Quantum (UREM) P Systems: Background, Definition and Computational Power

Alberto Leporati

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano – Bicocca
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
alberto.leporati@unimib.it

Summary. Quantum UREM P systems constitute an attempt to introduce in Membrane Computing notions and techniques deriving from quantum mechanics. As we will see, the approach we have adopted is different from what is usually done in Quantum Computing; in fact, we have been inspired by the functioning of some elementary operations that are used in quantum mechanics to exchange quanta of energy among quantum systems: creation and annihilation operators. In this paper we will provide the background which has led to the current definition of quantum UREM P systems, and we will recall some results concerning their computational power.

1 The Quest for Quantum P Systems

Membrane systems (also known as P systems) have been introduced by Gheorghe Păun in 1998 [27] as a new class of distributed and parallel computing devices, inspired by the structure and functioning of living cells. The basic model consists of a hierarchical structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes divide the Euclidean space into *regions*, that contain some *objects* (represented by symbols of an alphabet) and *evolution rules*. Using these rules, the objects may evolve and/or move from a region to a neighboring one. A *computation* starts from an initial configuration of the system and terminates when no evolution rule can be applied. Usually, the result of a computation is the multiset of objects contained into an *output membrane* or emitted from the skin of the system.

In what follows we assume the reader is already familiar with the basic notions and the terminology underlying P systems. For a layman–oriented introduction to P systems see [29], whereas for a systematic introduction we refer the reader to [28]. The latest information about P systems can be found in [32].

At the beginning of 2004, the Membrane Computing community started to query about the possibility to define a quantum version of P systems, and hence we started to work on the subject. A first paper [21] was presented in Palma de

Mallorca in November 2004. There, we proposed two options: either to follow the steps usually performed in Quantum Computing to define the quantum version of a given computation device, or to propose a completely new computation device which is based on the most elementary operation which can be conceived in physics: the exchange of a quantum of energy among two quantum systems. In the former case we would have obtained yet another quantum computation device whose computation steps are defined as the action of unitary operators, whose computations are logically reversible, and in which there are severe constraints on the amount of information which can be extracted from the system by measuring its state. In the latter case, instead, we felt that a new and interesting computation device could be introduced. Indeed, after a long and careful investigation, we decided to adopt creation and annihilation operators as the most elementary operations which can be performed by our computation device.

It was since 2001 that several authors introduced the notion of energy in P systems [1, 10, 31, 15, 22, 23]. Hence, we looked at the literature to find a model of P systems that was easily transformable in a quantum computation device. Our first choice, explored in [21], was to focus on energy–based P systems, in which a given amount of energy is associated to each object of the system. Moreover, instances of a special symbol e are used to denote free energy units occurring into the regions of the system. These energy units can be used to transform objects, using appropriate rules. The rules are defined according to conservativeness considerations. Indeed, in [21] we proposed two different versions of quantum P systems based on this classical model. Both versions were defined just like classical energy–based P systems, but for objects and rules. Objects were represented as pure states in the Hilbert space $\mathbb{C}^d$, $d \geq 2$, whereas the definition of rules differs between the two models. In the former, rules are defined as bijective functions — implemented as unitary operators — which transform the objects from the alphabet. In the latter, rules are defined as generic functions which map the alphabet into itself. Such functions are implemented using a generalization of the Conditional Quantum Control technique [3], and may yield to non unitary operators (a fact which is usually seen with suspect in traditional Quantum Computing).

However, several problems were pointed out in [21], the most serious being that it is difficult to avoid unwanted exchanges of energy among the objects, that yield the system to unintended states. Another difficulty was tied to the assignment of the amount of energy to every object of the system. In the original definition of energy–based P system, every object incorporated a different amount of energy; in other words, the amount of energy uniquely determined the kind of object and, by acquiring or releasing energy from the environment, one object was transformed to another kind of object. Under this definition, we were able in [22] to simulate a single Fredkin gate. However, in order to simulate an entire Fredkin circuit [23, 24] we were forced to relax the definition, and allow different kinds of objects to have the same amount of energy, otherwise the number of different kinds of objects would have become unmanageable. Last, but not the least, we have the problem of objects localization and control. How do we force an object to stay in a given

region for a long time, or to move to the desired region? Indeed, one notable feature of quantum systems is the so called "tunnel effect", thanks to which at every moment we have a positive probability that the object spontaneously leaves the current region. Of course this is a problem which is generally found when trying to control the behavior of a quantum system, but in the case of energy–based P systems the problem is exacerbated by the fact that the objects should move (only) as the effect of the application of a rule. This problem does not occur with quantum UREM P systems: there the objects interact by exchanging some amounts of energy, which are stored in quantum harmonic oscillators, but never move nor cross any membrane.

Looking for some alternatives, we considered the model of P systems introduced in [11], in which a non–negative integer value is assigned to each membrane. Such a value can be conveniently interpreted as the *energy* of the membrane. In these P systems, rules are assigned to the membranes rather than to the regions of the system. Every rule has the form $(in_i : \alpha, \Delta e, \beta)$ or $(out_i : \alpha, \Delta e, \beta)$, where i is the number of the membrane in a one-to-one labeling, α and β are symbols of the alphabet and Δe is a (possibly negative) integer number. The rule $(in_i : \alpha, \Delta e, \beta)$ is interpreted as follows: if a copy of α is in the region immediately surrounding membrane i, then this object crosses membrane i, is transformed to β, and modifies the energy of membrane i from the current value e_i to the new value $e_i + \Delta e$. Similarly, the rule $(out_i : \alpha, \Delta e, \beta)$ is interpreted as follows: if a copy of α is in the region surrounded by membrane i, then this object crosses membrane i, is transformed to β, and modifies the energy of membrane i from the current value e_i to the new value $e_i + \Delta e$. Both kinds of rules can be applied only if $e_i + \Delta e$ is non–negative. Since these rules transform one copy of an object to (one copy of) another object, in [11] they are referred to as *unit* rules. Hence, for conciseness, this model of P systems with unit rules and energy assigned to membranes is usually abbreviated as *UREM P systems*. An important observation is that in [11] the rules of UREM P systems are applied in a *sequential* way: at each computation step, *one* rule is selected from the pool of currently active rules, and it is applied. In [11] it has been proved that if we assign some local (that is, affecting only the membrane in which they are defined) priorities to the rules then UREM P systems are Turing complete, whereas if we omit the priorities then we do not get systems with universal computational power: indeed, we obtain a characterization of $PsMAT^{\lambda}$, the family of Parikh sets generated by context-free matrix grammars (without occurrence checking and with λ-rules).

So, finally, in [20] a *quantum* version of UREM P systems has been introduced, and it has been shown that such a model of computation is able to compute every partial recursive function (that is, it reaches the computational power of Turing machines) without the need to assign any priority between the rules of the system. In quantum UREM P systems, the rules $(in_i : \alpha, \Delta e, \beta)$ and $(out_i : \alpha, \Delta e, \beta)$ are realized through (not necessarily unitary) linear operators, which can be expressed as an appropriate composition of a truncated version of creation and annihilation operators. The operators which correspond to the rules have the form

$|\beta\rangle \langle\alpha| \otimes O$, where O is a linear operator which modifies the energy associated with the membrane (implemented as the state of a truncated quantum harmonic oscillator).

In [20] also Quantum Register Machines (QRMs, for short) have been introduced. It is our opinion that they could play the same role in proofs concerning the computational power of quantum computation devices as played by classical register machines for classical computing devices. Indeed, it has been shown in [20] that they are able to simulate any classical (deterministic) register machine, and hence they are (at least) Turing complete. The advantage of quantum UREM P systems over QRMs is that, due to the locality of interactions, the operators which correspond to the rules of the former are generally smaller than the operators corresponding to the instructions of the latter.

Finally, in [19] we have shown that, under the assumption that an external observer is able to discriminate a null vector from a non–null vector, the **NP**–complete problem 3-SAT can be solved using quantum (Fredkin) circuits, quantum register machines and quantum UREM P systems. Precisely, for each type of computation device we have proposed a *brute force* technique that exploits quantum parallelism (as well as the ability to alter quantum states by using creation and annihilation operators) to explore the whole space of assignments to the boolean variables of any given instance ϕ of 3-SAT, in order to determine whether at least one of such assignments satisfies ϕ. The solutions are presented in the so-called *semi–uniform* setting, which means that for every instance ϕ of 3-SAT a specific computation device (circuit, register machine or UREM P system) that solves it is built. Even if it is not formally proved, it is apparent that the proposed constructions can be performed in polynomial time by a classical deterministic Turing machine (whose output is a "reasonable" encoding of the machine, in the sense given in [16]).

