Reversible Simulation of Irreversible Computation

Ming Li*
Computer Science Department
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1
(mli@math.uwaterloo.ca)

Paul Vitányi†
CWI
Kruislaan 413
1098 SJ Amsterdam
The Netherlands
(paulv@cwi.nl)

Abstract

Reversible simulation of irreversible algorithms is analysed in the stylized form of a 'reversible' pebble game. While such simulations incur little overhead in additional computation time, they use a large amount of additional memory space during the computation. We show that among all simulations which can be modelled by the pebble game, Bennett's simulation is optimal in that it uses the least auxilliary space for the greatest number of simulated steps. We give a trade-off of storage space versus irreversible erasure. Examples of reversible algorithms are algorithms for quantum computers.

1 Introduction

The ultimate limits of miniaturization of computing devices, and therefore the speed of computation, are constrained by the increasing density of switching elements in the device. Linear speed up by shortening interconnects on a two-dimensional device is attended by cubing the dissipated energy per area unit per second. Namely, we square the number of switching elements per area unit and linearly increase the number of switching events per switch per time unit. The attending energy dissipation on this scale in the long run cannot be compensated for by cooling. Reduction of the energy dissipation per elementary computation step therefore determines future advances in computing power.

Since 1940 the dissipated energy per bit operation in a computing device has with remarkable regularity decreased by roughly one order of magnitude (tenfold) every five years, [8, 10]. Extrapolations of current trends show that the energy dissipation per binary logic operation needs to be reduced below kT (thermal noise) within 20 years. Here k is Boltzmann's constant and T the absolute temperature in degrees Kelvin, so that $kT \approx 3 \times 10^{-21}$ Joule at room temperature. Even at kT level, a future device containing 10^{18} gates in a cubic centimeter operating at a gigahertz dissipates about 3 million watts/second. For thermodynamic reasons, cooling the operating temperature of such a computing device to almost absolute zero (to get kT down) must dissipate at least as much energy in the cooling as it saves for the computing, [14].

Considerations of thermodynamics of computing started in the early fifties. J. von Neumann reputedly thought that a computer operating at temperature T must dissipate at least $kT \ln 2$ Joule per elementary bit operation, [16]. But R. Landauer [9] demonstrated that it is only the 'logically irreversible' operations in a physical computer that are required to dissipate energy by generating a corresponding amount of entropy for each bit of information that gets irreversibly erased. As a consequence, any arbitrarily large reversible computation can be performed on an appropriate physical device using only one unit of physical energy *in principle*.

Examples of logically reversible operations are 'copying' of records, and 'canceling' of one record with respect to an identical record provided it is known that they are identical. They are physically realizable (or almost realizable) without energy dissipation. Such operations occur when a program sets $y := x$ and later (reversibly) erases $x := 0$ while retaining the same value in y. We shall call such reversible erasure 'canceling' x against y. Irrespective of the original contents of variable x we can always restore x by $x := y$. However, if the program has no copy of the value in variable x which can be identified by examining the program without knowing the contents of the variables, then after (irreversibly) erasing $x := 0$ we cannot restore the original contents of x even though some variable z may have by chance the same contents. 'Copying' and 'canceling' are logically reversible, and their energy dissipation free execution gives substance to the idea that logically reversible computations can be performed with zero energy dissipation.

Generally, an operation is *logically reversible* if its inputs can always be deduced from the outputs. Erasure of information in a way such that it cannot be retrieved is not reversible. Erasing a bit irreversibly necessarily dissipates $kT \ln 2$ energy in a computer operating at temperature T. In contrast, computing in

*Supported in part by NSERC operating grant OGP-046506, ITRC, and a CGAT grant.

