A Quantum Turing Machine with a Local Hamiltonian

Tino Gramss

SFI WORKING PAPER: 1994-08-047

SANTA FE INSTITUTE

A quantum Turing machine with a local Hamiltonian

Tino Gramss
University of Göttingen, Drittes Physikalisches Institut,
Bürgerstrasse 42-44, D-37073 Göttingen, Germany,
email: gramss@physik3.gwdg.de

August 20, 1994

Abstract

A universal, deterministic Turing machine can be implemented as a closed, locally interacting quantum lattice system. As a consequence, many questions about the long-term dynamics of a quantum system with only local interactions are undecidable.

Furthermore, it follows that every Turing computable function is arbitrarily parallelizible if the computation is performed on the quantum machines described in this article.

1 Introduction

In [1, 2, 3, 4], the concept of a "quantum Turing machine" developed. A description of a quantum computer which is universal in the sense of complexity theory is given in [5]: It can simulate any other quantum Turing machine with only polynomial overhead.

These quantum computers exhibit global interactions. However, the classical Turing machines perform only local operations. Thus, one might have the suspicion that there are universal quantum computers with local Hamiltonians.

For the following reasons it appears important to investigate the power of quantum computers with only local interactions and time independent Hamiltonians as done in [6, 7, 8, 9, 10, 11]. Since there is no dissipation in quantum computation with time independent Hamiltonians, theoretical speed limits are just given by the lengths of the communication pathways

and the speed of light. Nonlocal interactions introduce longer communication pathways. Furthermore, quantum computers with nonlocal interactions are probably more difficult to realize. Finally, programming globally connected quantum machines generally is exponentially more difficult than for machines with only local interactions as argued in [11]. This work proves the existence of locally connected and closed computing quantum systems which are equivalent to universal, deterministic Turing machines.

2 Quantum mechanical, locally connected computing systems

If we want to describe a system which performs computation, we essentially want the system to start in an initial state $|\psi_0\rangle$, which describes the input to our problem in one or the other way. Then we ideally want the system to go through a dynamical evolution so that the final state is $|\psi_T\rangle$, which describes the solution to our problem in one or the other way. For an ideally closed quantum system, the Hamiltonian is time independent. Then, the evolution follows the solution of the Schrödinger equation

$$|\psi(t)\rangle = \exp(-i\mathbf{H}t)\,|\psi_0\rangle \quad . \tag{1}$$

The system exhibits only local interactions if its Hamiltonian $\mathbf{H}$ can be written as the sum of locally acting Hamiltonians.

The system is capable of simulating the computation of a given deterministic machine if a measurement at an arbitrary time only yields a result which belongs to a state of the deterministic machine. Such states are called "computational states". Clearly, the deterministic machine must be reversible, since the Schrödinger equation describes a reversible evolution. However, it has been shown that reversibility is no essential restriction for the power of computing systems [12, 13].

In [6], Feynman showed that a closed, locally interacting quantum system is capable of performing certain deterministic computations. His system consists of a one-dimensional chain of logical quantum gates. The input to this computer is presented to the first gate, the gate processes the information in a reversible way and yields the input for the second gate and so on. Finally, the last gate produces the output of the quantum computer. Not every computation can be performed in this way [15]. The Feynman computer is not universal.

In [7] and [10], Feynman's ideas are used to describe a quantum cellular automaton. The classical version of the automaton described in [10] is not universal, while the one described in [7] is. In the cases of the cellular automata, it is most probable to measure a state where part of the automaton has done more computational steps than another. It is not possible to obtain global synchrony as in the classical case since there is no global synchronous updating which can be performed by local means [14]. Also, it must be emphasized, that even the commonly adopted definition of universality for *classical* cellular automata differs substantially from the definition that is usually meant in complexity theory. For example, for an infinite size cellular automaton as described in [7], the initial conditions also have to be coded on an infinite number of sites. Clearly, that is not necessary for a Turing machine with an infinite size tape. Finite size machines on the other hand cannot be universal.

Another problem is that in all these cases, for the Feynman computer as for the cellular automata, the final result of a computation can only be obtained with a certain probability, which is smaller than one and typically varies with time. Other results of a measurement correspond to other computational states.

This problem has been solved for the Feynman computer by Peres [8]. By generalizing Feynman's idea, he showed that it is possible construct a quantum computer so that if a measurement is performed at a prescribed time T, the final state of the corresponding classical machine is obtained with certainty.

In the next section, ideas of Feynman and Peres will be generalized. This is necessary for the description of the quantum Turing machine. In section 4, a classical, reversible Turing machine M is described. We will make use of ideas given in [13]. In section 5, a local, unitary description is given for M. In section 6, a local Hamiltonian for the corresponding quantum Turing machine is derived.

3 Locally connected quantum systems without control bits

One of the most severe problems with using control bits is that it is not possible to make loops in a simple way. Especially, it is not possible to let exception criteria depend on the state of the computer [15]. However, this is essential for universal computing. Thus, it is necessary to let the path of

the computation depend on the state of the computer rather than on the state of the control bits. This is possible, control bits are not necessary to perform local deterministic computation as will now be shown.