In the rest of the paper we overview the basic notions of quantum mechanics which have led to the definition of quantum UREM P systems, and the results obtained so far about their computational power. Precisely, in section 2 we recall some basic notions on quantum computers, and we extend them to quantum systems which are able to assume a generic number $d \geq 2$ of base states. We also introduce some operators which can be used to operate on the states of such systems; for these operators, we first give a mathematical description and then we propose some possible physical interpretations. In sections 3 and 4 we give the precise definitions of both classical and quantum register machines and UREM P systems, respectively. In section 5 we prove that quantum UREM P systems are able to compute any partial recursive function, and hence they are (at least) as powerful as Turing machines. In section 6 we show how to build two families of quantum register machines and quantum UREM P systems, respectively, that solve (in the semi–uniform setting) the **NP**–complete decision problem 3-SAT. The conclusions, as well as some directions for future research, are given in section 7.

2 Quantum computers

From an abstract point of view, a quantum computer can be considered as made up of interacting parts. The elementary units (memory cells) that compose these parts are two–levels quantum systems called *qubits*. A qubit is typically implemented using the energy levels of a two–levels atom, or the two spin states of a spin–$\frac{1}{2}$ atomic nucleus, or a polarization photon. The mathematical description — independent of the practical realization — of a single qubit is based on the two–dimensional complex Hilbert space $\mathbb{C}^2$. The boolean truth values 0 and 1 are represented in this framework by the unit vectors of the canonical orthonormal basis, called the *computational basis* of $\mathbb{C}^2$:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad\qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Qubits are thus the quantum extension of the classical notion of bit, but whereas bits can only take two different values, 0 and 1, qubits are not confined to their two basis (also *pure*) states, $|0\rangle$ and $|1\rangle$, but can also exist in states which are coherent superpositions such as $\psi = c_0\,|0\rangle + c_1\,|1\rangle$, where c_0 and c_1 are complex numbers satisfying the condition $|c_0|^2 + |c_1|^2 = 1$. Performing a measurement of the state alters it. Indeed, performing a measurement on a qubit in the above superposition will return 0 with probability $|c_0|^2$ and 1 with probability $|c_1|^2$; the state of the qubit after the measurement (*post–measurement state*) will be $|0\rangle$ or $|1\rangle$, depending on the outcome.

A *quantum register* of size n (also called an *n–register*) is mathematically described by the Hilbert space $\otimes^n\mathbb{C}^2 = \underbrace{\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2}_{n \text{ times}}$, representing a set of n qubits labeled by the index $i \in \{1, \ldots, n\}$. An *n–configuration* (also *pattern*) is a vector $|x_1\rangle \otimes \ldots \otimes |x_n\rangle \in \otimes^n\mathbb{C}^2$, usually written as $|x_1, \ldots, x_n\rangle$, considered as a quantum realization of the boolean tuple $(x_1, \ldots, x_n)$. Let us recall that the dimension of $\otimes^n\mathbb{C}^2$ is 2^n and that $\{|x_1, \ldots, x_n\rangle : x_i \in \{0, 1\}\}$ is an orthonormal basis of this space called the *n*–register *computational basis*.

Computations are performed as follows. Each qubit of a given n–register is prepared in some particular pure state ($|0\rangle$ or $|1\rangle$) in order to realize the required n–configuration $|x_1, \ldots, x_n\rangle$, quantum realization of an input boolean tuple of length n. Then, a linear operator $G : \otimes^n\mathbb{C}^2 \to \otimes^n\mathbb{C}^2$ is applied to the n–register. The application of G has the effect of transforming the n–configuration $|x_1, \ldots, x_n\rangle$ into a new n–configuration $G(|x_1, \ldots, x_n\rangle) = |y_1, \ldots, y_n\rangle$, which is the quantum realization of the output tuple of the computer. We interpret such modification as a computation step performed by the quantum computer. The action of the operator G on a superposition $\Phi = \sum c^{i_1 \cdots i_n}\,|x_{i_1}, \ldots, x_{i_n}\rangle$, expressed as a linear combination of the elements of the n–register basis, is obtained by linearity: $G(\Phi) = \sum c^{i_1 \cdots i_n} G(|x_{i_1}, \ldots, x_{i_n}\rangle)$. We recall that linear operators which act on n–registers can be represented as order 2^n square matrices of complex entries. Usually (but not in this paper) such operators, as well as the corresponding matrices, are required to be unitary. In particular, this implies that the implemented

operations are logically reversible (an operation is *logically reversible* if its inputs can always be deduced from its outputs).

All these notions can be easily extended to quantum systems which have $d > 2$ pure states. In this setting, the d–valued versions of qubits are usually called *qudits* [17]. As it happens with qubits, a qudit is typically implemented using the energy levels of an atom or a nuclear spin. The mathematical description — independent of the practical realization — of a single qudit is based on the d–dimensional complex Hilbert space $\mathbb{C}^d$. In particular, the pure states $|0\rangle, \left|\frac{1}{d-1}\right\rangle, \left|\frac{2}{d-1}\right\rangle, \ldots, \left|\frac{d-2}{d-1}\right\rangle, |1\rangle$ are represented by the unit vectors of the canonical orthonormal basis, called the *computational basis* of $\mathbb{C}^d$:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad \left|\frac{1}{d-1}\right\rangle = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad \cdots, \quad \left|\frac{d-2}{d-1}\right\rangle = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

As before, a *quantum register* of size n can be defined as a collection of n qudits. It is mathematically described by the Hilbert space $\otimes^n \mathbb{C}^d$. An *n–configuration* is now a vector $|x_1\rangle \otimes \ldots \otimes |x_n\rangle \in \otimes^n \mathbb{C}^d$, simply written as $|x_1, \ldots, x_n\rangle$, for x_i running on $L_d = \left\{0, \frac{1}{d-1}, \frac{2}{d-1}, \ldots, \frac{d-2}{d-1}, 1\right\}$. An n–configuration can be viewed as the quantum realization of the "classical" tuple $(x_1, \ldots, x_n) \in L_d^n$. The dimension of $\otimes^n \mathbb{C}^d$ is d^n and the set $\{|x_1, \ldots, x_n\rangle : x_i \in L_d\}$ of all n–configurations is an orthonormal basis of this space, called the *n–register computational basis*. Notice that the set L_d can also be interpreted as a set of truth values, where 0 denotes falsity, 1 denotes truth and the other elements indicate different degrees of indefiniteness.

Let us now consider the set $\mathcal{E}_d = \left\{\varepsilon_0, \varepsilon_{\frac{1}{d-1}}, \varepsilon_{\frac{2}{d-1}}, \ldots, \varepsilon_{\frac{d-2}{d-1}}, \varepsilon_1\right\} \subseteq \mathbb{R}$ of real values; we can think to such quantities as energy values. To each element $v \in L_d$ we associate the energy level ε_v; moreover, let us assume that the values of $\mathcal{E}_d$ are all positive, equispaced, and ordered according to the corresponding objects: $0 < \varepsilon_0 < \varepsilon_{\frac{1}{d-1}} < \cdots < \varepsilon_{\frac{d-2}{d-1}} < \varepsilon_1$. If we denote by $\Delta\varepsilon$ the gap between two adjacent energy levels then the following linear relation holds:

$$\varepsilon_k = \varepsilon_0 + \Delta\varepsilon\,(d-1)\,k \qquad\qquad \forall\, k \in L_d \tag{1}$$

Notice that it is not required that $\varepsilon_0 = \Delta\varepsilon$. As explained in [21, 19], the values ε_k can be thought of as the energy eigenvalues of the infinite dimensional quantum harmonic oscillator truncated at the $(d-1)$-th excited level (see Figure 1), whose Hamiltonian on $\mathbb{C}^d$ is:

$$H = \begin{bmatrix} \varepsilon_0 & 0 & \ldots & 0 \\ 0 & \varepsilon_0 + \Delta\varepsilon & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \varepsilon_0 + (d-1)\Delta\varepsilon \end{bmatrix} \tag{2}$$

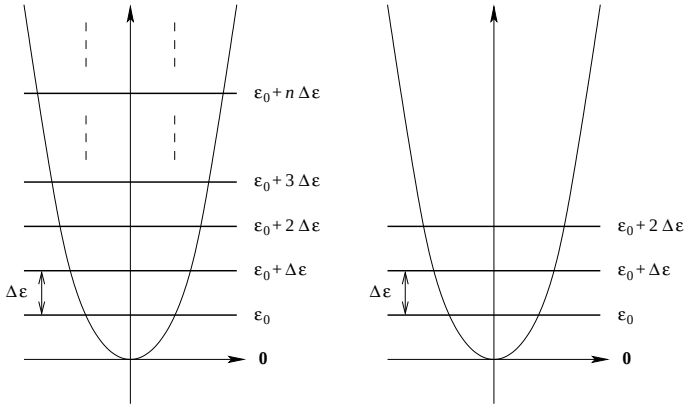

Fig. 1. Energy levels of the infinite dimensional (on the left) and of the truncated (on the right) quantum harmonic oscillator

