack}@informatik.rwth-aachen.de

Abstract. The ASM thesis says that every algorithm, of any kind, can be modeled step by step and on its natural abstraction level by an abstract state machine. The thesis has been derived from basic principles for sequential algorithms, and for parallel synchronous algorithms. The main result of this paper is that the ASM thesis also holds for quantum algorithms.
We first show that, and how, a general model of quantum algorithms (based on the combination of a classical computer with quantum circuits) can be modeled by abstract state machines. Following the approach of Blass and Gurevich to parallel algorithms, we then formulate general postulates for quantum algorithms and show that every algorithm satisfying these postulates is simulated by an appropriate ASM. These "quantum ASMs" are special cases of the parallel ASMs of Blass and Gurevich, but the only unbounded parallelism is inside the quantum operations (unitary transformations and measurements).

1 Introduction

Abstract state machines (ASMs) provide a model for computation on abstract structures, which is the basis of a successful methodology for specification of hardware and software systems, with practical applications in industrial environments (see e.g. [4]). But ASMs are also interesting for the theory of computation, for instance for the very fundamental question of what constitutes an algorithm. The ASM thesis says that *"every algorithm of any kind is modeled, step by step and on its natural abstraction level, by an abstract state machine"*. There is considerable empirical evidence for this thesis, and in important cases, the thesis can actually be derived from basic postulates. In [3], Gurevich showed that this is the case for *sequential algorithms*. He formulated three basic postulates for sequential algorithms and proved that every algorithm satisfying these three postulates can be simulated, step by step, by a sequential ASM. It should be stressed that this notion of simulation or, in other words, the equivalence of two algorithms, is meant in a much more precise way than, for instance, in the Church-Turing thesis where simulation just means to have the same input-output-behaviour (to compute the same function). Here one requires complete behavioural equivalence: the two algorithms have the same states and initial states, the same one-step transformation, and produce at every step the same output.

Later, Blass and Gurevich [1] extended that work to *parallel algorithms* which still work in sequential time, i.e. in discrete time steps governed by a global clock but with unbounded parallelism in each time step. For the more general model of distributed, asynchronous algorithms, there is so far no such treatment, but work seems to be under way.

However, it has been repeatedly asked, and so far not been answered in a satisfactory way, whether the ASM thesis also holds for nonstandard models of algorithms such as *quantum algorithms.* One of the reasons, why this question has not really been looked into, may be the widespread view in academia and industry that quantum computing provides a purely theoretical playground for mathematicians and physicists rather than "real algorithms". Indeed the main theoretical success of quantum computing is the polynomial-time quantum factorization algorithm by Shor [7], but on the practical side there are enormous difficulties for building quantum computers with more than a few qubits. The recent news that scientists at IBM have sucessfully factored the number 15 using a quantum computer with seven qubits [8] has probably not shaken that view. Still, the question remains, and provides a challenge for the advocates of the ASM thesis. First of all, to our present knowledge, the laws of physics seem not to exclude that large quantum computers can eventually be built, so the present state of quantum computers may just reflect our technological maturity rather than the potential of quantum computing. Second, quantum computing defines a mathematical model of computation (in fact: several models), and a thesis that comes with the ambition of sheding light on the general notion of algorithm should take care of this model as well.

Thus, we felt that the ASM thesis should apply also to quantum algorithms, and our intuition told us that it does. The purpose of this note is to explain that and how ASMs can model quantum algorithms.

2 Abstract State Machines

This paper assumes familiarity with the ASM model, in particular with Gurevich's paper [3] on sequential ASMs and the Blass-Gurevich paper [1] on parallel ASMs. Please consult these papers for unexplained definitions and terminology used here. In [1, 3] the ASM thesis for sequential and for parallel algorithms is derived from some basic postulates. Let us very briefly recall some of them. For sequential algorithms we follow here the presentation of [1] (rather than [3]) which makes the output that the algorithm produces at each step explicit, in form of a function out_A.

Postulates for sequential algorithms. A sequential algorithm with output is an object A satisfying the following three postulates.

S1: Sequential Time. A is associated with a set $\mathcal{S}(A)$ of *states*, a subset $\mathcal{I}(A) \subseteq \mathcal{S}(A)$ of *initial states*, a function $\tau_A : \mathcal{S}(A) \to \mathcal{S}(A)$, called the *one-step transformation* of A, and a function out_A assigning to each state $X \in \mathcal{S}(A)$ a set (or multiset) of elements of X, called the output of A at state X.

S2: Abstract State. All states of A are first-order structures with the same finite vocabulary and the same universe. Both $\mathcal{S}(A)$ and $\mathcal{I}(A)$ are closed under isomorphisms, and any isomorphism between two states X and Y is also an isomorphism between $\tau_A(X)$ and $\tau_A(Y)$, and maps $out_A(X)$ to $out_A(Y)$.

S3: Uniformly Bounded Exploration. There is a finite set T of terms in the vocabulary of A such that, whenever two states X and Y coincide on T, then the update sets and outputs of A on X and Y are the same: $\Delta(A, X) = \Delta(A, Y)$ and $out_A(X) = out_A(Y)$.

We recall that the update set $\Delta(A, X)$ of A on state X represents the atomic changes in X produced by the algorithm A. It can be defined as the set of expressions $f(\overline{a}) = b$ that are true in $\tau_A(X)$ but not in X (where f is a function symbol, and $\overline{a}$ and b are elements of the universe).