First, Feynman's and Peres's fundamental idea will be reviewed. We will use the notation of Peres [8]. Say, the computer consists of k reversible gates. Each gate is described by a unitary matrix $\mathbf{U}_i$, $i = 0, \ldots, k-1$. The ideal overall unitary evolution of the computer thus is $\mathbf{U}_{\text{ideal}} = \mathbf{U}_{k-1}\mathbf{U}_{k-2}\cdots\mathbf{U}_0$. It is now not possible to simply write the corresponding Hamiltonian as $\mathbf{H} = i \ln \mathbf{U}(1)$ since if the $\mathbf{U}_i$ do not commute, $\mathbf{H}$ is not local [8, 11].

Feynman was able to obtain a local Hamiltonian from the unitary matrices $\mathbf{U}_i$. He introduced a control bit to each of the gates. At the beginning of the computation, only the control bit associated with gate 0 is set to one. On each step of computation, the "cursor", the control bit set to one, is shifted. The cursor points to the gate, that has to perform the computation. In this way, the position of the cursor indicates, how many computational time steps have been performed. Peres's Hamiltonian then reads:

$$\mathbf{H}_{\mathrm{P}} = \sum_{i=0}^{k-1} \omega_i \left(|\, i \rangle \langle i-1 \,|\, \mathbf{U}_i + |\, i-1 \rangle \langle i \,|\, \mathbf{U}_i^{\dagger} \right) \quad . \tag{2}$$

In the Feynman case, $\omega_i = 1$. This clearly is an Hamiltonian since it is a Hermitian matrix. $|\, i \rangle$ is the cursor state after the execution of step i, i.e. after gate $\mathbf{U}_i$ has done its computation. The computational state $|\, \psi_{i-1} \rangle$ has then been converted to the state $|\, \psi_i \rangle$. The computational state $|\, \psi_i \rangle$ describes the state of the corresponding classical, reversible computer after i time steps. Thus, $|\, i \rangle \langle i-1 \,|\, \mathbf{U}_i$ shifts the cursor forward and the operation $\mathbf{U}_i$ is performed. Likewise, $|\, i-1 \rangle \langle i \,|\, \mathbf{U}_i^{\dagger}$ shifts the cursor backward and the inverse computation of gate i is performed.

Expanding the exponential $\exp(-i\mathbf{H}_{\mathrm{P}} t)$, it is now not difficult to see that only computational states can appear upon proper measurement: Considering only the subspace of the control bits, terms in the expansion like $|\, i+1 \rangle \langle i \mid j \rangle \langle j-1 \,|$, or $|\, i+1 \rangle \langle i \mid j \rangle \langle j+1 \,|$ vanish unless $i = j$. That is, the cursor is shifted properly by one position to the left or to the right, indicating how for far the computation has gone. It does not jump by more then one position. And no superposition of computational states arises from a single computational state.

Now it will be shown how to construct a Peres Hamiltonian without control bits. The trick is to somehow substitute the cursor states by more complicated computational states which nevertheless follow unambiguously

one after the other. Consider a computational state $|\ldots, x, y, z, \ldots\rangle$ where x, y, and z each stand for a certain number of quantum bits. The block of the bits x is adjacent to the block of the bits y, and the block of the bits y is adjacent to the block of the bits z. Say, the next computational state shall depend on the state $|\,y\rangle$. Now, divide the possible states of $|\,y\rangle$ into three sets $\mathcal{S}_0$, $\mathcal{S}_1$, and $\mathcal{S}_2$.

A reversible and local update rule for the computer reads as follows. If $|\,y\rangle \in \mathcal{S}_1$,

$$\mathbf{U}_1\,|\ldots, x, y, z, \ldots\rangle = |\ldots, x, y_1'(y,z), z'(y,z), \ldots\rangle \quad ,$$

else, nothing changes. Likewise, if $|\,y\rangle \in \mathcal{S}_2$,

$$\mathbf{U}_2\,|\ldots, x, y, z, \ldots\rangle = |\ldots, x'(x,y), y_2'(x,y), z, \ldots\rangle \quad .$$

For all states $|\,y\rangle \in \mathcal{S}_0$, the identity matrix applies.

That is, if the computer is in a state with $|\,y\rangle$ from $\mathcal{S}_1$, the state $|\,y\rangle$ and the adjacent state $|\,z\rangle$ is altered. If it is in a state with $|\,y\rangle$ from $\mathcal{S}_2$, $|\,y\rangle$ and $|\,x\rangle$ is altered. Else, nothing changes.