The unit vector $|H = \varepsilon_k\rangle = \left|\frac{k}{d-1}\right\rangle$, for $k \in \{0, 1, \ldots, d-1\}$, is the eigenvector of the state of energy $\varepsilon_0 + k\Delta\varepsilon$. To modify the state of a qudit we can use creation and annihilation operators on the Hilbert space $\mathbb{C}^d$, which are defined respectively as:

$$a^\dagger = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & \sqrt{2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{d-1} & 0 \end{bmatrix} \qquad a = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \sqrt{2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sqrt{d-1} \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

It is easily verified that the action of $a^\dagger$ on the vectors of the canonical orthonormal basis of $\mathbb{C}^d$ is the following:

$$a^\dagger \left|\frac{k}{d-1}\right\rangle = \sqrt{k+1}\left|\frac{k+1}{d-1}\right\rangle \qquad \text{for } k \in \{0, 1, \ldots, d-2\}$$
$$a^\dagger |1\rangle = \mathbf{0}$$

whereas the action of a is:

$$a \left|\frac{k}{d-1}\right\rangle = \sqrt{k}\left|\frac{k-1}{d-1}\right\rangle \qquad \text{for } k \in \{1, 2, \ldots, d-1\}$$
$$a |0\rangle = \mathbf{0}$$

Using $a^\dagger$ and a we can also introduce the following operators:

$$N = a^\dagger a = \begin{bmatrix} 0\,0\,0\cdots & 0 \\ 0\,1\,0\cdots & 0 \\ 0\,0\,2\cdots & 0 \\ \vdots\,\vdots\,\vdots\,\ddots & \vdots \\ 0\,0\,0\cdots & d-1 \end{bmatrix} \qquad aa^\dagger = \begin{bmatrix} 1\,0\cdots & 0 & 0 \\ 0\,2\cdots & 0 & 0 \\ \vdots\,\vdots\,\ddots & \vdots & \vdots \\ 0\,0\cdots & d-1 & 0 \\ 0\,0\cdots & 0 & 0 \end{bmatrix}$$

The eigenvalues of the self–adjoint operator N are $0, 1, 2, \ldots, d-1$, and the eigenvector corresponding to the generic eigenvalue k is $|N = k\rangle = \left|\frac{k}{d-1}\right\rangle$. This corresponds to the notation adopted in [17], where the qudit base states are denoted by $|0\rangle, |1\rangle, \ldots, |d-1\rangle$, and it is assumed that a qudit can be in a superposition of the d base states:

$$c_0\,|0\rangle + c_1\,|1\rangle + \ldots + c_{d-1}\,|d-1\rangle$$

with $c_i \in \mathbb{C}$ for $i \in \{0, 1, \ldots, d-1\}$, and $|c_0|^2 + |c_1|^2 + \ldots + |c_{d-1}|^2 = 1$.

One possible physical interpretation of N is that it describes the *number of particles* of physical systems consisting of a maximum number of $d-1$ particles. In order to add a particle to the k particles state $|N = k\rangle$ (thus making it switch to the "next" state $|N = k+1\rangle$) we apply the creation operator $a^\dagger$, while to remove a particle from this system (thus making it switch to the "previous" state $|N = k-1\rangle$) we apply the annihilation operator a. Since the maximum number of particles that can be simultaneously in the system is $d-1$, the application of the creation operator to a full $d-1$ particles system does not have any effect on the system, and returns as a result the null vector. Analogously, the application of the annihilation operator to an empty particle system does not affect the system and returns the null vector as a result.

Another physical interpretation of operators $a^\dagger$ and a, by operator N, follows from the possibility of expressing the Hamiltonian (2) as follows:

$$H = \varepsilon_0\,\mathbb{I} + \Delta\varepsilon\,N = \varepsilon_0\,\mathbb{I} + \Delta\varepsilon\,a^\dagger a$$

In this case $a^\dagger$ (resp., a) realizes the transition from the eigenstate of energy $\varepsilon_k = \varepsilon_0 + k\,\Delta\varepsilon$ to the "next" (resp., "previous") eigenstate of energy $\varepsilon_{k+1} = \varepsilon_0 + (k+1)\,\Delta\varepsilon$ (resp., $\varepsilon_{k-1} = \varepsilon_0 + (k-1)\,\Delta\varepsilon$) for any $0 \leq k < d-1$ (resp., $0 < k \leq d-1$), while it collapses the last excited (resp., ground) state of energy $\varepsilon_0 + (d-1)\,\Delta\varepsilon$ (resp., ε_0) to the null vector.

The collection of all linear operators on $\mathbb{C}^d$ is a d^2–dimensional linear space whose canonical basis is:

$$\{E_{x,y} = |y\rangle\,\langle x| \;:\; x, y \in L_d\}$$

Since $E_{x,y}\,|x\rangle = |y\rangle$ and $E_{x,y}\,|z\rangle = \mathbf{0}$ for every $z \in L_d$ such that $z \neq x$, this operator transforms the unit vector $|x\rangle$ into the unit vector $|y\rangle$, collapsing all the other vectors of the canonical orthonormal basis of $\mathbb{C}^d$ to the null vector. Each of the operators $E_{x,y}$ can be expressed, using the whole algebraic structure of the associative algebra of operators, as a suitable composition of creation and annihilation operators, as follows:

$$E_{\frac{i}{d-1},\frac{j}{d-1}} = \begin{cases} \frac{\sqrt{j!}}{(d-1)!} A^{d-2,d-1-j,0}_{a^\dagger,a^\dagger} & \text{if } i = 0 \\ \frac{\sqrt{j!}}{(d-1)!} A^{d-1,d-1-j,0}_{a,a^\dagger} & \text{if } i = 1 \text{ and } j \geq 1 \\ \frac{\sqrt{i!}}{(d-1)!\sqrt{j!}} A^{d-2-i,d-1,j}_{a^\dagger,a^\dagger} & \text{if } (i = 1,\ j = 0 \text{ and } d \geq 3) \text{ or } (1 < i < d-2 \text{ and } j \leq i) \\ \frac{\sqrt{j!}}{(d-1)!\sqrt{i!}} A^{i-1,d-1,d-1-j}_{a,a} & \text{if } (i = d-2,\ j = d-1 \text{ and } d \geq 3) \text{ or } (1 < i < d-2 \text{ and } j > i) \\ \frac{1}{\sqrt{(d-1)!j!(d-1)}} A^{d-1,j,0}_{a^\dagger,a} & \text{if } i = d-2 \text{ and } j \leq d-2 \\ \frac{1}{\sqrt{(d-1)!j!}} A^{d-2,j,0}_{a,a} & \text{if } i = d-1 \end{cases}$$

Here we just recall, in order to keep the length of the paper under a reasonable size, that an alternative interpretation of qudits is possible, based on the values which can be assumed by the z component of the angular momentum of semi–integer spin quantum systems. Also with this interpretation every linear operator, and in particular operators $E_{x,y}$, can be realized as appropriate compositions of *spin–rising* (J_+) and *spin–lowering* (J_-) operators, similarly to what we have done with creation and annihilation operators. For the details, we refer the reader to [21, 19].

3 Classical and Quantum Register Machines

A (classical, deterministic) *n–register machine* is a construct $M = (n, P, l_0, l_h)$, where n is the number of registers, P is a finite set of instructions injectively labeled with a given set $lab(M)$, l_0 is the label of the first instruction to be executed, and l_h is the label of the last instruction of P. Registers contain non–negative integer values. Without loss of generality, we can assume $lab(M) = \{1, 2, \ldots, m\}$, $l_0 = 1$ and $l_h = m$. The instructions of P have the following forms:

- $j : (INC(r), k)$, with $j, k \in lab(M)$
  This instruction increments the value contained in register r, and then jumps to instruction k.
- $j : (DEC(r), k, l)$, with $j, k, l \in lab(M)$
  If the value contained in register r is positive then decrement it and jump to instruction k. If the value of r is zero then jump to instruction l (without altering the contents of the register).
- $m : Halt$
  Stop the machine. Note that this instruction can only be assigned to the final label m.

Register machines provide a simple universal computational model. Indeed, the results proved in [12] (based on the results established in [25]) as well as in [13] and [14] immediately lead to the following proposition.