A sequential ASM is an ASM that uses only

- function updates: $f(t_1, \ldots, t_m) := t_0$,
- parallel combination of two rules: `do` R_1, R_2 `enddo`,
- if-then-else rules: `if` φ `then` R_1 `else` R_2 `endif` where φ is a Boolean term (i.e. a formula without quantifiers), and
- output rules: `output` t, where t is any term.

The main result of [3] is that every sequential algorithm satisfying postulates (S1) - (S3), can be simulated, step by step, by a sequential ASM. Step by step simulation means that the two algorithms have the same states and initial states, the same one-step transformation, and the same output function.

Parallel algorithms and parallel ASMs. The notion of a parallel algorithm is meant here to capture algorithms that do still work in discrete time steps, governed by a global clock (as opposed to distributed, asynchronous algorithms), but with unbounded parallelism in each time step. Hence, parallel algorithms need not satisfy the Uniformly Bounded Exploration Postulate (S3). We will see that quantum algorithms are, in this sense, parallel algorithms.

The postulates for parallel algorithms cannot be explained here in detail. We limit ourselves to the aspects that we need for dealing with quantum algorithms. A parallel algorithm splits into smaller processes. Ultimately this splitting leads to atomic processes (called *proclets*), each of which executes the same (sequential!) algorithm. In the simplest case, the global update set of the algorithm is just the union of the 'local' updates produced independently by the proclets. In general however, parallel algorithms need to combine and integrate the updates produced by their proclets to a global update set in a more complicated way that involves communication between proclets.

Blass and Gurevich argue that this communication of proclets and the manipulation of messages and update sets is best described in terms of *multisets* *and multiset operations.* The difference of multisets from sets is that multisets may have multiple occurrences of the same element. Formally, a multiset M over A is given by a function $\mathrm{Mult}_M : A \to \mathbb{N}$ giving for each $a \in A$ the multiplicity of $a \in M$ (only finite multiplicities are used here). We use the notation $\{\!\{\ldots\}\!\}$ to denote multisets. In particular, for a term $t(x)$, a set or multiset r, and a formula $\varphi(x)$, we write $\{\!\{t(x) : x \in r : \varphi(x)\}\!\}$ to denote the multiset M of all values of $t(x)$ subject to the condition that x is in r and satifies φ; more formally $\mathrm{Mult}_M(a) = \sum_{x:\varphi(x) \text{ and } t(x)=a} \mathrm{Mult}_r(x)$.

The **Background Postulate** for parallel algorithms states that the abstract states of parallel algorithms contain (among other things) all finite multisets of elements of the state and some basic operations on them. This is reflected in the definition of parallel ASMs which extend sequential ASMs in two essential ways:

Comprehension terms: Given terms $t(x), r$, and $\varphi(x)$ one may build the comprehension terms $\{\!\{t(x) : x \in r : \varphi(x)\}\!\}$.

Unbounded parallelism: If x is a variable, $R(x)$ is a rule and r is a term without free occurrences of x, then `forall` $x \in r$ `do` $R(x)$ `enddo` is a rule.

Note that both operations require parallelism (they do not satisfy the Uniformly Bounded Exploration Postulate). Also for parallel algorithms, the ASM thesis can be derived from appropriate postulates [1].

3 Models of Quantum Computation

For background on quantum computing we refer to the textbooks [2, 5, 6]. There are several theoretical models of quantum algorithms, including quantum circuits, quantum Turing machines, quantum automata, and quantum programming. What all these models have in common is that a *state* of a quantum algorithm is a *superposition of states* of a corresponding classical model.

Mathematically, a state of a quantum algorithm is a unit-length vector $|\psi\rangle \in H$ of a Hilbert space[1].

Quantum bits and quantum registers. A quantum bit (shortly, qubit) is a vector in the two-dimensional Hilbert-space H_2 spanned by the orthonormal vectors $|0\rangle$ and $|1\rangle$. Hence, a qubit has the form $\alpha|0\rangle + \beta|1\rangle$ with $\|\alpha\|^2 + \|\beta\|^2 = 1$. If we *measure* a such a qubit, the probability to observe 0 is $\|\alpha\|^2$ and the probability to observe 1 is $\|\beta\|^2$. In the case of an n-qubit quantum register, we work in the 2^n-dimensional Hilbert-space $H_{2^n} = H_2 \otimes \cdots \otimes H_2$ with orthonormal basis $\{|w\rangle : w \in \{0,1\}^n\}$ which we always assume to be ordered lexicographically, with $|00\cdots 0\rangle$ as first bases vector and $|11\cdots 1\rangle$ as the last one (the order of a basis is important for the representation of linear transformations by matrices). A state of such a quantum register has the form $\sum_{w\in\{0,1\}^n} \alpha_w |w\rangle$ with $\sum_{w\in\{0,1\}^n} \|\alpha_w\|^2 = 1$.

[1] A Hilbert space is a complex vector space H, equipped with an inner product mapping pairs of vectors $|x\rangle, |y\rangle$ to complex numbers $\langle x|y\rangle$. Further H must be complete with respect to the norm $\|x\| := \sqrt{\langle x|x\rangle}$ induced by the inner product (i.e. every Cauchy sequence in H has a limit in H).

