

Reversible Space Equals Deterministic Space

(Extended Abstract)

Klaus-Jörn Lange *
Univ. Tübingen

Pierre McKenzie †
Univ. de Montréal

Alain Tapp †
Univ. de Montréal

Abstract

This paper describes the simulation of an $S(n)$ space-bounded deterministic Turing machine by a reversible Turing machine operating in space $S(n)$. It thus answers a question posed by Bennett in 1989 and refutes the conjecture, made by Li and Vitanyi in 1996, that any reversible simulation of an irreversible computation must obey Bennett’s reversible pebble game rules.

1 Introduction

A Turing machine M is reversible iff the infinite graph of all configurations of M has indegree and outdegree one. Interest in reversibility arose at first in connection with the thermodynamics of computation, following Landauer’s demonstration in 1961 that, contrary to earlier intuition (see [Ne66]), physical laws do not preclude using an arbitrarily small amount of energy to perform logically reversible computing steps [La61]. More recently, renewed interest in the notion of reversibility was sparked by the prospect of quantum computers, whose observation-free computational steps are intrinsically reversible [De85, Sh94, Br95].

Early strategies to make a Turing machine reversible were terribly wasteful in terms of space: Lecerf’s method [Le63], rediscovered by Bennett [Be73], required space T to simulate a $T(n)$ -time $S(n)$ -space machine reversibly in time $O(T)$. Bennett then greatly improved on this by reducing the space to $O(S \log T)$ at the expense of an increase in time to $T^{1+\epsilon}$ [Be89]. Levine and Sherman refined the analysis of Bennett’s algorithm and characterized the tradeoff between time and space even more precisely [LeSh90].

Bennett questioned [Be89] whether the reversible simulation of an irreversible computation necessarily incurs

a non constant factor increase in space usage. Bennett offered both a pebble game formalizing his intuition that the space increase is unavoidable, and a possible source of contrary evidence arising from the surprisingly width-efficient reversible simulations of irreversible circuits. At the 1996 IEEE Computational Complexity conference, Li and Vitanyi took up Bennett’s suggestion and performed an in-depth analysis of Bennett’s pebble game [LiVi96a, LiVi96b]. They proved that any strategy obeying Bennett’s game rules indeed requires the extra $\Omega(\log T)$ multiplicative space factor, and they exhibited a trade-off between the need for extra space and the amount of irreversibility (in the form of irreversibly erased bits) which might be tolerated from the simulation. Li and Vitanyi then conjectured that all reversible simulations of an irreversible computation obey Bennett’s pebble game rules, hence that all such simulations require $\Omega(S \log T)$ space.

Here we refute Li and Vitanyi’s conjecture: Using a strategy which of course does not obey Bennett’s game rules, we reversibly simulate irreversible space S computations in space S . Our strategy is the extreme opposite of Lecerf’s “space-hungry” method: While we scrupulously preserve space, time becomes exponential in S . We offer two reasons to justify interest in such a “time-hungry” method. First, the new method is proof that Bennett’s game rules do not capture all strategies, leaving open the possibility that, unobstructed by Li and Vitanyi’s lower bounds, more efficient reversible simulations of irreversible computation should exist. Secondly, for problems in $DSPACE(\log)$, Bennett’s simulation uses space (\log^2) , while our new method uses only $\log n$ space and polynomial time. This could be interesting in the context of quantum computing, for then, space seems to be more of a concern than time. (Storing many entangled q-bits seems more difficult than applying basic unitary transformations.)

Section 2 in this extended abstract contains preliminaries and discusses the notion of reversibility. Section 3 presents our main result, first in detail in the context of linear space, and then in the more general context of any space bound. Section 4 concludes with a discussion.

*Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany.

lange@informatik.uni-tuebingen.de

†Dép. d’informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128 succursale Centre-Ville, Montréal, H3C 3J7 Canada. Work supported by NSERC of Canada and by FCAR du Québec. {mckenzie, tappa}@iro.umontreal.ca

2 Preliminaries

We assume familiarity with basic notions of complexity theory such as can be found in [HoU179]. We refer to the finite set of states of a Turing machine as to its set of “local states”. We assume that, when computing a function f on input x , a deterministic Turing machine halts in a unique final configuration with $f(x)$ alone on its tape.

2.1 Reversible Turing machines

Definition. A Turing machine is *reversible* iff the infinite graph of M 's configurations, in which an arc (C, C') indicates that M can go from configuration C to configuration C' in a single transition, has indegree and outdegree at most one. ■

Following Bennett [Be89], we impose the following restrictions on the transitions of a Turing machine and claim that these are sufficient to imply reversibility.