Proposition 1. *For any partial recursive function $f : \mathbb{N}^\alpha \to \mathbb{N}^\beta$ there exists a deterministic $(\max\{\alpha, \beta\} + 2)$–register machine M computing f in such a way*

that, when starting with $(n_1, \ldots, n_\alpha) \in \mathbb{N}^\alpha$ *in registers* 1 *to* α*, M has computed* $f(n_1, \ldots, n_\alpha) = (r_1, \ldots, r_\beta)$ *if it halts in the final label* l_h *with registers* 1 *to* β *containing* r_1 *to* r_β*, and all other registers being empty; if the final label cannot be reached, then* $f(n_1, \ldots, n_\alpha)$ *remains undefined.*

A *quantum n–register machine* is defined exactly as in the classical case, as a four–tuple $M = (n, P, l_0, l_h)$. Each register of the machine can be associated to an infinite dimensional quantum harmonic oscillator capable to assume the base states $|\varepsilon_0\rangle, |\varepsilon_1\rangle, |\varepsilon_2\rangle, \ldots$, corresponding to its energy levels, as described in section 2. The program counter of the machine is instead realized through a quantum system capable to assume m different base states, from the set $\{|x\rangle : x \in L_m\}$. For simplicity, the instructions of P are denoted in the usual way:

$$j : (INC(i), k) \qquad \text{and} \qquad j : (DEC(i), k, l)$$

This time, however, these instructions are appropriate linear operators acting on the Hilbert space whose vectors describe the (global) state of M. Precisely, the instruction $j : (INC(r), k)$ is defined as the operator

$$O^{INC}_{j,r,k} = |p_k\rangle \langle p_j| \otimes \left(\otimes^{r-1}\mathbb{I}\right) \otimes a^\dagger \otimes \left(\otimes^{n-r}\mathbb{I}\right)$$

with $\mathbb{I}$ the identity operator on $\mathcal{H}$ (the Hilbert space in which the state vectors of the infinite dimensional quantum harmonic oscillators associated with the registers exist), whereas the instruction $j : (DEC(r), k, l)$ is defined as the operator

$$\begin{aligned} O^{DEC}_{j,r,k,l} = |p_l\rangle \langle p_j| \otimes \left(\otimes^{r-1}\mathbb{I}\right) \otimes |\varepsilon_0\rangle \langle \varepsilon_0| \otimes \left(\otimes^{n-r}\mathbb{I}\right) + \\ |p_k\rangle \langle p_j| \otimes \left(\otimes^{r-1}\mathbb{I}\right) \otimes a \otimes \left(\otimes^{n-r}\mathbb{I}\right) \end{aligned}$$

Hence the program P can be formally defined as the sum O_P of all these operators:

$$O_P = \sum_{j,r,k} O^{INC}_{j,r,k} + \sum_{j,r,k,l} O^{DEC}_{j,r,k,l}$$

Thus O_P is the global operator which describes a computation step of M. The Halt instruction is simply executed by doing nothing when the program counter assumes the value $|p_m\rangle$. For such a value, O_P would produce the null vector as a result; however, in what follows we will add a term to O_P that allows us to extract the solution of the problem from a prefixed register when the program counter assumes the value $|p_m\rangle$.

A *configuration* of M is given by the value of the program counter and the values contained in the registers. From a mathematical point of view, a configuration of M is a vector of the Hilbert space $\mathbb{C}^m \otimes (\otimes^n \mathcal{H})$. A transition between two configurations is obtained by executing one instruction of P (the one pointed at by the program counter), that is, by applying the operator O_P to the current configuration of M.

As shown in [20], QRMs can simulate any (classical, deterministic) register machine, and thus they are (at least) computationally complete.

4 Classical and Quantum UREM P Systems

We are now ready to focus our attention to P systems. As stated in the introduction, quantum UREM P systems have been introduced in [20] as a quantum version of UREM P systems. Hence, let us start by recalling the definition of the classical model of computation.

A UREM P system [11] of degree $d+1$ is a construct Π of the form:

$$\Pi = (A, \mu, e_0, \ldots, e_d, w_0, \ldots, w_d, R_0, \ldots, R_d)$$

where:

- A is an alphabet of *objects*;
- μ is a *membrane structure*, with the membranes labeled by numbers $0, \ldots, d$ in a one-to-one manner;
- $e_0, \ldots, e_d$ are the initial energy values assigned to the membranes $0, \ldots, d$. In what follows we assume that $e_0, \ldots, e_d$ are non–negative integers;
- $w_0, \ldots, w_d$ are multisets over A associated with the regions $0, \ldots, d$ of μ;
- $R_0, \ldots, R_d$ are finite sets of *unit rules* associated with the membranes $0, \ldots, d$. Each rule has the form $(\alpha : a, \Delta e, b)$, where $\alpha \in \{in, out\}$, $a, b \in A$, and $|\Delta e|$ is the amount of energy that — for $\Delta e \geq 0$ — is added to or — for $\Delta e < 0$ — is subtracted from e_i (the energy assigned to membrane i) by the application of the rule.

By writing $(\alpha_i : a, \Delta e, b)$ instead of $(\alpha : a, \Delta e, b) \in R_i$, we can specify only one set of rules R with

$$R = \{(\alpha_i : a, \Delta e, b) \ : \ (\alpha : a, \Delta e, b) \in R_i, \ 0 \leq i \leq d\}$$

The *initial configuration* of Π consists of $e_0, \ldots, e_d$ and $w_0, \ldots, w_d$. The transition from a configuration to another one is performed by non–deterministically choosing one rule from some R_i and applying it (observe that here we consider a *sequential* model of applying the rules instead of choosing rules in a maximally parallel way, as it is often required in P systems). Applying $(in_i : a, \Delta e, b)$ means that an object a (being in the membrane immediately outside of i) is changed into b while entering membrane i, thereby changing the energy value e_i of membrane i by Δe. On the other hand, the application of a rule $(out_i : a, \Delta e, b)$ changes object a into b while leaving membrane i, and changes the energy value e_i by Δe. The rules can be applied only if the amount e_i of energy assigned to membrane i fulfills the requirement $e_i + \Delta e \geq 0$. Moreover, we use some sort of local priorities: if there are two or more applicable rules in membrane i, then one of the rules with $\max |\Delta e|$ has to be used.

A sequence of transitions is called a *computation*; it is *successful* if and only if it halts. The *result* of a successful computation is considered to be the distribution of energies among the membranes (a non–halting computation does not produce a result). If we consider the energy distribution of the membrane structure as

the input to be analysed, we obtain a model for accepting sets of (vectors of) non–negative integers.

The following result, proved in [11], establishes computational completeness for this model of P systems.

Proposition 2. *Every partial recursive function* $f : \mathbb{N}^\alpha \to \mathbb{N}^\beta$ *can be computed by a UREM P system with (at most)* $\max\{\alpha, \beta\} + 3$ *membranes.*

It is interesting to note that the proof of this proposition is obtained by simulating register machines. In the simulation, a P system is defined which contains one subsystem for each register of the simulated machine. The contents of the register are expressed as the energy value e_i assigned to the i-th subsystem. A single object is present in the system at every computation step, which stores the label of the instruction of the program P currently simulated. Increment instructions are simulated in two steps by using the rules $(in_i : p_j, 1, \widetilde{p_j})$ and $(out_i : \widetilde{p_j}, 0, p_k)$. Decrement instructions are also simulated in two steps, by using the rules $(in_i : p_j, 0, \widetilde{p_j})$ and $(out_i : \widetilde{p_j}, -1, p_k)$ or $(out_i : \widetilde{p_j}, 0, p_l)$. The use of priorities associated to these last rules is crucial to correctly simulate a decrement instruction. For the details of the proof we refer the reader to [11].

On the other hand, by omitting the priority feature we do not get systems with universal computational power. Precisely, in [11] it is proved that P systems with unit rules and energy assigned to membranes without priorities and with an arbitrary number of membranes characterize the family $PsMAT^\lambda$ of Parikh sets generated by context–free matrix grammars (without occurrence checking and with λ-rules).

In *quantum* UREM P systems, all the elements of the model (multisets, the membrane hierarchy, configurations, and computations) are defined just like the corresponding elements of the classical P systems, but for objects and rules. The objects of A are represented as pure states of a quantum system. If the alphabet contains $d \geq 2$ elements then, recalling the notation introduced in section 2, without loss of generality we can put $A = \left\{ |0\rangle, \left|\frac{1}{d-1}\right\rangle, \left|\frac{2}{d-1}\right\rangle, \ldots, \left|\frac{d-2}{d-1}\right\rangle, |1\rangle \right\}$, that is, $A = \{|a\rangle \ : \ a \in L_d\}$. As stated above, the quantum system will also be able to assume as a state any superposition of the kind:

$$c_0 |0\rangle + c_{\frac{1}{d-1}} \left|\frac{1}{d-1}\right\rangle + \ldots + c_{\frac{d-2}{d-1}} \left|\frac{d-2}{d-1}\right\rangle + c_1 |1\rangle$$

with $c_0, c_{\frac{1}{d-1}}, \ldots, c_{\frac{d-2}{d-1}}, c_1 \in \mathbb{C}$ such that $\sum_{i=0}^{d-1} \left| c_{\frac{i}{d-1}} \right|^2 = 1$. A multiset is simply a collection of quantum systems, each in its own state.