†Partially supported by the European Union through NeuroCOLT ESPRIT Working Group Nr. 8556, and by NWO through NFI Project ALADDIN under Contract number NF 62-376 and NSERC under International Scientific Exchange Award ISE0125663. Affiliations are CWI and the University of Amsterdam.

a logically reversible way says nothing about whether or not the computation dissipates energy. It merely means that the laws of physics do not require such a computer to dissipate energy. Logically reversible computers built from reversible circuits, [7], or the reversible Turing machine, [2], implemented with current technology will presumably dissipate energy but may conceivably be implemented by future technology in an adiabatic fashion. Current conventional electronic technologies for implementing 'adiabatic' logically reversible computation are discussed in [14, 15]. Hypothetical computers that are both logically and physically perfectly reversible and perfectly free from energy dissipation are the billiard ball computer, [7], and the coherent quantum computer, [6, 5]. This paper is a preliminary version of part of a more extensive treatment of time and space versus energy trade-offs in reversible computing, and reversible simulation of irreversible computing, in [13].

2 Previous Work

Currently, we are used to design computational procedures containing irreversible operations. To perform the intended computations without energy dissipation the related computation procedures need to become completely reversible. Fortunately, all irreversible computations can be simulated in a reversible manner, [11, 1]. All known reversible simulations do not change the computation time significantly, but do require considerable amounts of auxiliary memory space. In this type of simulation one needs to save on space; time is already almost optimal.

The reversible simulation in [1] of T steps of an irreversible computation from x to $f(x)$ reversibly computes from input x to output $\langle x, f(x)\rangle$ in $T' = O(T)$ time. However, since this reversible simulation at some time instant has to record the entire history of the irreversible computation, its space use increases linear with the number of simulated steps T. That is, if the simulated irreversible computation uses S space, then for some constant $c > 1$ the simulation uses $T' \approx c + cT$ time and $S' \approx c + c(S+T)$ space. This can be an unacceptable amount of space for many practically useful computations.

The question arises whether one can reduce the amount of auxiliary space needed by the simulation by a more clever simulation method or by allowing limited amounts of irreversibility.

In [3] another elegant simulation technique is devised reducing the auxiliary storage space. This simulation does not save the entire history of the irreversible computation but it breaks up the simulated computation into segments of about S steps and saves in a hierarchical manner *checkpoints* consisting of complete instantaneous descriptions of the simulated machine (entire tape contents, tape heads positions, state of the finite control). After a later checkpoint is reached and saved, the simulating machine reversibly undoes its intermediate computation reversibly erasing the intermediate history and reversibly canceling the previously saved checkpoint. Subsequently, the computation is resumed from the new checkpoint onwards.

The reversible computation simulates k^n segments of length m of irreversible computation in $(2k-1)^n$ segments of length $\Theta(m+S)$ of reversible computation using $n(k-1)+1$ checkpoint registers using $\Theta(m+S)$ space each, for each k, n, m.

This way it is established that there are various trade-offs possible in time-space in between $T' = \Theta(T)$ and $S' = \Theta(TS)$ at one extreme ($k = 1, m = T, n = 1$) and (with the corrections of [12]) $T' = \Theta(T^{1+\epsilon}/S^{\epsilon})$ and $S' = \Theta(c(\epsilon)S(1 + \log T/S))$ with $c(\epsilon) = \epsilon 2^{1/\epsilon}$ for each $\epsilon > 0$, using always the same simulation method but with different parameters k, n where $\epsilon = \log_k(2k-1)$ and $m = \Theta(S)$. Typically, for $k = 2$ we have $\epsilon = \log 3$. Since for $T > 2^S$ the machine goes into a computational loop, we always have $S \leq \log T$. Therefore, it follows from Bennett's simulation that each irreversible Turing machine using space S can be simulated by a reversible machine using space S^2 in polynomial time.

3 Results

Here we propose the quantitative study of exchanges of computing resources such as time and space for irreversibility which we believe will be relevant for the physics of future computation devices.