A state $|\psi\rangle$ in the product space $H \otimes H'$ is decomposable if it can be written as a tensor product $|\psi\rangle = |\varphi\rangle \otimes |\varphi'\rangle$ with $|\varphi\rangle \in H$ and $|\varphi'\rangle \in H'$. Otherwise it is called *entangled*.

Quantum circuits. The most influential model for quantum computation (probably due to the fact that most of the work on quantum algorithms is done by physicists) has been the quantum circuits model. There is not a uniform terminology here. While some authors use the notions quantum circuits, quantum algorithm and quantum computers interchangeably, others reserve the term quantum circuit for a sequence of quantum gates (with or without measurement), and view quantum computers or quantum algorithms as a more general model that may combine classical parts with quantum circuits and measurement steps. We follow here the second approach.

Definition 1. A *quantum gate on m qubits* is a unitary transformation U on a 2^m-dimensional Hilbert space H_{2^m}. Such a tranformation can be described by a unitary matrix[2] which we also denote by U. For our purposes, it is convenient to view this matrix as a function $U : \{0,1\}^m \times \{0,1\}^m \to \mathbb{C}$. The entry $U(x,y)$ gives the probability amplitude of the transition from base state $|y\rangle$ to base state $|x\rangle$.

Example. The Hadamard gate operates on H_2 and is defined by the matrix

$$H = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

which maps $|0\rangle$ to $|0'\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $|1'\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

Example. The CNOT (controlled NOT) gate is another gate which is used rather often. It operates on $H_4 = H_2 \otimes H_2$ and can be described by the operation $|a,b\rangle \to |a, a \oplus b\rangle$ or, equivalently, by the matrix

$$C = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{pmatrix}.$$

Definition 2. Let Ω be a fixed collection of quantum gates. A *quantum circuit* on n-qubits over Ω is a unitary transformation on H_{2^n} which is defined by a finite sequence of operations of the form

$$\text{“apply } U \text{ to the qubits } i_1, \ldots, i_m\text{”} \tag{1}$$

where U is an m-qubit gate from Ω, and $i_1, \ldots, i_m$ are distinct indices in $\{1, \ldots, n\}$.

[2] A matrix U is unitary, if its transpose conjugate U^* is its inverse: $U^*U = I$.

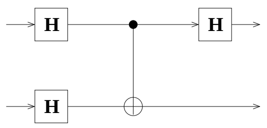
Mathematically, operation 1 is described by the matrix

$$U_{i_1 \ldots i_m} := P^{-1}_{i_1 \ldots i_m}(U \otimes I_{2^{n-m}})P_{i_1 \ldots i_m}$$

where $P_{i_1 \ldots i_m}$ is a permutation matrix moving the qubits $i_1, \ldots, i_m$ to the positions $1, \ldots, m$ and $(U \otimes I_{2^{n-m}})$ is the tensor product of U (operating on the first m qubits) with the identity matrix of dimension 2^{n-m} (operating on the remaining qubits). Given the $2^m \times 2^m$-matrix U, and the numbers $i_1, \ldots, i_m$ the entries of the $2^n \times 2^n$-matrix $U_{i_1 \ldots i_m}$ are

$$U_{i_1 \ldots i_m}(x, y) := \begin{cases} U(x_{i_1} \cdots x_{i_m}, y_{i_1} \cdots y_{i_m}) & \text{if } x_k = y_k \text{ for all } k \notin \{i_1, \ldots, i_m\} \\ 0 & \text{otherwise.} \end{cases}$$

Remark. A popular notation for quantum circuits represents each qubit by a wire and the gates operating on them by boxes or circles (with special symbols for popular gates like Hadamard or CNOT). Here is an example.



The meaning of this picture is that before the application of the CNOT gate Hadamard gates are applied to both qubits separately. Afterwards, a Hadamard gate is applied to the first qubit. Mathematically, the circuit is described by the matrix $(H \otimes I_2)C(H \otimes H)$. Consider its operation on $|01\rangle$. The application of the Hadamard-gates leads to $\frac{1}{2}(|00\rangle + |10\rangle - |01\rangle - |11\rangle)$. The application of the CNOT-gate then leads to $\frac{1}{2}(|00\rangle + |11\rangle - |01\rangle - |10\rangle)$. The application of the Hadamard-gate to the first qubit finally leads to $\frac{1}{\sqrt{2}}(|10\rangle - |11\rangle)$.

Measurement. Let $H = M_1 + \cdots + M_k$ be a decomposition of the Hilbert space H into orthogonal subspaces. Then every $|\psi\rangle \in H$ can be written in a unique way as $|\psi\rangle = |\psi_1\rangle + \cdots + |\psi_k\rangle$, with $|\psi_i\rangle \in M_i$. Measurement of a state $|\psi\rangle$ (of unit length) with respect to the *observable* $\{M_1, \ldots, M_k\}$ means that $|\psi\rangle$ is mapped to $\beta|\psi_i\rangle$ for one i, where i is picked with probability $\langle\psi_i|\psi_i\rangle$ and $\beta|\psi_i\rangle$ has unit length.