Note that for $\mathbf{U}_1$ to be unitary, it is not necessary that the functions y_1' and z' that map one bit string onto another are reversible. Only the overall mapping from $|\,y, z\rangle$ to $|\,y_1', z'\rangle$ must be. Likewise for $\mathbf{U}_2$, y_2' and x'. The unitary matrices can be written as

$$\mathbf{U}_1 = \sum_{|\,y\rangle \in \mathcal{S}_1} \sum_{z} |\,y_1'(y_1, z), z'(y_1, z)\rangle\,\langle y_1, z\,| \quad ,$$

where the sums extend over all possible states $|\,z\rangle$ and all $|\,y\rangle \in \mathcal{S}_1$ and likewise

$$\mathbf{U}_2 = \sum_{|\,y\rangle \in \mathcal{S}_2} \sum_{x} |\,x'(x, y), y_2'(x, y)\rangle\,\langle x, y\,| \quad .$$

We will construct a computer that is described by many of the update rules $\mathbf{U}_i$ as described above. Following Feynman's ideas, the Hamiltonian can now be written

$$\mathbf{H} = \sum_i \omega_i\,(\mathbf{U}_i + \mathbf{U}_i^\dagger) \tag{3}$$

without mentioning any control bits explicitly. It is easily possible to extend the ideas to more neighbors of the block y. The Hamiltonian (2) and those for the cellular automata can then be found as special cases of (3).

Expending (1) with the Hamiltonian from (3) we find terms like

$$\ldots + \frac{(-i)^k}{k!}\mathbf{T}_{i_k}\mathbf{T}_{i_{k-1}}\cdots\mathbf{T}_{i_1} + \ldots \ .$$

where $\mathbf{T}_i$ might be $\omega_i\mathbf{U}_i$ or $\omega_i\mathbf{U}_i^\dagger$. Considering two subsequent matrices $\mathbf{T}_i\mathbf{T}_j$ we may find a case like $\mathbf{U}_i\mathbf{U}_i^\dagger = \mathbf{I}$ or

$$\begin{aligned}&\mathbf{U}_2\mathbf{U}_1\,|\ldots x,y,z\ldots\rangle \\ &= \begin{cases} \mathbf{0} & \text{if } |\,y_1'\rangle \neq |\,y_2\rangle \\ \mathbf{U}_2\,|\,x,y_1',z'\rangle = |\,x',y_2',z'\rangle & \text{else} \end{cases},\end{aligned}$$

and similar for other cases. If the $\mathbf{U}_i$ are chosen properly, only terms that correspond to forward or backward computation appear.

For the Turing machine described below, the $\mathbf{U}_i$ are chosen in a way so that only two adjacent blocks of bits are updated simultaneously. For the rest of the computer, the identity matrix applies. This resembles the Feynman computer, where the blocks to update are the old and the new cursor at position $i-1$ and i and the bits of the computational part associated with gate i.

In such cases, if (1) is expanded, only computational states can be found as the terms of the expansion.

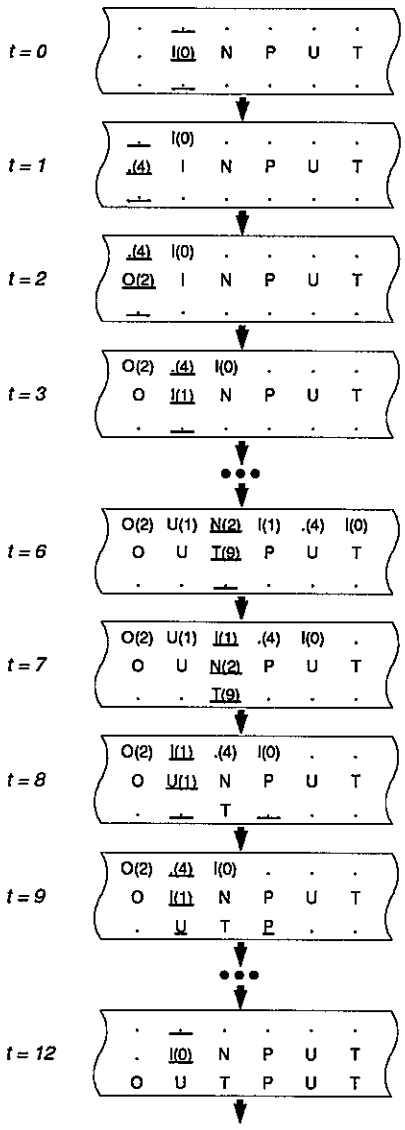
In other cases, where different parts of the computer may be updated simultaneously, the same problem as for the cellular automata occurs: Upon measurement, parts of the computer may have evolved further than others. This is not a problem for the machine that will now be described, since until the final result is stored, only one local region is updated applying $\mathbf{U}$.

4 A classical, reversible Turing machine

Following a blueprint of Bennett [13], we will now construct a reversible Turing machine $\mathbf{M}$ which simulates a usual non-reversible Turing machine $\mathbf{M}^*$.

Let $\mathbf{M}^*$ work on an alphabet $\mathcal{A} = \{a_0, a_1, \ldots, a_n\}$. a_0 is the symbol for an empty field of the tape. The head of $\mathbf{M}^*$ can be in $m+1$ different states $0, 1, \ldots, m$.