We analyse the advantage of adding limited irreversibility to an otherwise reversible simulation of conventional irreversible computations. This may be of some practical relevance for adiabatic computing. Our point of departure is the general method of Bennett [3] to reversibly simulate irreversible algorithms in the stylized form of a pebble game. While such reversible simulations incur little overhead in additional computation time, they use an unacceptable amount of additional memory space during the computation. We show that among all simulations which can be modelled by the pebble game, Bennett's simulation is optimal in that it uses the least auxilliary space for the greatest number of simulated steps.

- If S is the space used by the simulated irreversible computation, then the simulator uses nS space to simulate $(2^n-1)S$ steps of the simulated computation.

- We demonstrate that no reversible simulation which can be modeled by the pebble game can simulate that many steps using $(n-1)S$ space.

- On the other hand, we show that at the cost of a limited amount of erasure the simulation can be made more space efficient. We can save kS space in the reversible simulation at a cost of $(2^{k+2}-1)S$ irreversible bit erasures, for all k with $1 \leq k \leq n$.

4 Reversible Simulation

Analyzing the simulation method of [3] shows that it is essentially no better than the simple [1] simulation in terms of time versus irreversible erasure trade-off. Extra irreversible erasing can reduce the simulation time of the former method to $\Theta(T)$, but the 'simple' method has $\Theta(T)$ simulation time without irreversible erasures anyway, but at the cost of large space consumption. Therefore, it is crucial to decrease the extra space required for the pure reversible simulation without increasing time if possible, and in any case further reduce the extra space at the cost of limited numbers of irreversible erasures.

Since there is no better general reversible simulation of an irreversible computation known as the above one, and it seems likely that each proposed method must have similar history preserving features, analysis of this particular style of simulation may in fact give results with more general validity. We establish lower bounds on space use and upper bounds on space versus irreversible erasure trade-offs.

To analyse such trade-offs we use Bennett's brief suggestion in [3] that a reversible simulation can be modelled by the following 'reversible' pebble game. Let G be a linear list of nodes $\{1, 2, \ldots, T_G\}$. We define a *pebble game* on G as follows. The game proceeds in a discrete sequence of steps of a single *player*. There are n pebbles which can be put on nodes of G. At any time the set of pebbles is divided in pebbles on nodes of G and the remaining pebbles which are called *free* pebbles. At each step either an existing free pebble can be put on a node of G (and is thus removed from the free pebble pool) or be removed from a node of G (and is added to the free pebble pool). The rules of the game are as follows.

1. Initially G is unpebbled and there is a pool of free pebbles.
2. In each step the player can put either
   (a) a free pebble on node 1 or remove a pebble from node 1, or
   (b) for some node $i > 1$, put a free pebble on node i or remove a pebble from node i, provided node $i-1$ is pebbled at the time.
3. The player wins the game if he pebbles node T_G and subsequently removes all pebbles from G.

The maximum number n of pebbles which are simultaneously on G at any time in the game gives the space complexity nS of the simulation. If one deletes a pebble not following the above rules, then this means a block of bits of size S is erased irreversibly. The limitation to Bennett's simulation is in fact space, rather than time. When space is limited, we may not have enough place to store garbage, and these garbage bits will have to be irreversibly erased. We establish a tight lower bound for *any* strategy for the pebble game in order to obtain a space-irreversibility trade-off.

Lemma 1 *There is no winning strategy with n pebbles for $T_G \geq 2^n$.*

Proof. Fix any pebbling strategy for the player. To prove the lemma it suffices to show that the player cannot reach node $f(k) = 2^k$ using k pebbles, and also remove all the pebbles at the end, for $k := 1, 2, \ldots$. We proceed by induction.

Basis: $k = 1$. It is straightforward to establish $f(1) = 2$ cannot be reached with 1 pebble.

Induction: $k \rightarrow k+1$. Assume that $f(i) = 2^i$ cannot be reached with i pebbles, for $i = 1, \ldots, k$, has been established. Consider pebbling G using $k+1$ pebbles. Assume, that the player can pebble node $f(k)+1 = 2^k+1$ (otherwise the induction is finished).