While one can in principle do measurements with respect to any orthogonal decomposition of the state space, in practice it normally suffices to do measurements with respect to one or more qubits in the computational basis. More precisely, we need operations

"measure with respect to qubits $i_1, \ldots, i_m$"

which means that the current state $|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha(x)|x\rangle$ is written as $|\psi\rangle = \sum_{u \in \{0,1\}^m} |\psi_u\rangle$ where $|\psi_u\rangle = \sum_{x: x_{i_1} \cdots x_{i_m} = u_1 \cdots u_m} \alpha(x)|x\rangle$. We pick one u according to the probability distribution $p(u) = \sum_{x: x_{i_1} \cdots x_{i_m} = u_1 \cdots u_m} \|\alpha(x)\|^2$, and project $|\psi\rangle$ to $\beta|\psi_u\rangle$ with $\beta = 1/\sqrt{p(u)}$ so that the resulting vector has again unit length.

We view a quantum algorithm as a combination of a classical sequential algorithm with quantum circuits. In its classical phases the algorithm may prepare any computational basis state $|x\rangle$ (where $x \in \{0,1\}^n$ for arbitrary n) and in its quantum phases a quantum circuit is applied. To summarize, a general model of quantum algorithms can be described as follows.

Classical and quantum part. A quantum algorithm is a combination of a classical algorithm with quantum circuits. The collection Ω of quantum gates that are used by the algorithm is fixed, but the quantum circuits used by the algorithm may work with any number n of qubits.

Quantum mode. In quantum mode, a quantum circuit on some number n of qubits is applied to a computational basis state $|x\rangle = |x_0 \ldots x_{n-1}\rangle$. As result the circuit produces a (usually entangled) state $|\psi\rangle \in H_{2^n}$.

Measurement. After application of a quantum circuit, the algorithm returns to classical mode (or another quantum circuit) by a measurement step in the computational basis of one or more qubits.

4 Simulating Quantum Algorithms by ASMs

An obvious, but minor, obstacle for the simulation of quantum algorithms by ASMs is that quantum algorithms are genuinely *probabilistic* algorithms, genuine in the sense that the probabilism is not simulated by a deterministic generation of pseudo-random numbers, but by the inherent probabilism in quantum mechanical processes. We thus have to assume that the simulating ASMs have access to true random numbers. Therefore, our quantum ASM will make use of a dynamic function **random** providing at each state access to a random real number in the interval $[0,1]$.

The power of quantum computing comes from the possibility to process in one step a large number of entangled classical states. To put it differently, quantum algorithms go beyond classical concepts of sequential algorithm.

Proposition 1. *Quantum algorithms are not sequential algorithms in the sense of postulates* **(S1) - (S3)** *and can thus not be simulated by sequential ASM.*

Proof. Both unitary evolution steps and measurement steps violate the Uniformly Bounded Exploration Postulate. □

On the other side, it is not difficult to simulate quantum algorithms by parallel ASMs in the sense of [1] (we will make this precise below). But this is not the point. The point of quantum computing is to use the inherent parallelism in quantum effects for constructing more efficient algorithms than classical sequential algorithms. Indeed quantum algorithm can perform certain tasks (like searching databases or factoring integers) more efficiently than classical sequential algorithms can do. But we do not want to add anything else than "being quantum" to the paradigm of sequential algorithms.

A quantum state is a superposition of classical states. A computation step of a quantum algorithm changes a quantum state by modifying in parallel the underlying classical states (the base vectors of the Hilbert space). Hence it is clear that everything a quantum algorithm can do can also be done by a classical parallel algorithm. But quantum computers (if they can be built) implement exponential parallelism without using exponential ressources!

From the quantum point of view, quantum algorithms are sequential, the only parallelism being inside the basic operations of unitary transformation and measurement. An adequate simulation of quantum algorithms by ASMs should take this into account. Rather than going up to arbitrary parallel ASMs, we therefore define quantum ASMs as sequential ASMs, enriched by two kinds of rules that take care of the quantum operations.

States. Quantum ASMs must have states that are sufficiently rich to describe the mathematics of quantum gates and quantum measurement. For circuits over a fixed basis Ω of quantum gates, the background of the states should contain, along with the requirements given in [1], the basic arithmetic operations on complex numbers, natural numbers, binary strings, appropriate descriptions for the quantum gates in Ω (appropriately parametrised if Ω is infinite), and the like. In particular we assume the presence of basic operations on multisets, and a summation operator mapping any finite multiset M of complex numbers to the sum over its elements. (Formally, if $M \subseteq \mathbb{C}$ is given by $\mathrm{Mult}_M : \mathbb{C} \to \mathbb{N}$, then $\sum M = \sum_{z \in \mathbb{C}} \mathrm{Mult}_M(z) \cdot z$.)

Suppose the algorithm enters a quantum phase on n qubits, starting with a classically prepared vector $|x\rangle = |x_0 \dots x_{n-1}\rangle$ in the computational basis of H_{2^n}, and should perform a sequence of unitary operations of the type described in Definition 2. The corresponding state X of the algorithm must contain the information n and $|x\rangle$. The entangled states $|\psi\rangle$ that will be produced by the unitary evolution of the system can be desribed by a dynamic function Ψ : $\{0,1\}^* \to \mathbb{C}$ where

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \Psi(x)|x\rangle.$$

Given n and x the ASM initialises Ψ by $\Psi(x) = 1$ and $\Psi(y) = 0$ for $y \neq x$.