In order to represent the energy values assigned to membranes we must use quantum systems which can exist in an infinite (countable) number of states. Hence we assume that every membrane of the quantum P system has an associated infinite dimensional quantum harmonic oscillator whose state represents the energy value assigned to the membrane. To modify the state of such harmonic oscillator we can use the infinite dimensional version of creation ($a^\dagger$) and annihilation (a)

operators[1] described in section 2, which are commonly used in quantum mechanics. The actions of $a^\dagger$ and a on the state of an infinite dimensional harmonic oscillator are analogous to the actions on the states of truncated harmonic oscillators; the only difference is that in the former case there is no state with maximum energy, and hence the creation operator never produces the null vector. Also in this case it is possible to express operators $E_{x,y} = |y\rangle \langle x|$ as appropriate compositions of $a^\dagger$ and a.

As in the classical case, rules are associated to the membranes rather than to the regions enclosed by them. Each rule of R_i is an operator of the form

$$|y\rangle \langle x| \otimes O, \qquad \text{with } x, y \in L_d \tag{3}$$

where O is a linear operator which can be expressed by an appropriate composition of operators $a^\dagger$ and a. The part $|y\rangle \langle x|$ is the *guard* of the rule: it makes the rule "active" (that is, the rule produces an effect) if and only if a quantum system in the basis state $|x\rangle$ is present. The semantics of rule (3) is the following: If an object in state $|x\rangle$ is present in the region immediately outside membrane i, then the state of the object is changed to $|y\rangle$ and the operator O is applied to the state of the harmonic oscillator associated with the membrane. Notice that the application of O can result in the null vector, so that the rule has no effect even if its guard is satisfied; this fact is equivalent to the condition $e_i + \Delta e \geq 0$ on the energy of membrane i required in the classical case. Differently from the classical case, no local priorities are assigned to the rules. If two or more rules are associated to membrane i, then they are summed. This means that, indeed, we can think to each membrane as having only one rule with many guards. When an object is present, the inactive parts of the rule (those for which the guard is not satisfied) produce the null vector as a result. If the region in which the object occurs contains two or more membranes, then all their rules are applied to the object. Observe that the object which activates the rules never crosses the membranes. This means that the objects specified in the initial configuration can change their state but never move to a different region. Notwithstanding, transmission of information between different membranes is possible, since different objects may modify in different ways the energy state of the harmonic oscillators associated with the membranes.

The application of one or more rules determines a *transition* between two configurations. A *halting configuration* is a configuration in which no rule can be applied. A sequence of transitions is a *computation*. A computation is *successful* if and only if it *halts*, that is, reaches a halting configuration. The *result* of a successful computation is considered to be the distribution of energies among the membranes in the halting configuration. A non–halting computation does not produce a result. Just like in the classical case, if we consider the energy distribution of the membrane structure as the input to be analyzed, we obtain a model for accepting sets of (vectors of) non–negative integers.

[1] We recall that an alternative formulation that uses spin–rising (J_+) and spin–lowering (J_-) operators instead of creation and annihilation is also possible.

5 Computational Completeness of Quantum UREM P Systems

In this section we prove that quantum P systems with unit rules and energy assigned to membranes are computationally complete, that is, they are able to compute any partial recursive function $f : \mathbf{N}^\alpha \to \mathbf{N}^\beta$. As in the classical case, the proof is obtained by simulating register machines.

Theorem 1. *Every partial recursive function $f : \mathbf{N}^\alpha \to \mathbf{N}^\beta$ can be computed by a quantum P system with unit rules and energy assigned to membranes with (at most)* $\max\{\alpha, \beta\} + 3$ *membranes.*

Proof. Let $M = (n, P, 1, m)$ be a deterministic n–register machine that computes f. Let m be the number of instructions of P. The initial instruction of P has the label 1, and the halting instruction has the label m. Observe that, according to Proposition 1, $n = \max\{\alpha, \beta\} + 2$ is enough.

The input values $x_1, \ldots, x_\alpha$ are expected to be in the first α registers, and the output values are expected to be in registers 1 to β at the end of a successful computation. Moreover, without loss of generality, we may assume that at the beginning of a computation all the registers except (eventually) the registers 1 to α contain zero.

We construct the quantum P system

$$\Pi = (A, \mu, e_0, \ldots, e_n, w_0, \ldots, w_n, R_0, \ldots, R_n)$$

where:

- $A = \{|j\rangle \mid j \in L_m\}$
- $\mu = [_0[_1]_1 \cdots [_\alpha]_\alpha \cdots [_n]_n]_0$
- $e_i = \begin{cases} |\varepsilon_{x_i}\rangle & \text{for } 1 \le i \le \alpha \\ |\varepsilon_0\rangle & \text{for } \alpha + 1 \le i \le n \\ \mathbf{0} & \text{(the null vector) for } i = 0 \end{cases}$
- $w_0 = |0\rangle$
- $w_i = \emptyset \qquad \text{for } 1 \le i \le n$
- $R_0 = \emptyset$
- $R_i = \sum_{j=1}^{m} O_{i_j} \qquad \text{for } 1 \le i \le n$
  where the O_{i_j}'s are local operators which simulate instructions of the kind $j : (INC(i), k)$ and $j : (DEC(i), k, l)$ (one local operator for each increment or decrement operation which affects register i). The details on how the O_{i_j}'s are defined are given below.

The value contained into register i, $1 \le i \le n$, is represented by the energy value $e_i = |\varepsilon_{x_i}\rangle$ of the infinite dimensional quantum harmonic oscillator associated with membrane i. Figure 2 depicts a typical configuration of Π. The skin contains one object of the kind $|j\rangle$, $j \in L_m$, which mimics the program counter of machine M. Precisely, if the program counter of M has the value $k \in \{1, 2, \ldots, m\}$ then

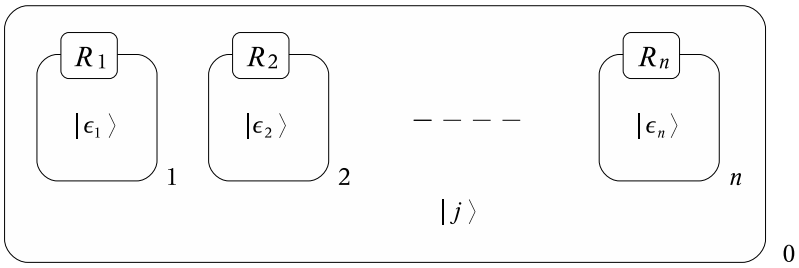

Fig. 2. A configuration of the simulating P system

the object present in region 0 is $\left|\frac{k-1}{m-1}\right\rangle$. In order to avoid cumbersome notation, in what follows we denote by $|p_k\rangle$ the state $\left|\frac{k-1}{m-1}\right\rangle$ of the quantum system which mimics the program counter.

The sets of rules R_i depend upon the instructions of P. Precisely, the simulation works as follows.

1. Increment instructions $j : (INC(i), k)$ are simulated by a guarded rule of the kind $|p_k\rangle \langle p_j| \otimes a^\dagger \in R_i$.
   If the object $|p_j\rangle$ is present in region 0, then the rule transforms it into object $|p_k\rangle$ and increments the energy level of the harmonic oscillator contained into membrane i.
2. Decrement instructions $j : (DEC(i), k, l)$ are simulated by a guarded rule of the kind:
   $$|p_l\rangle \langle p_j| \otimes |\varepsilon_0\rangle \langle \varepsilon_0| + |p_k\rangle \langle p_j| \otimes a \in R_i$$
   In fact, let us assume that the object $|p_j\rangle$ is present in region 0 (if $|p_j\rangle$ is not present then the above rule produces the null operator), and let us denote by O the above rule. The harmonic oscillator may be in the base state $|\varepsilon_0\rangle$ or in a base state $|\varepsilon_x\rangle$ with x a positive integer.
   If the state of the harmonic oscillator is $|\varepsilon_0\rangle$ then the rule produces:
   $$\begin{aligned} &O(|p_j\rangle \otimes |\varepsilon_0\rangle) = \\ &\quad = (|p_l\rangle \langle p_j| \otimes |\varepsilon_0\rangle \langle \varepsilon_0|)(|p_j\rangle \otimes |\varepsilon_0\rangle) + (|p_k\rangle \langle p_j| \otimes a)(|p_j\rangle \otimes |\varepsilon_0\rangle) = \\ &\quad = |p_l\rangle \otimes |\varepsilon_0\rangle + |p_k\rangle \otimes \mathbf{0} = |p_l\rangle \otimes |\varepsilon_0\rangle \end{aligned}$$
   that is, the state of the oscillator is unaltered and the program counter is set to $|p_l\rangle$.
   If the state of the harmonic oscillator is $|\varepsilon_x\rangle$, for a positive integer x, then the rule produces:
   $$\begin{aligned} &O(|p_j\rangle \otimes |\varepsilon_x\rangle) = \\ &\quad = (|p_l\rangle \langle p_j| \otimes |\varepsilon_0\rangle \langle \varepsilon_0|)(|p_j\rangle \otimes |\varepsilon_x\rangle) + (|p_k\rangle \langle p_j| \otimes a)(|p_j\rangle \otimes |\varepsilon_x\rangle) = \\ &\quad = |p_l\rangle \otimes \mathbf{0} + |p_k\rangle \otimes a\,|\varepsilon_x\rangle = |p_k\rangle \otimes |\varepsilon_{x-1}\rangle \end{aligned}$$

that is, the energy level of the harmonic oscillator is decremented and the program counter is set to $|p_k\rangle$.