The actions of $\mathbf{M}$ are illustrated in figure 1. The reversible machine consists of three interacting, deterministic Turing machines $\mathbf{M}_0$, $\mathbf{M}_1$, and $\mathbf{M}_2$. The fields of the tapes are labeled by the tuples $\mathbf{f} = (p, q)$, where the



t=0: The input for the computation is written on the middle tape. The heads point to the underlined symbols. A dot indicates an empty field.
t=1: The machine starts its computation. The "history" is stored on the upper tape. To ensure reversibility, it is also necessary to store the head states.
t = 2: The former entry .(4) of the middle tape has been stored onto the upper tape by simply overwriting an empty field on the upper tape.
t = 3: The entry O(2) has been stored on the upper tape. Since the upper field was not empty, all its entries had to be shifted to the right to provide an empty field.
t=6: The result of the computation ("output") is now written onto the middle tape.
t=7: While the computation on the middle and the upper tape is reversed, the result is copied to the lower tape.
t=8: Two heads spread out on the lower tape to perform the copy process.
t=9: The reversible computation and the copy process proceed. Note that the middle and upper tape carry the same entries as at t=3.
t=12: The initial entries can be found on the middle and the upper tape. The copy process has finished. The heads on the lower tape keep moving to the left and to the right, "copying" nothing than empty fields onto empty fields. A useless computation on the upper tapes continues, but the output on the lower tape will not be overwritten.

Figure 1: A reversible Turing machine

index q is an integer $\ldots, -2, -1, 0, 1, \ldots$ and the index for the tape p may take the values $0, 1, 2$.

M_1 corresponds to the non-reversible Turing machine M^*. The heads of M_1 can be in $m+1$ different states $0, 1, \ldots, m$. M_0 and M_2 are needed to render the computation reversible.

M_0 and M_1 work on the the same alphabet as M^*. M_2 uses an enlarged alphabet with $(n+1)(m+1)$ symbols, $\mathcal{A}_2 = \mathcal{A} \times \{0, 1, \ldots, m\}$. All the combinations of symbols and head states of M_1 can be found in $\mathcal{A}_2$. A symbol from $\mathcal{A}_2$ will be written (a, v), where a denotes the corresponding symbol form $\mathcal{A}$ and v is a head state of M_1.

In field $\mathbf{f}$, we find the symbol $s(\mathbf{f}) = s(p, q)$.

Initially, all fields of the tape carry the symbol a_0 except a finite-size portion of tape 1, which carries the program and the input. At the beginning, the head positions are $\mathbf{f}_0 = (0, 0)$, $\mathbf{f}_1 = (1, 0)$, $\mathbf{f}_2 = (2, 0)$. The initial state of the middle head is 0. To indicate that the computation has finished, this head state will be set to m.

The evolution of M proceeds by four stages. In the first stage the operations are similar to those of a usual, non-reversible Turing machine. Depending on the symbol under the middle head and on the state the head is in, the machine now performs one of the following three actions on the middle tape: It moves the head to the right, or it moves the head to the left, or it writes a new symbol on the field $\mathbf{f}_h = (p_h, q_h)$ under the head. Then, the state of the head is altered. The new head state and which of the three actions is taken is given by a finite size Turing table with at most $(n+1) \times (m+1)$ rows and four columns. In the first two rows we find the possible combinations of symbols under the head and head states. In the third row, we find a symbol from $\mathcal{A}$ if this symbol will be written, or the symbols l or r to indicate that the head shall move left or right. In the fourth row, the new head state can be found. Thus, a row of a Turing table reads a_i, v_j, b_k, v_l, where $a_i \in \mathcal{A}$, $v_{i,j} \in 0, \ldots, m$ and $b_k \in \mathcal{A} \cup \{r, l\}$. It is interpreted as: "If the symbol under the head is a_i and the head state is v_j, perform the action b_k and set the head state to v_l".

Since the machine must work in a reversible way, it may not simply overwrite the symbol $s(1, q_h)$. Also, it is necessary to store the head states of M_1 upon each step of computation [16]. Thus, on each step of computation, in case a symbol a is written onto the middle tape and the head was in state v, the symbols with $q \geq q_h$ in the upper tape are shifted to the right and the old entry in $s(\mathbf{f}_h)$ together with the old head state is inserted in a field

of the upper tape. If no symbol is written, (a_0, v) is inserted. Thus:

$$s_{\text{new}}(p,q) = \begin{cases} s_{\text{old}}(p, q-1) & \text{if } p = 2, q > q_h \\ s_{\text{old}}(p,q) & \text{if } p = 2, q < q_h \\ s_{\text{old}}(1, q_h) & \text{if } p = 2, q = q_h \\ (a, v) \text{ cf. } (a_0, v) & \text{if } p = 1, q = q_h \\ s_{\text{old}}(p,q) & \text{else} \end{cases} \tag{4}$$

This scenario goes on until the machine has calculated its final result, which can now be found on tape 1. It is not difficult to see that it is possible to implement shift operations as described above on an ordinary Turing machine.

In the second stage, all the symbols on tape 1 which are not equal to a_0 are copied one by one onto tape 0. Thus,

$$s_{\text{new}}(p,q) = \begin{cases} s_{\text{old}}(1,q) & \text{if } p = 0 \\ s_{\text{old}}(p,q) & \text{else} \end{cases}$$

The third stage is the clean-up stage: The evolution of the first stage is reversed. Additionally, at the end of the third stage, the state of the middle head is set to m. The computation is finished.