Unitary transformations. To make precise how the operation

“apply U to the qubits $i_1, \dots, i_m$”

is described by an ASM rule observe that for every basis state $|x\rangle$ the relevant (i.e. non-zero) entries $U_{i_1 \dots i_m}(x, y)$ of the matrix $U_{i_1 \dots i_m}$ are those for which $x_k = y_k$ for all $k \notin \{i_1, \dots, i_m\}$. For $x \in \{0,1\}^n$, $i < n$ and $z \in \{0,1\}$, let $x[i/z]$ be the word obtained from x by replacing the i-th bit x_i by z. The application of $U_{i_1 \dots i_m}$ to the state defined by $\Psi : \{0,1\}^n \to \mathbb{C}$ is described by the rule

```
forall x ∈ {0,1}^n do
        Ψ(x) := ∑{{U(x_{i_1} ··· x_{i_m}, z_1 ··· z_m)Ψ(y) :
                    z_1,...,z_m ∈ {0,1}, y = x[i_1/z_1] ··· [i_m/z_m]}}
enddo
```

Measurement. Given an entangled state $|\psi\rangle \in H_{2^n}$ and indices $i_1, \ldots, i_m$, measurement with respect to qubits $i_1, \ldots, i_m$ means computing the probabilities $p(u) = \langle\psi_u|\psi_u\rangle$ of the associated decomposition $|\psi\rangle = \sum_{u\in\{0,1\}^m} |\psi_u\rangle$, selecting one u according to this distribution, and projecting $|\psi\rangle$ to $\beta|\psi_u\rangle$.

We can implement this by selecting a random number $r \in [0,1]$, decomposing the unit interval into segments $S(u)$ (where $u \in \{0,1\}^m$) of length $p(u)$ and picking the u for which r is contained in $S(u)$. Again, this can be described in ASM-notation by a straightforward application of a **forall-do** rule and summation over a multiset. We assume that `random` is a nullary function whose value in each state is a random real number between 0 and 1. (At each fixed state the value of `random` of course is the same when it is called several times in parallel.)

```
forall x ∈ {0,1}^n, u ∈ {0,1}^m do
        if φ(x,u) then  Ψ(x) := 1/√p(u) Ψ(x) else Ψ(x) := 0
enddo
```

where

$$
\begin{aligned}
\varphi(x,u) &:= (\ell(u) \leq \texttt{random} < \ell(u) + p(u)) \wedge x_{i_1} = u_1 \wedge \cdots \wedge x_{i_m} = u_m \\
\ell(u) &:= \sum \{\!\{ p(v) : v \in \{0,1\}^m : v <_{\mathrm{lex}} u \}\!\} \\
p(u) &:= \sum \{\!\{ \|\Psi(y)\|^2 : y \in \{0,1\}^n : y_{i_1} = u_1 \wedge \cdots \wedge y_{i_m} = u_m \}\!\}
\end{aligned}
$$

5 Postulates for Quantum Algorithms

Having given a description of one (albeit rather general) model of quantum algorithms by ASMs the question arises if also in the quantum case, the ASM thesis can be derived from some basic postulates that all models for quantum computation satisfy.

We describe such postulates here. As in the case of sequential and parallel algorithms, we need the Sequential Time Postulate (S1) and the Abstract State Postulate (S2). The Uniform Bounded Exploration Postulate (S3) will have to be modified and we will need postulates decribing unitary transformation steps and measurement steps. Concerning the probabilism inherent in quantum algorithms we follow the convention that probabilistic algorithms have a function `random` in their vocabulary that takes as values real numbers in the interval $[0,1]$. Update sets $\Delta(A,X)$ that do not depend on the value of `random` are called deterministic, otherwise they are probabilistic. The value of `random` is supposed to be changed uniformly, independently and externally (i.e. not by the algorithm) at any step. Another possibility would be to modify the sequential time postulate so that the next step transformation is of the form $\tau_A : \mathcal{S}(A) \times [0,1] \to \mathcal{S}(A)$.

We also need a variant of the Background Postulate, tailored for quantum algorithms, which gives some minimal requirements on what the states should contain, and which also imposes a closure condition.

Recall that a state of a computation should contain everything that is necessary to determine the future progress of that computation. Hence a state of a quantum algorithm does not merely consist of an element of an appropriate Hilbert space, but also of a classical state containing information on the unitary operations used, and of course also on the classical parts of the computation. Hence we essentially adapt the notion of state from [1, 3] by augmenting it with an element $|\psi\rangle$ of a Hilbert space. Sticking to the format of first-order structures, such an element is formally described by a function $\Psi : B \to \mathbb{C}$ from the basis B of the Hilbert space to the complex numbers. We use the notation $X = (X_c, |\psi\rangle)$ for the states of a quantum algorithm where X_c is the *classical part* of the state X (everything except Ψ).

Q1: Quantum State Postulate. Besides the conventions from [3], all states contain the three constants `classical`, `quantum`, and `measure` (which are interpreted by distinct elements) and a nullary function `mode` whose value is one of these three elements. Further all states contain a nullary function symbol `basis` whose value is a finite or countable set B, and a function symbol Ψ whose value is a function from B to the complex numbers $\mathbb{C}$, describing a unit length element $|\psi\rangle$ of the Hilbert space with orthonormal basis B. Further the state space $\mathcal{S}(A)$ of any quantum algorithm is closed under variations of $|\psi\rangle$, i.e., if $(X_c, |\psi\rangle) \in \mathcal{S}(A)$ then also $(X_c, |\varphi\rangle) \in \mathcal{S}(A)$ for every unit length $|\varphi\rangle$ of the same Hilbert space.