The set R_i of rules is obtained by summing all the operators which affect (increment or decrement) register i. The Halt instruction is simply simulated by doing nothing with the object $|p_m\rangle$ when it appears in region 0.

It is apparent from the description given above that after the simulation of each instruction each energy value e_i equals the value contained into register i, with $1 \leq i \leq m$. Hence, when the halting symbol $|p_m\rangle$ appears in region 0, the energy values $e_1, \ldots, e_\beta$ equal the output of the program P. ∎

Let us conclude this section by observing that, in order to obtain computational completeness, it is not necessary that the objects cross the membranes. This fact avoids one of the problems raised in [21]: the existence of a "magic" quantum transportation mechanism which is able to move objects according to the target contained into the rule. In quantum P systems with unit rules and energy assigned to membranes, the only problem is to keep the object $|p_j\rangle$ localized in region 0, so that it never enters into the other regions. In other words, the major problem of this kind of quantum P systems is to oppose the tunnel effect.

It should also be evident that the proof of Theorem 1 can be modified to show that quantum P systems are able to simulate quantum register machines. Indeed, the notable difference between the quantum P systems described above and quantum register machines is that in the latter model we modify the values contained into registers using *global* operators (if a given register must not be modified then the identity operator is applied to its state) whereas in the former model we operate locally, on a smaller Hilbert space. Hence, as it happens in classical P systems, membranes are used to divide the site where the computation occurs into independent local areas. The effect of each rule is local, in the sense that the rule affects only the state of one subsystem. Due to the simulations mentioned above, we can order these computational models with respect to their computational power, as follows:

deterministic register machines	$\leq$	quantum register machines	$\leq$	quantum P systems with unit rules and energy assigned to membranes

Quantum register machines can thus be used as a tool to study the computational power of other quantum models of computation, just like it happens in the classical case.

6 Solving 3-SAT with QRMs and with Quantum UREM P Systems

Quantum UREM P systems are not only able to compute all partial recursive functions, like Turing machines, but they can also be very efficient computation

devices. Indeed, in this section we show how we can solve in polynomial time the **NP**–complete decision problem 3-SAT by quantum register machines and by quantum UREM P systems. As we will see, the solution provided by quantum UREM P systems will be even more efficient that the one obtained with QRMs.

It is important to stress that our solutions assume that a specific non–unitary operator, built using the truncated version of creation and annihilation operators, can be realized as an instruction of quantum register machines and as a rule of quantum UREM P systems, respectively. The construction relies also upon the assumption that an external observer is able to discriminate, as the result of a measurement, a null vector from a non–null vector.

6.1 The 3-SAT problem

A *boolean* variable is a variable which can assume one of two possible truth values: TRUE and FALSE. As usually done in the literature, we will denote TRUE by 1 and FALSE by 0. A *literal* is either a directed or a negated boolean variable. A *clause* is a disjunction of literals, whereas a 3-*clause* is a disjunction of exactly three literals. Given a set $X = \{x_1, x_2, \ldots, x_n\}$ of boolean variables, an *assignment* is a mapping $a : X \to \{0, 1\}$ that associates to each variable a truth value. The number of all possible assignments to the variables of X is 2^n. We say that an assignment *satisfies* the clause C if, assigned the truth values to all the variables which occur in C, the evaluation of C (considered as a boolean formula) gives 1 as a result.

The 3-SAT decision problem is defined as follows.

Problem 1. NAME: 3-SAT.

- INSTANCE: a set $C = \{C_1, C_2, \ldots, C_m\}$ of 3-clauses, built on a finite set $\{x_1, x_2, \ldots, x_n\}$ of boolean variables.
- QUESTION: is there an assignment of the variables $x_1, x_2, \ldots, x_n$ that satisfies all the clauses in C?

Notice that the number m of possible 3-clauses is polynomially bounded with respect to n: in fact, since each clause contains exactly three literals, we can have at most $(2n)^3 = 8n^3$ clauses.

In what follows we will equivalently say that an instance of 3-SAT is a boolean formula ϕ_n, built on n free variables and expressed in conjunctive normal form, with each clause containing exactly three literals. The formula ϕ_n is thus the conjunction of the above clauses.

It is well known [16] that 3-SAT is an **NP**–complete problem.

6.2 Solving 3-SAT with quantum register machines

Let ϕ_n be an instance of 3-SAT containing n free variables. We will first show how to evaluate ϕ_n with a classical register machine; then, we will initialize the input registers with a superposition of all possible assignments, we will compute the corresponding superposition of output values into an output register, and finally

we will apply the linear operator $2^n |1\rangle \langle 1|$ to the output register to check whether ϕ_n is a positive instance of 3-SAT.

The register machine that we use to evaluate ϕ_n is composed by $n+1$ registers. The first n registers correspond (in a one-to-one manner) to the free variables of ϕ_n, while the last register is used to compute the output value. The *structure* of the program used to evaluate ϕ_n is the following:

$$\begin{array}{rl} & \phi = 0 \\ & \textbf{if } C_1 = 0 \textbf{ then goto } \text{end} \\ & \textbf{if } C_2 = 0 \textbf{ then goto } \text{end} \\ & \vdots \\ & \textbf{if } C_m = 0 \textbf{ then goto } \text{end} \\ & \phi = 1 \\ \text{end:} & \end{array}$$

where ϕ denotes the output register, and $C_1, C_2, \ldots, C_m$ are the clauses of ϕ_n. Let $X_{i,j}$, with $j \in \{1, 2, 3\}$, be the literals (directed or negated variables) which occur in the clause C_i (hence $C_i = X_{i,1} \vee X_{i,2} \vee X_{i,3}$). We can thus write the above structure of the program, at a finer grain, as follows:

$$\begin{array}{rl} & \phi = 0 \\ & \textbf{if } X_{1,1} = 1 \textbf{ then goto } \text{end}_1 \\ & \textbf{if } X_{1,2} = 1 \textbf{ then goto } \text{end}_1 \\ & \textbf{if } X_{1,3} = 1 \textbf{ then goto } \text{end}_1 \\ & \textbf{goto } \text{end} \\ \text{end}_1\text{:} & \textbf{if } X_{2,1} = 1 \textbf{ then goto } \text{end}_2 \\ & \textbf{if } X_{2,2} = 1 \textbf{ then goto } \text{end}_2 \\ & \\ & \textbf{if } X_{2,3} = 1 \textbf{ then goto } \text{end}_2 \\ & \textbf{goto } \text{end} \\ \text{end}_2\text{:} & \cdots\cdots \\ & \vdots \\ \text{end}_{m-1}\text{:} & \textbf{if } X_{m,1} = 1 \textbf{ then goto } \text{end} \\ & \textbf{if } X_{m,2} = 1 \textbf{ then goto } \text{end} \\ & \textbf{if } X_{m,3} = 1 \textbf{ then goto } \text{end} \\ & \phi = 1 \\ \text{end:} & \end{array} \tag{4}$$

In the above structure it is assumed that each literal $X_{i,j}$, with $1 \leq i \leq m$ and $j \in \{1, 2, 3\}$, is substituted with the corresponding variable which occurs in it; moreover, if the variable occurs negated into the literal then the comparison must be done with 0 instead of 1:

$$\textbf{if } X_{i,j} = 0 \textbf{ then goto } \text{end}_i$$

Since the free variables of ϕ_n are bijectively associated with the first n registers of the machine, in order to evaluate ϕ_n we need a method to check whether a given register contains 0 (or 1) without destroying its value. Let us assume that, when the program counter of the machine reaches the value k, we have to execute the following instruction:

$$k\text{: } \textbf{if } X_{i,j} = 1 \textbf{ then goto } \text{end}_i$$

We translate such instruction as follows (where, instead of $X_{i,j}$, we specify the register which corresponds to the variable indicated in $X_{i,j}$):

$$\begin{aligned} k\text{: } & DEC(X_{i,j}), k+1, k+2 \\ k+1\text{: } & INC(X_{i,j}), \text{end}_i \end{aligned}$$

The instruction:

$$k\text{: } \textbf{if } X_{i,j} = 0 \textbf{ then goto } \text{end}_i$$

is instead translated as follows:

$$\begin{aligned} k\text{: } & DEC(X_{i,j}), k+1, \text{end}_i \\ k+1\text{: } & INC(X_{i,j}), k+2 \end{aligned}$$

Notice that the only difference with the previous sequence of instructions is in the position of "end_i" and "$k+2$". Moreover, the structure of the program is always the same. As a consequence, given an instance ϕ_n of 3-SAT, the program P of a register machine which evaluates ϕ_n can be obtained in a very straightforward (mechanical) way.