Then, by the rules of the game, there must be a *least* step t such that for *all times* $t' > t$ there are pebbles on some nodes in $f(k)+1, f(k)+2, \ldots, T_G$. Among other things, this implies that at step $t+1$ node $f(k)+1$ is pebbled.

Partition the first $f(k)-2$ nodes of G into disjoint consecutive regions: starting with node 1, region L_i consists of the next block of $f(k-i)$ nodes, for $i = 1, \ldots, k-1$. That is, $L_i = \{\sum_{j=1}^{k-i+1} 2^{k-j} + 1, \ldots, \sum_{j=1}^{k-i} 2^{k-j}\}$. The regions $L_1, \ldots, L_{k-1}$ cover nodes $1, \ldots, f(k)-2$. Denote the remainder of G but for nodes $f(k)-1, f(k)$ by R, that is $R = G - \{f(k)-1, f(k)\} - \bigcup_{i=1}^{k-1} L_i = \{f(k)+1, f(k)+2, \ldots, T_G\}$.

Consider the game from step $t+1$ onwards. If there is always at least one pebble on nodes $1, \ldots, f(k)$, then by inductive assumption the player can pebble with one initial pebble on $f(k)+1$ and the remaining $k-1$ free pebbles at most $f(k)-1$ nodes and hence no further than node $2f(k)-1 = 2^{k+1}-1$, and the induction is finished.

Therefore, to possibly pebble node 2^{k+1} the player needs to remove all pebbles from nodes $1, \ldots, f(k)$ first. Because node $f(k)+1$ was pebbled at step $t+1$, we know that node $f(k)$ did have a pebble at that time according to the game rules. By assumption, from time $t+1$ there will henceforth always be a leading pebble in region R. Moreover, at time $t+1$ there is a pebble on node $f(k)$. To remove all the pebbles in range $1, \ldots, f(k)$, the following requirements have to be satisfied.

- From time $t+1$ onwards, there must always be a pebble at a strategic location in L_1 until the last remaining pebble in $G - (L_1 \cup R) = \{f(k-1)+1, \ldots, f(k)\}$ is removed. Otherwise with at most $k-1$ pebbles, the player cannot cross the unpebbled region L_1 (because $|L_1| = f(k-1)$) to reach and remove the *finally last remaining* pebble in the range $G - (L_1 \cup R)$. There are only $k-1$ pebbles available because from time $t+1$ on we have a pebble in region R, and at least one pebble in $H = G - (L_1 \cup R)$.
- From time $t+1$ onwards, there must always be a pebble at a strategic location in L_2 until the last remaining pebble in $G - (L_1 \cup L_2 \cup R) = \{f(k-1)+f(k-2)+1, \ldots, f(k)\}$ is removed. Otherwise, with at most $k-2$ pebbles, the player cannot cross the unpebbled region L_2 (because $|L_2| = f(k-2)$) to reach and remove the *finally last remaining* pebble in the range $G - (L_1 \cup L_2 \cup R)$. There are only $k-2$ pebbles available because from time $t+1$ on we have a pebble in region R, a pebble in L_1 (to help removing the last remaining pebble in L_2), and at least one pebble in $H = G - (L_1 \cup L_2 \cup R)$.

- By iteration of the argument, there must be a pebble in each region L_i at time $t+1$, for $i = 1,\ldots,k-1$.

But these requirements use up $k-1$ pebbles located in regions $L_1,\ldots,L_{k-1}$. None of these regions can become pebble-free before we free the pebble on node $f(k)$, that is, the kth pebble. The $(k+1)$st pebble is in region R forever after step $t+1$. Therefore, there is no pebble left to pebble node $f(k)-1$ which is not in $R\bigcup\{f(k)\}\bigcup_{i=1}^{k-1} L_i$. Hence it is impossible to remove all k pebbles from the first nodes $1,\ldots,f(k)$. Thus, leaving one pebble in region $\{1,\ldots,f(k)\}$ with at most k remaining pebbles, by inductive assumption, the player can pebble no farther than node $2f(k)-1$, which finishes the induction. $\Box$

Lemma 2 *There is a winning strategy with n pebbles for $T_G = 2^n - 1$.*

Proof. Bennett's simulation [3] is a winning strategy. We describe his game strategy as the pebble game $G = \{1,\ldots,T_G\}$, recursively. Let $I_k = I_{k-1}i_{k-1}I_{k-2}i_{k-2}\ldots I_1 i_1 I_0 i_0$ where I_j is a sequence of 2^j-1 consecutive locations in G, and i_j is the node directly following I_j, for $j=0,1,\ldots,k-1$. Note that $|I_0| = 0$.