The function `mode` divides the state space into three classes, the classical states, the quantum states, and the measure states, each satisfying different postulates on the next step transformation.

The remaining postulates should ensure the following properties: For quantum states $(X_c, |\psi\rangle)$, the update of $|\psi\rangle$ is defined by a unitary transformation U (that depends only on X_c and A but not on $|\psi\rangle$). For measurement states $(X_c, |\psi\rangle)$ the update of $|\psi\rangle$ is defined by a (probabilistic) projection. Finally, the classical part X_c of any state $(X_c, |\psi\rangle)$ is updated by a sequential algorithm. The only unbounded parallelism of a quantum algorithm is the simultaneous update of all amplitudes in unitary transformation and measurement steps.

Remark. One might criticise this last requirement as too restricted: why not consider "quantum parallel algorithms"? The answer is that the very point of quantum algorithms is to use quantum effects in order to implement with polynomial ressources what classically requires exponential parallelism. Of course one could define computation models that combine quantum computing with unbounded classical parallelism, but this seems not interesting since we lose the advantage of quantum algorithms without going beyond the power of classical parallel algorithms.

It is relatively straightforward to formulate postulates for the first two properties.

Q2: Unitary Transformation Postulate. If $X = (X_c, |\psi\rangle)$ is a quantum state then $\tau_A(X)$ is either a quantum or a measure state, with the same value of `basis`. The change of $|\psi\rangle$ from X to $\tau_A(X)$ is described by a unitary operator U_X (which only depends on X_c).

Q3: Measurement Postulate: If $X = (X_c, |\psi\rangle)$ is a measure state then the change of $|\psi\rangle$ from X to $\tau_A(X)$ is decribed by a collection $\{M_{X,j} : j \in J\}$ of linear operators (which only depends on X_c) such that $\sum_{j \in J} M^*_{X,j} M_{X,j} = I$ and the index set J is linearly ordered.

The value of $|\psi\rangle$ after the measurement step is described by

$$\frac{M_{X,j}|\psi\rangle}{\sqrt{\langle\psi|M^*_{X,j} M_{X,j}|\psi\rangle}}$$

for one j, the probability for each j being $p(j) = \langle\psi|M^*_{X,j} M_{X,j}|\psi\rangle$.

Remark. When we say that a linear operator W defining the update of $|\psi\rangle$ in a state $X = (X_c, |\psi\rangle)$ depends only on X_c, we mean that the same operator W applies to every $X' = (X_c, |\psi'\rangle)$ with the same classical part X_c.

It is a little more delicate to formulate in a precise, yet natural way that the only parallelism is inside unitary transformation and measurement steps. In fact there are several possibilities.

We split the update sets of a quantum algorithm A on state $X = (X_c, |\psi\rangle)$ into the classical updates of the functions in X_c and the updates of $|\psi\rangle$, i.e., $\Delta(A, X) = \Delta(A, X_c) \cup \Delta(A, |\psi\rangle)$. A high-level formulation of the sequentiality postulate might be the following.

Q4: Sequentiality Postulate. For every quantum algorithm A, the classical update sets and all entries of the unitary transformation matrices and the projection matrices at quantum states and measure states are computed by sequential algorithms.

Here is a more explicit version of postulate Q4.

Q4': Sequentiality Postulate (detailed version). For every quantum algorithm A, there exist sequential algorithms A_1, A_2, A_3 with the following properties:

(1) A_1 has the same states as A and at every state $X = (X_c, |\psi\rangle)$ of A, the classical update set $\Delta(A, X_c)$ equals $\Delta(A_1, X)$, and in the case that X is a classical state, $\Delta(A_1, X)$ is the entire update set $\Delta(A, X)$,
(2) A_2 has states (X_c, x, y) consisting of the classical part of a quantum state X of A, and base vectors x, y in `basis`, and the set of outputs $out_{A_1}((X_c, x, y))$ contains exactly the entry $U_X(x, y)$ of the unitary transformation U_X.
(3) A_3 has states (X_c, x, y, j) consisting of the classical part of a measure state X of A, base vectors x, y in `basis`, and $j \in J$, and the set of outputs $out_{A_1}((X_c, x, y, j))$ contains exactly the entry $M_{X,j}(x, y)$ of the projection matrix $M_{X,j}$ at state X.

The same postulate can also be formulated in a different way, similar to the Uniformly Bounded Exploration Postulate for sequential algorithms.

Q4": Limited Exploration Postulate. Let $\Upsilon = \Upsilon_c \cup \{\Psi\}$ be the vocabulary of the quantum algorithm A. There exist finite sets T_1, T_2, T_3 of terms with the following properties.
(1) T_1 is a set of ground terms over vocabulary Υ and whenever two states X and X' coincide on T_1, then $\Delta(A, X_c) = \Delta(A, X_c')$, and if X and X' happen to be classical states, then even $\Delta(A, X) = \Delta(A, X')$.
(2) T_2 has vocabulary Υ_c and for all $t \in T_2$, free$(t) \subseteq \{x, y\}$. Whenever X and X' are two quantum states of A, a, b are elements of `basis` in X, and a', b' are elements of `basis` in X', such that (X_c, a, b) and (X_c', a', b') coincide on T_2, then the entries $U_X(a, b)$ and $U_{X'}(a', b')$ of the appropriate unitary transformations are equal.
(3) T_3 has vocabulary Υ_c and for all $t \in T_2$, free$(t) \subseteq \{x, y, j\}$. Whenever X and X' are two measure states of A, a, b are elements of `basis` in X, and a', b' are elements of `basis` in X', such that (X_c, a, b, k) and (X_c', a', b', k') coincide on T_3, then the entries $M_{X,k}(a, b)$ and $M_{X',k'}(a', b')$ of the appropriate projections are equal.