On a *classical* register machine, this program computes the value of ϕ_n for a given assignment to its variables $x_1, x_2, \ldots, x_n$. On a *quantum* register machine we can initialize the registers with the following state:

$$\otimes^{n-1} H_1 \left|0\right\rangle \otimes \left|0\right\rangle$$

which sets the output register ϕ to 0 and the registers corresponding to $x_1, x_2, \ldots, x_n$ to a superposition of all possible assignments. Then, we apply the global operator O_P which corresponds to the program P until the program counter reaches the value $|p_{\text{end}}\rangle$, thus computing in the output register a superposition of all classical results. The operator O_P is built as described in section 3, with the only difference that now it contains also the term:

$$\begin{aligned} &\left|p_{\text{end}}\right\rangle \left\langle p_{\text{end}}\right| \otimes \text{ID}_n \otimes 2^n \left|1\right\rangle \left\langle 1\right| = \\ &\qquad \left|p_{\text{end}}\right\rangle \left\langle p_{\text{end}}\right| \otimes \text{ID}_n \otimes \Big[\underbrace{(\left|1\right\rangle \left\langle 1\right| + \left|1\right\rangle \left\langle 1\right|) \circ \ldots \circ (\left|1\right\rangle \left\langle 1\right| + \left|1\right\rangle \left\langle 1\right|)}_{n \text{ times}} \Big] \end{aligned}$$

which extracts the result from the output register when the program counter assumes the value $|p_{\text{end}}\rangle$. The number of times we have to apply O_P is equal to the

length of P, that is, $2 \cdot 3m + 2 = 6m + 2$: two instructions for each literal in every clause, plus two final instructions.

Now, if ϕ_n is not satisfiable then the contents of the output register is $|0\rangle$, and when the program counter reaches the value $|p_{\text{end}}\rangle$ the operator O_P transforms it to the null vector. On the other hand, if ϕ_n is satisfiable then the contents of the output register will be a superposition $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, with $\alpha_1 \neq 0$. By applying the operator O_P we obtain (here $|\psi_n\rangle$ denotes the state of the n input registers):

$$\begin{aligned}
&O_P\big(|p_{\text{end}}\rangle \otimes |\psi_n\rangle \otimes (\alpha_0 |0\rangle + \alpha_1 |1\rangle)\big) = \\
&\quad = \big(|p_{\text{end}}\rangle \langle p_{\text{end}}| \otimes \text{ID}_n \otimes 2^n |1\rangle \langle 1| \big) \cdot \\
&\qquad\qquad \cdot \big(|p_{\text{end}}\rangle \otimes |\psi_n\rangle \otimes (\alpha_0 |0\rangle + \alpha_1 |1\rangle)\big) = \\
&\quad = |p_{\text{end}}\rangle \langle p_{\text{end}}|p_{\text{end}}\rangle \otimes \text{ID}_n |\psi_n\rangle \otimes 2^n |1\rangle \langle 1| (\alpha_0 |0\rangle + \alpha_1 |1\rangle) = \\
&\quad = |p_{\text{end}}\rangle \otimes |\psi_n\rangle \otimes (2^n \alpha_0 |1\rangle \langle 1|0\rangle + 2^n \alpha_1 |1\rangle \langle 1|1\rangle) = \\
&\quad = |p_{\text{end}}\rangle \otimes |\psi_n\rangle \otimes (\mathbf{0} + 2^n \alpha_1 |1\rangle) = \\
&\quad = |p_{\text{end}}\rangle \otimes |\psi_n\rangle \otimes 2^n \alpha_1 |1\rangle
\end{aligned}$$

that is, a non–null vector.

We can thus conclude that if an external observer is able to discriminate between a null vector and a non–null vector, and it is possible to build and apply the operator $2^n |1\rangle \langle 1| = E_{1,1} = N = a^\dagger a$ to the output register of a QRM, then we have a family of QRMs that solve 3-SAT in polynomial time. This solution is given in a *semi–uniform* setting: in particular, the program P executed by the QRM depends upon the instance ϕ_n of 3-SAT we want to solve.

6.3 Solving 3-SAT with Quantum UREM P Systems

In this section we finally show how to build a (semi–uniform) family of quantum UREM P systems that solves 3-SAT. Let ϕ_n be an instance of 3-SAT containing n free variables. The structure and the initial configuration of the P system that determines whether ϕ_n is satisfiable is similar to what shown in Figure 2, the only difference being that there are $n + 1$ subsystems instead of n.

As we have done with quantum register machines, let us start by showing how to evaluate ϕ_n for a given assignment of truth values to its variables $x_1, \ldots, x_n$. The input values are set as the energies $|\varepsilon_{x_i}\rangle$ of the harmonic oscillators associated with the membranes from 1 to n. The energy (eventually) associated with the skin membrane is not used. The $(n + 1)$-th membrane, whose harmonic oscillator will contain the output at the end of the computation, is initialized with $|\varepsilon_0\rangle$. The alphabet A consists of all the possible values which can be assumed by the program counter of the QRM that evaluates ϕ_n. In the initial configuration the P system contains only one copy of the object $|p_1\rangle$, corresponding to the initial value of the program counter, in the region enclosed by the skin membrane.

The evaluation of ϕ_n could be performed by simulating the QRM obtained from ϕ_n as explained in the previous section. However, we can obtain a slightly

more efficient P system as follows. We start from the program structure (4), which can be obtained from ϕ_n in a straightforward way. Now, let us suppose we must execute the following instruction:

$$k\text{: } \textbf{if } X_{i,j} = 1 \textbf{ then goto } \text{end}_i$$

As told above, this instruction is performed as follows in a register machine:

$$\begin{aligned} k\text{: } & DEC(X_{i,j}), k+1, k+2 \\ k+1\text{: } & INC(X_{i,j}), \text{end}_i \end{aligned}$$

If we had to simulate these two instructions using a quantum UREM P system, we should use the following sum of rules:

$$\underbrace{\left(|p_{\text{end}_i}\rangle \langle p_{k+1}| \otimes a^\dagger\right)}_{k+1\text{: } INC(X_{i,j}), \text{end}_i} + \underbrace{\left(|p_{k+2}\rangle \langle p_k| \otimes |\varepsilon_0\rangle \langle \varepsilon_0| + |p_{k+1}\rangle \langle p_k| \otimes a\right)}_{k\text{: } DEC(X_{i,j}), k+1, k+2} \in R_\ell$$

where $\ell = \langle i, j\rangle$ is the index of the variable (in the set $\{x_1, x_2, \ldots, x_n\}$) which occurs in literal $X_{i,j}$. As we can see, this operator produces the vector $|p_{k+2}\rangle \otimes |\varepsilon_0\rangle$ if the harmonic oscillator of membrane ℓ is in state $|\varepsilon_0\rangle$; otherwise, it produces the vector $|p_{\text{end}_i}\rangle \otimes |\varepsilon_1\rangle$. Hence we can simplify the above expression as follows:

$$\begin{aligned} |p_{\text{end}_i}\rangle \langle p_k| \otimes |\varepsilon_1\rangle \langle \varepsilon_1| + |p_{k+2}\rangle \langle p_k| \otimes |\varepsilon_0\rangle \langle \varepsilon_0| = \\ = |p_{\text{end}_i}\rangle \langle p_k| \otimes a^\dagger a + |p_{k+2}\rangle \langle p_k| \otimes aa^\dagger \end{aligned}$$

We denote this operator by $O^{(1)}_{i,j,k}$. Analogously, if the instruction to be executed is:

$$k\text{: } \textbf{if } X_{i,j} = 0 \textbf{ then goto } \text{end}_i$$

then we use the operator

$$O^{(0)}_{i,j,k} = |p_{\text{end}_i}\rangle \langle p_k| \otimes aa^\dagger + |p_{k+2}\rangle \langle p_k| \otimes a^\dagger a \in R_\ell$$

which produces the vector $|p_{k+2}\rangle \otimes |\varepsilon_1\rangle$ if the harmonic oscillator of membrane ℓ is in state $|\varepsilon_1\rangle$, otherwise it produces the vector $|p_{\text{end}_i}\rangle \otimes |\varepsilon_0\rangle$.