As is illustrated in figure 1, the second and the third stage may take place simultaneously.

We end up with the same tape contents as at the beginning, except that in tape 0 we find the final result. We find the heads in there start positions $\mathbf{f}_0$, $\mathbf{f}_1$, and $\mathbf{f}_2$.

A reversible machine is not allowed to stop computation. This is why a fourth stage has to be introduced. In this stage some "useless" computations have to be done. Since none of the machine's states may repeat (otherwise reversibility is violated [18]), an infinite size portion of the tape has to be available for this task. It can be accomplished in the following way. As soon as the middle head is in state m, it continues writing a symbol $a \neq a_0$ onto the tape at position $\mathbf{f}_1$. On each write-operation the symbols are shifted according to (4).

5 Local, unitary description of M

In this section, we describe a local, unitary evolution of the quantum computer and how it relates to M.

Let the three tapes consist of a lattice of spin-$\frac{1}{2}$-variables, each representing a "bit" in our quantum computer. Each spin can be in the state "up" for a logical 1 and "down" for a logical 0. Thus, a 0-bit will correspond to the quantum state $|0\rangle$ and a 1-bit to $|1\rangle$, where $|0\rangle$ and $|1\rangle$ are the eigenstates of the Pauli matrix σ_z [19].

To each field of the tapes,

$$\lceil \log_2(n+1) \rceil + \lceil \log_2(m+1) \rceil + 2$$

spins are associated, where $\lceil k \rceil$ stands for the smallest integer larger than or equal to k. $\lceil \log_2(n+1) \rceil$ spins are needed to code one of the $n+1$ possible symbols on the field. $\lceil \log_2(m+1) \rceil$ spins are needed to code the state of the head. One additional "head spin" tells, whether the head points to the field ($|1\rangle$) or not ($|0\rangle$). The head spins set to $|1\rangle$ will move around on the lattice indicating the position of the heads. The second additional bit is a reverse flag: It tells whether the machine performs forward computation ($|0\rangle$) or reverse computation ($|1\rangle$) as necessary during the clean up stage. The whole lattice consists of three adjacent rows of fields, each row corresponds to one tape.

To code a symbol a and a head state v, we write $|a, v, h, r\rangle$. h and r stand for the head flag cf. for the reverse flag. a, v, h, and r are written as binaries.

Now we ask for local unitary matrices that describe the behavior of M.

Forward computation

First, consider the stage of forward computation. In this case, the reverse flag of each field is set to $|0\rangle$. Now, the task is to find a unitary matrix, which describes the action of one step of computation according to the Turing table if the middle head is in position **f**. Say, the head of M points to the symbol a_i, and the head state is v_j. Now we have to find the unitary transformation which belongs to the row a_i, v_j, b_k, v_l.

Let us assume for the moment that the field $(2,k)$ is empty, $s(2,k) = (a_0, v_0)$. Quantum mechanically, this can be expressed in the following way: Field $(1,k)$ is in state $|s(1,k)\rangle = |a_i, v_j, 1, 0\rangle$ and field $(2,k)$ is in state $|s(2,k)\rangle = |a_0, v_0, 1, 0\rangle$. $|s(1, k \pm 1)\rangle = |a^{\pm}, v_0, 0, 0\rangle$ and $|s(0,k)\rangle = |a_0, v_0, h, 0\rangle, h \in \{0,1\}$.

The state of the fields under consideration is their tensor product. Sym-

bolically:

$$\begin{array}{lccc} \text{tape 2:} & |.,.,.,.\rangle & |a_0,v_0,1,0\rangle & |.,.,.,.\rangle \\ \text{tape 1:} & |a^-,v_0,0,0\rangle & |a_i,v_j,1,0\rangle & |a^+,v_0,0,0\rangle \\ \text{tape 0:} & |.,.,.,.\rangle & |.,.,.,.\rangle & |.,.,.,.,\rangle \end{array}$$

In this case, on the subspace spanned by $|s(1,k)\rangle$, $|s(1,k\pm 1)\rangle$, and $|s(2,k)\rangle$, the required unitary evolution according to the Turing table exists: The evolution is reversible and, since the states are orthogonal, the corresponding matrix is a permutation matrix.

Now say, field $(2,k)$ is not empty. In this case, all the symbols on tape 2 have to be shifted right and finally, $|s(2,k)\rangle$ has to be set to $|a_0,v_0,1,0\rangle$. During the shift operation, the actions on tape 1 have to be suspended, i.e. the identity matrix applies.

The problem now is that the shift operation has to be performed by local, unitary means. It can be performed in the following way.

From the position the head of tape 1 points to, a signal (spin wave) has to be send to the far right end of tape 2, indicating that one empty field has to be provided. As soon as the spin wave reaches the first empty field, subsequent exchange operations of the form $|a,v,1,0\rangle \leftrightarrow |a_0,v_0,0,0\rangle$ "carry" an empty field to the head position of tape 1.