Let $F(k,I_k)$ be the program to pebble an initially pebble-free interval I_k of length 2^k-1 of G, starting with k free pebbles and a pebble-free I_k and ending with k pebbles on I_k including one pebble on the last node of I_k.

Let $F^{-1}(k,I_k)$ be the program starting with the end configuration of $F(k,I_k)$ and executing the operation sequence of $F(k,I_k)$ in reverse, each operation replaced by its inverse which undoes what the original operation did, ending with $F(k,I_k)$'s initial configuration. We give the precise procedure in self-explanatory pseudo PASCAL.

Procedure $F(k,I_k)$:
for $i := 1,2,\ldots,k$:
    $F(k-i, I_{k-i})$;
    put pebble on node i_{k-i} ;
    $F^{-1}(k-i, I_{k-i})$

Procedure $F^{-1}(k,I_k)$:
for $i := k,k-1,\ldots,1$:
    $F^{-1}(k-i, I_{k-i})$;
    remove pebble on node i_{k-i} ;
    $F(k-i, I_{k-i})$

Note that this way both $F(0,I_0)$ and $F^{-1}(0,I_0)$ are 'skip' operations which don't change anything. The size T_G of a pebble game which is won using this strategy using n pebbles is $|I_n| = 2^n - 1$. Moreover, if $F(k,I_k)$ takes $t(k)$ steps we find $t(k) = 2t(k-1) + \cdots + f(1) + k - 1$. Then, $t(k) = 3t(k-1) - 1$. That is, the number of steps T'_G of a winning play of a pebble game of size $T_G = 2^n - 1$ is $T'_G \approx 3^n$, that is, $T'_G \approx T_G^{\log 3}$. $\Box$

The simulation given in [3] follows the rules of the pebble game of length $T_G = 2^n - 1$ with n pebbles above. A winning strategy for a game of length T_G using n pebbles corresponds with reversibly simulating T_G segments of S steps of an irreversible computation using S space such that the reversible simulator uses $T' \approx ST'_G \approx ST_G^{\log 3}$ steps and total space $S' = nS$. The space S' corresponds to the maximal number of pebbles on G at any time during the game. The placement or removal of a pebble in the game corresponds to the reversible copying or reversible cancelation of a 'checkpoint' consisting of the entire instantaneous description of size S (work tape contents, location of heads, state of finite control) of the simulated irreversible machine. The total time $T_G S$ used by the irreversible computation is broken up in segments of size S so that the reversible copying and canceling of a checkpoints takes about the same number of steps as the computation segments in between checkpoints. [1]

We can now formulate a trade-off between space used by a polynomial time reversible computation and irreversible erasures. First we show that allowing a limited amount of erasure in an otherwise reversible computation means that we can get by with less work space. Therefore, we define an *m-erasure* pebble game as the pebble game above but with the additional rule

- In at most m steps the player can remove a pebble from any node $i > 1$ without node $i-1$ being pebbled at the time.

An m-erasure pebble game corresponds with an otherwise reversible computation using mS irreversible bit erasures, where S is the space used by the irreversible computation being simulated.