Since the value $|p_{k+1}\rangle$ is no longer used, we can "compact" the program by redefining the operators $O^{(0)}_{i,j,k}$ and $O^{(1)}_{i,j,k}$ respectively as:

$$\begin{aligned} O^{(0)}_{i,j,k} &= |p_{\text{end}_i}\rangle \langle p_k| \otimes aa^\dagger + |p_{k+1}\rangle \langle p_k| \otimes a^\dagger a \\ O^{(1)}_{i,j,k} &= |p_{\text{end}_i}\rangle \langle p_k| \otimes a^\dagger a + |p_{k+1}\rangle \langle p_k| \otimes aa^\dagger \end{aligned}$$

The "**goto** end" instructions in (4) can be executed as if they were **if** statements whose condition is the negation of the condition given in the previous **if**. Hence the two instructions:

7: **if** $X_{2,3} = 1$ **then goto** end_2
8: **goto** end

can be thought of as:

7: **if** $X_{2,3} = 1$ **then goto** end_2
8: **if** $X_{2,3} = 0$ **then goto** end

which are realized by the operators $O^{(1)}_{2,3,7}$ and $O^{(0)}_{2,3,8}$ (to be added to membrane $\langle 2, 3 \rangle$). The last instruction ($\phi = 1$) of the program can be implemented as follows:

$$|p_{\text{end}}\rangle \langle p_{\text{end}-1}| \otimes a^\dagger$$

to be added to membrane $n + 1$.

For each membrane $i \in \{1, 2, \ldots, n\}$, the set of rules R_i is obtained by summing all the operators which concern variable x_i.

Note that the formulation given in terms of quantum P systems is simpler than the one obtained with QRMs. As usual, if we consider a single assignment to the variables of ϕ_n then at the end of the computation we will obtain the result of the evaluation of ϕ_n as the energy of the output membrane. Instead, if we initialize the harmonic oscillators of the n input membranes with a uniform superposition of all possible classical assignments to $x_1, x_2, \ldots, x_n$, then at the end of the computation the harmonic oscillator of membrane $n + 1$ will be in one of the following states:

- $|0\rangle$, if ϕ_n is not satisfiable;
- a superposition $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, with $\alpha_1 \neq 0$, if ϕ_n is satisfiable.

Once again, we add the rule:

$$|p_{\text{end}}\rangle \langle p_{\text{end}}| \otimes 2^n |1\rangle \langle 1| \in R_{n+1}$$

to membrane $n + 1$ to extract the result.

We have thus obtained a family of quantum UREM P systems which solves 3-SAT in polynomial time. Also this scheme works in the *semi–uniform* setting: in fact, it is immediately verified that the rules of the system depend upon the instance ϕ_n of 3-SAT to be solved.

7 Conclusions and Directions for Future Research

In this paper we have overviewed the state of the art concerning quantum UREM P systems. Starting from basic notions of quantum computers and quantum mechanics, we have seen how quantum register machines and quantum UREM P systems can be defined.

Subsequently, we have proved that such quantum models of computation are computationally complete, that is, they are able to compute any partial recursive function $f : \mathbf{N}^\alpha \to \mathbf{N}^\beta$. This result has been obtained by simulating classical

deterministic register machines. Moreover, we have shown a family of QRMs and a family of quantum UREM P systems that solve (in the semi–uniform setting) the 3-SAT **NP**–complete decision problem in polynomial time. Their construction relies upon the following assumptions: (1) an external observer is able to discriminate, as the result of a measurement, a null vector from a non–null vector, and (2) a specific non–unitary operator, which can be expressed using creation and annihilation operators, can be realized as an instruction of the quantum register machine, and as a rule of the quantum P system, respectively.

One possible direction for future research is to study the computational properties of quantum P systems which contain and process entangled objects. Another line of research is to study the limits of the computational power of quantum P systems by attacking harder than NP-complete problems. In particular, we conjecture that EXP-complete problems can be solved in polynomial time with quantum P systems.

References

1. G. Alford. Membrane systems with heat control. In *Pre–Proceedings of the Workshop on Membrane Computing (WMC-CdeA2002)*, Curtea de Arges, Romania, August 2002.
2. A. Alhazov, R. Freund, A. Leporati, M. Oswald, C. Zandron. (Tissue) P Systems with Unit Rules and Energy Assigned to Membranes. *Fundamenta Informaticae*, 74:391–408, 2006.
3. A. Barenco, D. Deutsch, A. Ekert, R. Jozsa. Conditional quantum control and logic gates. *Physical Review Letters*, 74:4083–4086, 1995.
4. G. Benenti, G. Casati, G. Strini. *Principles of quantum computation and information – volume I: basic concepts*, World Scientific, 2004.
5. P. Benioff. Quantum Mechanical Hamiltonian Models of Discrete Processes. *Journal of Mathematical Physics*, 22:495–507, 1981.
6. P. Benioff. Quantum Mechanical Hamiltonian Models of Computers. *Annals of the New York Academy of Science*, 480:475–486, 1986.
7. D. Deutsch. Quantum Theory, the Church–Turing Principle, and the Universal Quantum Computer. In *Proceedings of the Royal Society of London*, A 400:97–117, 1985.
8. R.P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.
9. R.P. Feynman. Quantum Mechanical Computers. *Optics News*, 11:11–20, 1985.
10. R. Freund, Energy-Controlled P systems. In Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (eds.), *Membrane Computing*, Proceedings of the International Workshop WMC-CdeA 2002, Curtea de Argeş, Romania, August 2002, LNCS 2597, Springer–Verlag, Berlin, 2003, pp. 247–260.
11. R. Freund, A. Leporati, M. Oswald, C. Zandron. Sequential P Systems with Unit Rules and Energy Assigned to Membranes. In *Proceedings of Machines, Computations and Universality, (MCU 2004)*, Saint–Petersburg, Russia, September 21–24, 2004, LNCS 3354, Spriger–Verlag, Berlin, 2005, pp. 200–210.
12. R. Freund, M. Oswald. GP Systems with Forbidding Context. *Fundamenta Informaticae*, 49(1-3):81–102, 2002.

13. R. Freund, Gh. Păun. On the Number of Non-terminals in Graph-controlled, Programmed, and Matrix Grammars. In: M. Margenstern, Y. Rogozhin (eds.), *Proc. Conf. Universal Machines and Computations*, Chişinău, 2001, LNCS 2055, Springer–Verlag, Berlin, 2001, pp. 214–225.
14. R. Freund, Gh. Păun. From Regulated Rewriting to Computing with Membranes: Collapsing Hierarchies. *Theoretical Computer Science*, 312:143–188, 2004.
15. P. Frisco. The conformon–P system: a molecular and cell biology–inspired computability model. *Theoretical Computer Science*, 312:295–319, 2004.
16. M.R. Garey, D.S. Johnson. *Computers and Intractability. A Guide to the Theory on NP–Completeness*. W.H. Freeman and Company, 1979.
17. D. Gottesman. Fault-tolerant quantum computation with higher-dimensional systems. *Chaos, Solitons, and Fractals*, 10:1749–1758, 1999.
18. J. Gruska. *Quantum Computing*. McGraw–Hill, 1999.
19. A. Leporati, S. Felloni. Three "Quantum" Algorithms to Solve 3-SAT. *Theoretical Computer Science*, 372:218–241, 2007.
20. A. Leporati, G. Mauri, C. Zandron. Quantum Sequential P Systems with Unit Rules and Energy Assigned to Membranes. In R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa (eds.), *Membrane Computing: 6^{th} International Workshop (WMC 2005)*, Vienna, Austria, July 18–21, 2005, LNCS 3850, Springer–Verlag, Berlin, 2006, pp. 310–325.
21. A. Leporati, D. Pescini, C. Zandron. Quantum Energy–based P Systems. In *Proceedings of the First Brainstorming Workshop on Uncertainty in Membrane Computing*, Palma de Mallorca, Spain, November 8–10, 2004, pp. 145–167.
22. A. Leporati, C. Zandron, G. Mauri. Simulating the Fredkin gate with energy–based P systems, *Journal of Universal Computer Science*, 10(5):600–619, 2004.
23. A. Leporati, C. Zandron, G. Mauri. Universal Families of Reversible P Systems. In *Proceedings of Machines, Computations and Universality (MCU 2004)*, Saint–Petersburg, Russia, September 21–24, 2004, LNCS 3354, Springer–Verlag, Berlin, 2005, pp. 257–268.
24. A. Leporati, C. Zandron, G. Mauri. Reversible P systems to simulate Fredkin circuits. *Fundamenta Informaticae*, 74:529–548, 2006.
25. M.L. Minsky. *Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, New Jersey, 1967.
26. M.A. Nielsen, I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
27. Gh. Păun. Computing with Membranes. *Journal of Computer and System Sciences*, 1(61):108–143, 2000. See also Turku Centre for Computer Science – TUCS Report No. 208, 1998.
28. Gh. Păun. *Membrane Computing. An Introduction*. Springer–Verlag, Berlin, 2002.
29. Gh. Păun, M.J. Pérez-Jiménez. Recent computing models inspired from biology: DNA and membrane computing. *Theoria*, 18:72–84, 2003.
30. Gh. Păun, G. Rozenberg. A guide to membrane computing. *Theoretical Computer Science*, 287(1):73–100, 2002.
31. Gh. Păun, Y. Suzuki, H. Tanaka. P systems with energy accounting. *International Journal Computer Math.*, 78(3):343–364, 2001.
32. The P systems Web page: `http://psystems.disco.unimib.it/`