This, again, can be done by locally acting, unitary transformations. Say the head bit of tape 1 points to the files $(2,k)$, while the head bit of tape 2 points to $(2,k')$. The rules read as follows:

a Bit shift: If there is no empty field at position $(2,k'+1)$, shift the head bit 2 right. Symbolically:

$$\begin{array}{lcc} \text{tape 2:} & \ldots|a,v,1,0\rangle & |a',v',0,0\rangle\ldots \\ & \downarrow & \\ \text{tape 2:} & \ldots|a,v,0,0\rangle & |a',v',1,0\rangle\ldots \end{array}$$

b Exchange: If there is an empty field at $(2,k'+1)$, exchange the empty field and the field $(2,k')$ and shift the head spin 2 appropriately and in a reversible way:

$$\begin{array}{lccc} \text{tape 2:} & \ldots|a',v',0,0\rangle & |a,v,1,0\rangle & |a_0,v_0,0,0\rangle\ldots \\ & \downarrow & & \\ \text{tape 2:} & \ldots|a',v',0,0\rangle & |a_0,v_0,1,0\rangle & |a,v,0,0\rangle\ldots \\ & \downarrow & & \\ \text{tape 2:} & \ldots|a',v',1,0\rangle & |a_0,v_0,0,0\rangle & |a,v,0,0\rangle\ldots \end{array}$$

This operation is repeated, until the empty field is at position $(2,k)$, i.e. until head 1 and head 2 are aligned, a decision which can be done by local means. The last line of the scenario described above is omitted in this case. Instead, the operation according to the Turing table takes place.

After a computational step according to the Turing table, $|s(2,k)\rangle = |a_i, v_j, 1, 0\rangle$, the old state of machine $\mathbf{M}_1$ is saved. The head bit in row 1 has been shifted to the right, to the left, or it has not moved. In case it has moved, say to field $(1,k)$, it might be that field $(2,k)$ is already empty (beside head spin 2, which will be set). Then, the next Turing step (a step according to the Turing table for the machine $\mathbf{M}^*$) may follow immediately.

On each Turing step, the head spin 2 is shifted in the same way as head spin 1. Furthermore, if the head is shifted, the state bits of the previous field is set to v_0. Since this happens along with the copy process to tape 2 (i.e. with the same unitary transformation), reversibility is not violated.

Copy to tape 0

If the head state is set to v_m, the computation of the corresponding machine M^* has finished. In this case, the operations on tape 1 and tape 2 are suspended, the identity matrix applies. At the same time, a copy process is set to work that saves the result, which can be found on tape 1, onto tape 0. This, again, can be done in a local and unitary way. At the beginning, all the head spins of tape 0 are set to $|0\rangle$. As soon as the head state at position $(1,k)$ is set to v_m, two spin waves starting at $(0,k)$ are send along the head spin sites of tape 0, one to the left and one to the right. Among other things, which will be discussed in a moment, a copy operation onto a field of tape 0 is performed, if the head spin 0 passes by:

$$
\begin{array}{lllll}
\text{tape 1:} & \ldots|a^-, v_0, 0, 0\rangle & |a, v, 1, 0\rangle & |a^+, v_0, 0, 0\rangle \ldots \\
\text{tape 0:} & \ldots|a_0, v_0, 0, 0\rangle & |a_0, v_0, 0, 0\rangle & |a_0, v_0, 0, 0\rangle \ldots \\
 & \downarrow & & \\
\text{tape 1:} & \ldots|a^-, v_0, 0, 0\rangle & |a', v_m, 1, 0\rangle & |a^+, v_0, 0, 0\rangle \ldots \\
\text{tape 0:} & \ldots|a_0, v_0, 0, 0\rangle & |a', v_m, 1, 0\rangle & |a_0, v_0, 0, 0\rangle \ldots \\
 & \downarrow & & \\
\text{tape 1:} & \ldots|a^-, v_0, 0, 0\rangle & |a', v_m, 1, 1\rangle & |a^+, v_0, 0, 0\rangle \ldots \\
\text{tape 0:} & \ldots|a^-, v_0, 1, 0\rangle & |a^-, v_m, 1, 0\rangle & |a^+, v_0, 1, 0\rangle \ldots
\end{array}
$$

Note that the head spins set to $|1\rangle$ are not reset. This is a simple way to indicate that the copy process has taken part and should not be repeated.

Of course, this makes the notion of a "head spin" not compatible to the usual one, where only one head belongs to one tape.

The copy process may go on ad infinitum. From a certain point on, only empty fields are copied to tape 0. If nothing would change locally at this points, local unitarity would be violated. But the head spins are still spreading on tape 0. This is how reversibility is obtained.

Reverse computation

As soon as the head spins on tape 0 are set, a spin wave on the reverse flags spreads out to indicate that the reverse computation starts. The reverse flag is set subsequently to the step which produces the state $|a', v_m, 1, 0\rangle$ as shown in the last line of the previous diagram. Consider the fields on which one of the local, unitary operators works. If none of the reverse flags are set, the forward operation as described above is performed. If all those reverse flags are set, the reverse operation is performed. If part of them are set, nothing happens.