Lemma 3 *There is a winning strategy with n pebbles and $2m-1$ erasures for pebble games G with $T_G = m2^{n-1}$, for all $m \geq 1$.*

Proof. The strategy is to advance in blocks of size $2^{n-1}-1$ using $n-1$ pebbles without erasures (as in Lemma 2), put the nth pebble in front, and invert the advancement process to free all the pebbles in the block. The last remaining pebble has no predecessor and needs to be irreversibly erased except in the initial block. The initial pebble is put in front of the lastly placed nth pebble which, having done its duty as springboard for this block, is subsequently irreversibly erased. Therefore, the advancement of each block requires two erasures, except the first block which requires one, yielding a total of

[1] In addition to the rules of the pebble game there is a permanently pebbled initial node so that the simulation actually uses $n+1$ pebbles for a pebble game with n pebbles of length T_G+1. The simulation uses $n+1 = S'/S$ pebbles for a simulated number of $S(T_G+1)$ steps of the irreversible computation.

$2m-1$ erasures. Let $G=\{1,2,\ldots,T_G\}$ be segmented as $B_1 b_1 \ldots B_m b_m$, where each B_i is a copy of interval I_{n-1} above and b_i is the node following B_i, for $i=1,\ldots,m$. Hence, $T_G = m2^{n-1}$. We give the precise procedure in self-explanatory pseudo PASCAL using the procedures given in the proof of Lemma 2.

Procedure $A(n,m,G)$:
for $i := 1,2,\ldots,m$:
  $F(n-1,B_i)$;
  erase pebble on node b_{i-1} ;
  put pebble on node b_i ;
  $F^{-1}(n-1,B_i)$ (removal of pebble from first node of B_i is an erasure)

The simulation time T'_G is $T'_G \approx 2m \cdot 3^{n-1}+2 \approx 2m(T_G/m)^{\log 3} = 2m^{1-\log 3} T_G^{\log 3}$ for $T_G = m2^{n-1}$. □

Theorem 1 (Space-Irreversibility Trade-off) *(i) Pebble games G of size 2^n-1 can be won using n pebbles but not using $n-1$ pebbles.*

(ii) If G is a pebble game with a winning strategy using n pebbles without erasures, then there is also a winning strategy for G using E erasures and $n-\log(E+1)$ pebbles (for E is an odd integer at least 1).

Proof. (i) By Lemmas 2, 1.

(ii) By (i), $T_G = 2^n - 1$ is the maximum length of a pebble game G for which there is a winning strategy using n pebbles and no erasures. By Lemma 3, we can pebble a game G of length $T_G = m2^{n-\log m} = 2^n$ using $n+1-\log m$ pebbles and $2m-1$ erasures. □

We analyse the consequences of Theorem 1. It is convenient to consider the special sequence of values $E := 2^{k+2}-1$ for $k := 0,1,\ldots$. Let G be Bennett's pebble game of Lemma 2 of length $T_G = 2^n-1$. It can be won using n pebbles without erasures, or using $n-k$ pebbles plus $2^{k+2}-1$ erasures (which gives a gain over not erasing as in Lemma 2 only for $k \geq 1$), but not using $n-1$ pebbles.

Therefore, we can exchange space use for irreversible erasures. Such a trade-off can be used to reduce the excessive space requirements of the reversible simulation. The correspondence between the erasure pebble game and the otherwise reversible computations using irreversible erasures that if the pebble game uses $n-k$ pebbles and $2^{k+2}-1$ erasures, then the otherwise reversible computation uses $(n-k)S$ space and erases $(2^{k+2}-1)S$ bits irreversibly.

Therefore, a reversible simulation of an irreversible computation of length $T=(2^n-1)S$ can be done using nS space using $(T/S)^{\log 3}S$ time, but is impossible using $(n-1)S$ space. It can also be performed using $(n-k)S$ space, $(2^{k+2}-1)S$ irreversible bit erasures and $2^{(k+1)(1-\log 3)+1}(T/S)^{\log 3}S$ time. In the extreme case we use no space to store the history and erase about $4T$ bits. This corresponds to the fact that an irreversible computation may overwrite its scanned symbol irreversibly at each step.