Example: Quantum Turing Machines. One of the popular models in the theory of quantum algorithms are quantum Turing machines (QTMs). For details, see e.g. [2, 5]. We verify that QTMs satisfy our postulates. A *quantum Turing machine (QTM)* is a tuple $M = (Q, \Sigma, q_0, q_a, q_r, \delta)$ with alphabet Σ, finite set of control states Q, initial state q_0, accepting state q_a, rejecting state q_r, and a transition function $\delta : Q \times \Sigma \times \Sigma \times Q \times \{-1, 0, 1\} \to \mathbb{C}$, where $\delta(q, \sigma, , q', \sigma', m)$ defines the amplitude that whenever the machine is in state q and reads symbol σ, it replaces σ with σ', enters state q' and moves the head in direction m. The transition function must satisfy the condition that the induced transformation on the configuration space is unitary.

As in the case of classical Turing machines, a configuration of a QTM is determined by the content of the tape, the current state, and the position of the head. Let C_M denote the set of all configurations of M and let H_M be the Hilbert space with orthonormal basis $\{|c\rangle : c \in C_M\}$. The transition function

δ induces a mapping $a : C_M \times C_M \to \mathbb{C}$ giving for every pair (c, c') of configurations the amplitude of the transition of M from the basis state $|c'\rangle$ to $|c\rangle$. The *quantum evolution* defined by M on H_M maps $|\psi\rangle = \sum_{c \in C_M} \Psi(c)|c\rangle$ to $U_M|\psi\rangle = \sum_{c \in C_M} \sum_{c' \in C_M} a(c, c')\Psi(c')|c\rangle$. The standard measurement is the measurement with respect to the basis $\{|c\rangle : c \in C_M\}$ (at the end of the computation).

Recall that the states of an algorithm are supposed to contain everything that is necessary to determine the future progress of a computation. For a QTM M this means that the states contain the static function `basis` whose value is the set of configurations C_M. Further the states contain functions `state`: $C_M \to Q$, `content`: $C_M \times \mathbb{Z} \to \Sigma$, `head`: $C_M \to \mathbb{Z}$ and a relation `replace`$\subseteq C_M \times \Sigma \times Q \times \{-1, 0, 1\} \times C_M$ describing the configurations and the cross-linking between the configurations (namely, `replace`(c, σ, q, z, c') holds iff c' results from c by replacing the contents of the `head`(c)-th cell of c with σ, the state of c with q and moving the head according to z).

The sequential algorithm computing for every state X the entries $U_X(c, c')$ of the associated unitary transformation returns $\delta(q_1, \sigma_1, \sigma_2, q_2, d)$ in the case that `replace`$(c', \sigma_2, q_2, d, c)$ holds where $\sigma_1 =$ `content`$(c', $`head`$(c'))$ and $q_1 =$ `state`(c'). Otherwise, it returns 0. Note that for each c there are only finitely many non-zero values $U(c, c')$. Concerning measurement steps in a quantum Turing machine, the set J in the postulate Q3 is C_M. The sequential algorithm demanded in postulate Q4', returns for a basis vector b the value 1 if it is in the subspace spanned by the basis vector v and 0 otherwise.

6 The Equivalence Theorem

The following lemma follows easily from the definition of sequential ASMs.

Lemma 1. *For every sequential ASM rule there exists an equivalent ASM rule of the form*

$$\begin{array}{l} \texttt{do-in-parallel} \\ \quad \texttt{if } \varphi_1 \texttt{ then } R_1 \\ \quad \texttt{if } \varphi_2 \texttt{ then } R_2 \\ \qquad \vdots \\ \quad \texttt{if } \varphi_n \texttt{ then } R_n \\ \texttt{enddo} \end{array}$$

where R_i is an update rule of the form $s_i := t_i$ or an output-rule `Output`(t_i).

Theorem 1. *Every quantum algorithm is equivalent to an ASM working with the same background.*

Proof. As already mentioned the states of a quantum algorithm can be partitioned into three classes: classical states, quantum states and measurement states.

Classical states. Postulate Q4' and the sequential ASM thesis imply that there exists an ASM A_c whose update set equals the classical update set of the quantum algorithm on any of its states. In the case of a classical state, the update set of A_c is the entire update set.