"Useless" computation

For the machine M, an additional stage subsequent to the reverse computation had to be introduced to obtain reversibility. This is not necessary for the quantum machine described so far, since the spin wave associated with the copying process spreads out infinitely. This is the substitute for the useless computations of M.

6 Locally interacting Quantum Turing machine

Similar to the idea of Feynman, the Hamiltonian $\mathbf{H}$ for the Turing machine is

$$\mathbf{H} = \mathbf{U}_l + \mathbf{U}_l^\dagger = \sum_i \mathbf{U}_i + \sum_i \mathbf{U}_i^\dagger \ . \tag{5}$$

Only superpositions of computational states arise if $\mathbf{U}$ from (1) is applied to such a superposition. Upon proper measurement, a state of the reversible Turing machine M after a certain number of computational steps will be found. This is because the $\mathbf{U}_i$ and the computational states are chosen in a way so that the computation may only follow the computational path in a unambiguous manner. This is similar to the original Feynman case, where the uniqueness of the computational path is simply maintained by

the unambiguous succession of the cursor states to which the computation is coupled.

The question now is, whether it is possible to choose the ω_i in a way that it is possible to get a result with certainty. Two problems occur.

First, in order to apply Peres's idea [8] to the Feynman computer, it was necessary to know the time T at which the result has been computed in advance *exactly*. Since this time cannot be predicted by a general algorithm, no *universal* Turing machine can be constructed with the Peres idea. However, for many calculations we might be able to give reasonable upper time limits, after which we expect the computation to have finished. For the machine described above, since the result is stored safely, we do not have to know the exact T.

The second problem is that the Peres idea only works, if we know exactly in what order the unitary matrices have to be applied, since the coefficient ω_i belongs to the unitary matrix at computational time i. This resembles the old problem of the Benioff computer [1] where the whole computational process had to be known in advance.

7 Parallelizability

In [11], it is shown how certain limitations of computational speed arise for local quantum computation as described in [6, 7, 10]. This limitations apply as well to the computer described in this article. Until the computation is finished, i.e. until a head state of tape 1 is set to v_m, only one computational operation is performed as $\mathrm{U_I}$ is applied. (Afterwards, more operations are performed since the spreading of copy spins and reverse flags appears simultaneously and in both directions.) In this case, the limit for the quantum mechanical average of computational speed is $|\langle \mathbf{V} \rangle| \leq 2\epsilon/\hbar$, where the computational velocity is measured as "operations per time interval". ϵ is an energy of the order of a typical spin-spin-coupling.

However, this limitation on quantum speed applies only to the quantum mechanical ensemble average and does not apply for a single measurement. This can easily be seen by expanding the exponential. Say we start the computation in state $|\psi_0\rangle$ and we get a state $|\psi_k\rangle$ that contains a final result after k computational steps of the machine M. That means, k local unitary transformations U_0, U_1, $\cdots$, U_{k-1}, applied to $|\psi_0\rangle$ in correct order

yield $|\psi_k\rangle$. Then, expanding the exponential, we find a term

$$\ldots + \frac{(-i)^k}{t^k} \mathrm{U}_{k-1}\mathrm{U}_{k-2}\ldots\mathrm{U}_0 + \ldots$$

Thus, upon measurement, since the computational states are orthogonal to each other, we find $|\psi_T\rangle$ with a probability $\left|\frac{1}{i^{2T}}\right|$. Even after arbitrary short times τ, we will find the final result with a certain (small) probability.

This can be expressed in a different way. Say, you have n quantum computers as described above. Let all of them start under the same conditions, to evaluate a Turing computable (i.e. recursive) function $f(i)$. Then it is possible to make simultaneous measurements at time τ on all of the quantum computers. Since all n computers evolve statistically independent, the probability to find a final result can be made arbitrarily close to 1 by increasing n. Thus, with local quantum computation, it is possible to achieve arbitrary "parallelizability" for all Turing computable functions.

8 Undecidability

Using the ideas in this article, we find to every Turing machine a lattice with local interactions. Thus, to every undecidable question we find that the long term dynamics of the corresponding lattice system is undecidable as well.

One of these undecidable questions is, whether or not the spin wave associated to the copy process will ever occur. This is because the spin wave spreads out as soon as the computation has finished. To answer the question whether a computation finishes is Turing's famous Halting problem, which is undecidable.

It was shown in [20] that by analyzing the Hamiltonian the existence of a zero-energy ground state in a Potts spin lattice with local interactions may be undecidable. This is another kind of undecidability as the one under discussion. In general, it is only by infinitely slow processes (i.e. only with an infinitely large number of time steps) possible to decide by physical means, whether spin lattice systems have a zero-energy ground state. This is even true for systems with a relatively simple Hamiltonian, where the question for such a ground state is certainly mathematically decidable, if only we were allowed to analyze the given Hamiltonian mathematically. Thus, questions about the long term dynamics of physical systems as described in [20] do not lead to undecidability in the sense of complexity theory. However, underlying mathematical questions are indeed undecidable.