Definition 1 Consider a simulation using S' storage space and T' time which computes $y=\langle x, f(x)\rangle$ from x in order to simulate an irreversible computation using S storage space and T time which computes $f(x)$ from x. The *irreversible simulation cost* $B^{S'}(x,y)$ of the simulation is the number of irreversibly erased bits in the simulation (with the parameters S,T,T' understood).

If the irreversible simulated computation from x to $f(x)$ uses T steps, then for $S'=nS$ and $n=\log(T/S)$ we have above treated the most space parsimonious simulation which yields $B^{S'}(x,y)=0$, with $y=\langle x,f(x)\rangle$.

Corollary 1 (Space-Irreversibility Trade-off) *Simulating a $T=(2^n-1)S$ step irreversible computation from x to $f(x)$ using S space by a computation from x to $y=\langle x,f(x)\rangle$, the irreversible simulation cost satisfies:*

(i) $B^{(n-k)S}(x,y) \leq B^{nS}(x,y) + (2^{k+2}-1)S$, for $n \geq k \geq 1$.

(ii) $B^{(n-1)S}(x,y) > B^{nS}(x,y)$, for $n \geq 1$.

For the most space parsimonious simulation with $n=\log(T/S)$ this means that

$$B^{S(\log(T/S)-k)}(x,y) \leq B^{S\log(T/S)}(x,y) + (2^{k+2}-1)S.$$

We *conjecture* that *all* reversible simulations of an irreversible computation can essentially be represented as the pebble game defined above, and that consequently the lower bound of Lemma 1 applies to all reversible simulations of irreversible computations. If this conjecture is true then the trade-offs above turn into a space-irreversibility hierarchy for polynomial time computations.

References

[1] C.H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, 17:525–532, 1973.

[2] C.H. Bennett. The thermodynamics of computation—a review. *Int. J. Theoret. Phys.*, 21(1982), 905-940.

[3] C.H. Bennett. Time-space trade-offs for reversible computation. *SIAM J. Comput.*, 18(1989), 766-776.

[4] C.H. Bennett, P. Gács, M. Li, P.M.B. Vitányi and W.H Zurek, Thermodynamics of computation and information distance *Proc. 25th ACM Symp. Theory of Computation.* ACM Press, 1993, 21-30.

[5] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Royal Society London.* Vol. A400(1985), 97-117.

[6] R. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(1986), 507-531. (Originally published in *Optics News*, February 1985.)

[7] E. Fredkin and T. Toffoli. Conservative logic. *Int. J. Theoret. Phys.*, 21(1982),219-253.

[8] R.W. Keyes, *IBM J. Res. Dev.*, 32(1988), 24-28.

[9] R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Develop.*, 5:183–191, 1961.

[10] R. Landauer, Dissipation and noise immunity in computation and communication, *Nature*, 335(1988), 779-784.

[11] Y. Lecerf, Machines de Turing réversibles. Récursive insolubilité en $n \in N$ de l'équation $u = \theta^n$, où θ est un "isomorphisme de codes", *Comptes Rendus*, 257(1963), 2597-2600.

[12] R.Y. Levine and A.T. Sherman, A note on Bennett's time-space trade-off for reversible computation, *SIAM J. Comput.*, 19:4(1990), 673-677.

[13] M. Li and P.M.B. Vitányi, Reversibility and adiabatic computation: trading time and space for energy, *Proc. Royal Society of London, Series A*, 1996, to appear.

[14] R.C. Merkle, Reversible electronic logic using switches, *Nanotechnology*, 4(1993), 21-40.

[15] Proc. 1981 Physics and Computation Workshop. *Int. J. Theoret. Phys.*, 21(1982). *Proc. IEEE 1992 Physics and Computation Workshop.* IEEE Computer Society Press, 1992. *Proc. IEEE 1994 Physics and Computation Workshop.* IEEE Computer Society Press, 1994.

[16] J. von Neumann. *Theory of Self-Reproducing Automata.* A.W. Burks, Ed., Univ. Illinois Press, Urbana, 1966.