Quantum states. Postulate Q4' and the sequential ASM thesis for ASMs with output imply that there exists an ASM whose output for every quantum state X and every pair of base vectors $b, c \in \texttt{basis}$ contains exactly the entry $U_X(b, c)$. Further, there is an equivalent ASM with a program of the form given in Lemma 1 and with $\mathrm{free}(\varphi_i), \mathrm{free}(t_i) \subseteq \{b, c\}$. Since we are only interested in the output we can assume that all rules R_i, $i \in \{1, ..., n\}$ are output-rules of the form $\texttt{Output}(t_i)$. The following ASM A_q simulates the execution of the unitary transformation induced by the quantum algorithm in the case of a quantum state (w.l.o.g., we assume that $\bigvee_{i=1}^{n} \varphi_i$ is valid and for all $k, l \in \{1, ..., n\}, k \neq l$ the formula $\varphi_k \wedge \varphi_l$ is not satisfiable):

$$\begin{array}{l} \texttt{forall } b \texttt{ : } b \in \texttt{basis do} \\ \quad \Psi(b) := \sum_{i=1}^{n} \sum \{\!\{ t_i(b,c) \cdot \Psi(c) : c \in \texttt{basis} : \varphi_i(b,c) \}\!\} \\ \texttt{enddo} \end{array}$$

Measure States. With analogous arguments, we infer that there exists an ASM with a program of the form given in Lemma 1 whose output for every measure state X, every pair of base vectors $b, c \in B$ and every $j \in J$ contains exactly the entry $M_{X,j}(b, c)$. The rules R_i, $i \in \{1, ..., n\}$, are output-rules of the form $\texttt{Output}(t_i)$ and $\mathrm{free}(\varphi_i), \mathrm{free}(t_i) \subseteq \{b, c, j\}$. The following ASM A_m simulates the measurement induced by the quantum algorithm in the case of a measure state and an already chosen value for `random`.

$$\begin{array}{l} \texttt{forall } b, i \texttt{ : } b \in \texttt{basis} \wedge i \in J \texttt{ do} \\ \quad \texttt{if } \ell(i) \leq \texttt{random} < \ell(i) + p(i) \texttt{ then} \\ \quad\quad \Psi(b) := \frac{1}{\sqrt{n(i)}} \sum_{j=1}^{n} \sum \{\!\{ t_j(b,c) \cdot \Psi(c) : c \in \texttt{basis} : \varphi_j(b,c,i) \}\!\} \\ \texttt{enddo} \end{array}$$

where

$\ell(j)$ is the probability that the subspace belonging to a k with $k < j$ is measured (the set J is linearly ordered by postulate Q3).

$p(j)$ is the probability that the subspace belonging to j is measured.

$n(j)$ is the inverse of the square of the renormalization factor. It is obtained by summing the $\Psi_{\mathrm{new}}(b)^* \cdot \Psi_{\mathrm{new}}(b)$ for all base-vectors b where Ψ_{new} is the amplitude function after executing the transformation M_j and before carrying out normalization.

Hence $\ell(j), p(j), n(j)$ can be defined as follows:

$$\ell(j) = \sum \{\!\{ p(i) :: i \in J \wedge v < j \}\!\}$$

$$p(j) = \sum \{\!\{ \Psi(b)^* \cdot \Psi_{\mathrm{new}}(b) : b \in \texttt{basis} : \mathit{true} \}\!\}$$

$$n(j) = \sum \{\!\{ \Psi_{\mathrm{new}}(b)^* \cdot \Psi_{\mathrm{new}}(b) : b \in \texttt{basis} : \mathit{true} \}\!\}$$

with $\Psi_{\text{new}}(b) = \sum_{j=1}^{n} \sum \{\!\{ t_j(b,c) \cdot \Psi(c) : b \in \texttt{basis} : \varphi_j(b,c,j) \}\!\}$. The desired ASM combines A_c, A_q, and A_m as follows:

```
do-in-parallel
   A_c
   if mode = quantum then A_q
   if mode = measure then A_m
enddo
```

This completes the proof of the theorem. □

Remark. From the proof of Theorem 1, we see that it is not necessary to allow arbitrary parallelism in the ASMs that simulate quantum algorithms. The classical update sets are generated by a sequential ASM. Unbounded parallelism does appear in the simulation of the quantum steps and the measure steps, but in a very restricted way. We have only one forall-instruction (without nestings) and an update rule envolving a uniformly bounded number of summations over a multiset. Comprehension terms are not nested (the ASM given in the proof uses nested comprehension terms but it can be rewritten without nestings). Hence, the equivalence theorem could be formulated in a stronger way by allowing only ASMs satisfying the above constraints.

There are also many cases where multisets are not needed. One example are quantum Turing machines. In this case, every base vector (i.e. every configuration) is connected via a non-zero entry of the transoformation matrix to a uniformly bounded number of other base vectors (depending only on the algorithm, not on the input). Summations over the multisets for the quantum states can thus be replaced by sums over a uniformly bounded number of elements.

References

1. A. Blass and Y. Gurevich, *Abstract state machines capture parallel algorithms*, Tech. Rep. MSR-TR-2001-117, Microsoft Research, 2001.
2. J. Gruska, *Quantum Computing*, McGraw-Hill, 1999.
3. Y. Gurevich, *Sequential Abstract State Machines capture sequential algorithms*, ACM Transactions on Computational Logic, 1 (2000), pp. 77–111.
4. ———, *Logician in the land of OS: Abstract State Machines in Microsoft*, in Proc. 16th IEEE Symp. on Logic in Computer Science, 2001, pp. 129–136.
5. M. Hirvensalo, *Quantum Computing*, Springer, 2001.
6. M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.
7. P. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Journal on Computing, 26 (1997), pp. 1484–1509.
8. L. Vandersypen, M. Steffen, G. Breyta, C. Yannoni, M. Sherwood, and I. Chuang, *Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance*, Nature, 414 (2001), pp. 883–887.