9 Conclusion

I showed that it is possible to implement Turing computation on a two-dimensional spin lattice system. The correspondence between reversible Turing machines and the system described in this article is very close. This makes it easy to apply complexity theoretical results about reversible machines to locally acting quantum computers.

Certain questions about the dynamics of classical chaotic systems are undecidable, as shown in [21, 22]. Since it was shown that quantum computation can be universal, it follows that there are many undecidable questions concerning the long term dynamics of quantum systems.

Speed limits for the quantum computations discussed in this article can be found, which are due to the locality of the Hamiltonian. However, in contrast to computation by classical means, there is no limit for the parallelizability of quantum computation. If there is no restriction on the number of quantum processors, every Turing computable function can be evaluated in arbitrarily short time.

Acknowledgements

This work was supported by the Santa Fe Institute and by a postdoctoral grant from the Deutsche Forschungsgemeinschaft (German Research Community).

References

[1] P.A.Benioff: "Quantum-mechanical models of Turing-Machines that dissipate no energy." *Phys.Rev.Let.* **48** (23), 1980, pp. 1581-1585.

[2] P.A.Benioff: "A Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines." *J.Stat.Mechanics* **22**, 1980, pp. 563-591.

[3] P.A.Benioff: "Quantum mechanical Hamiltonian models of discrete processes that erase their own history – Application to Turing-Machines." *Int.J.Theo.Phys.* **21** (3-4), 1982, pp. 177-201.

[4] D.Deutsch: "Quantum theory, the Church-Turing principle and the universal quantum computer." *Proc. R. Soc. Lond.* **A400**, 1985, pp. 97-117.

[5] E.Bernstein and U.Vazirani: "Quantum Complexity Theory." *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, 1993, pp. 11-20.

[6] R.Feynman: "Quantum Mechanical Computers" *Foundations of Physics* **16**, No. 6, 1986, pp. 507-531.

[7] N.Margolus: "Quantum Computation" *New Techniques in Quantum Measurement Theory*, in Annals of the New York Academy of Sciences, **480**, 1986, pp. 487-497.

[8] A.Peres: "Reversible Logic and quantum computation." *Phys. Rev. A* **32**, No.6, 1985, pp. 3266-3276.

[9] G.Grössing and A.Zeilinger: "Structures in quantum cellular automata." *Physica* **B151**, 1988, pp.366-370.

[10] N.Margolus: "Parallel Quantum Computation" in "Complexity, Entropy, and the Physics of Information, SFI Studies in the Sciences of Complexity", **VIII**, Ed. W.H.Zurek, Addison-Wesley 1990, pp.273-287.

[11] T.Gramss: "On the speed of quantum computation", Santa Fe Institute Working Paper Series 94-04-017, 1994.

[12] C.Bennett: "Logical reversibility of computation." *IBM J. Res. Develop.* **6**, 1979, pp. 525-532.

[13] C.Bennett: "Time-space trade-offs for reversible computation." *Siam J. Comp.* **18(4)**, 1989, pp. 766-776.

[14] A.Peres: "Measurement of time by quantum clocks" *Am. J. Phys.*, Vol. 48, 1980, p.552.

[15] With the Feynman setup, only primitive–recursive functions can be evaluated, as will be published elsewhere. For universal computation, it is necessary to evaluate recursive functions.

[16] The storage of head states is not provided by the machine described in [13]. However, this is necessary for the following reason: Consider two states of M_1 where the tape contents and the position and state of the head is the same. In this case, in contradiction to reversibility, it is not clear what the previous state was. Clearly, in this case M^*, the corresponding non-reversible machine, would work in a cyclic way, what

is somehow useless. Thus, one might be tempted to exclude cyclicity. But this is not possible by a general algorithm, since, according to Rice's theorem [17] the question of cyclicity it is not decidable.

[17] H.G.Rice: "Classes of recursively enumerable sets and their decision problem." *Transactions of the American Mathematical Society* **74**, 1953, pp. 341-360. See also H.Rogers, Jr.: *Theory of Recursive Functions and Effective Computability.* MIT Press, 1987, p. 34.

[18] There is only one case, where reversibility is not violated but the state sequence of the machine is cyclic: The whole computation must form a cycle. Only in this case are there no ambiguities in case of the "backward" computation. But then the machine will have to erase the result in row 1 occasionally. Thus, if we look at our machine, even if we waited an arbitrary long time, we never can be sure to find the result.

[19] Alternatively, the machine described here can be simulated on a 2-dimensional Potts spin system with fewer spin variables.

[20] I.Kanter: "Undecidability Principle and the Uncertainty Principle even for Classical Systems." *Phys.Rev.Let.* **64**, No. 4, 332-335, 1990.

[21] C.Moore: "Unpredictability and Undecidability in Dynamical Systems." *Phys.Rev.Let.* **64**, No. 20, 2354-2357, 1990.

[22] C.Moore: "Smooth Maps of the Interval and the Real Line Capable of Universal Computation." Santa Fe Institute Working Paper Series 93-01-001, 1993.