A transition is either *moving* (i.e. the tape head moves), or *stationary*. Each moving transition must be oblivious (i.e. it depends only on the local state and not on the symbol read from the tape). We also require that no pair of transitions *intersect*, in the following sense. We say that two stationary transitions intersect if their execution leaves the machine in the same local state with the same symbol under the tape head. We say that two moving transitions intersect if they lead from different local states to the same local state. Finally, we say that a stationary transition and a moving transition intersect if they lead to the same local state.

We can extend these syntactic restrictions on the transitions of a machine to the case of a multi-tape machine, but these become trickier to describe. Intuitively however, we only require that the local state and symbols under the heads uniquely determine the most recent transition.

2.2 Computing functions reversibly

Unlike in the deterministic computation of a function f on input x , the tape content beyond the value of $f(x)$ at the end of a reversible computation is a concern. In any reversible computation model (even other than Turing machines), one must know the content of a memory cell in order to use this cell in some useful computation later. This is because *erasing is a fundamentally irreversible action*. Since spoiled memory is no longer useful, it is important that the memory be restored to its initial state at the end of a reversible computation.

Two notions of memory “cleanliness” have been studied in the literature: These correspond to input-saving and to input-erasing Turing machines [Be89]. In an input-saving computation, the final tape content is $\#x\#f(x)\#$. In an input-erasing computation, only $f(x)$ remains on the tape.

The latter notion was only considered useful when f is injective.

In this paper, we observe that reversibly computing an arbitrary function f in an input-erasing way is possible, *without* first embedding f in an injective function. We thus adopt the input-erasing mode of computing f , even when f is arbitrary. An interesting feature of our reversible machine on input x is that the machine will cycle through all the inverse images of $f(x)$: the lack of injectivity of f only translates into the fact that the machine cannot tell which of these inverse images was its actual input x .

3 Main result

3.1 Linear space

Theorem 3.1 *Any bijective function computed in deterministic space precisely equal to the input length can be computed reversibly in the same space.*

Proof. We begin by describing the idea intuitively. As with Bennett’s reversible simulations of irreversible computations, our high level strategy is simple, but care is needed when filling in the details because the syntactic conditions required at the transition function level can be tricky to enforce and verify.

The main idea for simulating a machine without using more space is to reversibly cycle through the configuration tree of the machine. For our purposes, it will suffice to consider the configuration tree as an undirected tree, in which each edge is duplicated. We will then in effect perform an Euler tour of the resulting tree. A similar technique was used by Sipser [Si80] to simulate an $S(n)$ space-bounded Turing machine *accepting* a language Y (i.e. with no bounds on space when the input $w \notin Y$) by a Turing machine *deciding* Y in space $S(n)$.

Let $G_\infty(M)$ be the infinite configuration graph of a single worktape linear space deterministic Turing machine M . Write $C_0(w) = \# \binom{w_1}{q_{0,M}} w_2 w_3 \cdots w_n \#$ for the initial configuration of M on input $w \in \Sigma^*$, that is, the local initial state of M is $q_{0,M}$ and M 's input is placed within markers. Consider the weakly connected component¹ G_w of $G_\infty(M)$ which contains $C_0(w)$ and which contains no configuration of M using more than linear space. Without loss of generality we make the following assumptions concerning G_w :

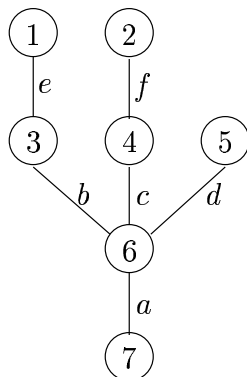
- The graph G_w is finite because M is space bounded. (We assume without loss of generality that the machine *never* moves outside the space markers.)

¹Induced by the connected component in the associated undirected graph.

- The indegree of $C_0(w)$ is zero. This is accomplished by preventing the reversible machine M from ever transiting back into its initial local state.
- The outdegree of any final configuration of M is zero.
- The head positions in two adjacent configurations of M in G_w differ by one. This is done by disallowing transitions which leave the head stationary.
- G_w contains no directed cycle. In other words, we assume that M 's computation on w necessarily ends in a final configuration. Note that some linear space configurations of M could participate in directed cycles, but that these could not be weakly connected to $C_0(w)$.
- G_w is acyclic even when its arcs are replaced by (undirected) edges. This follows from our previous assumption because M is deterministic and hence the directed G_w has outdegree one.

We think of G_w as a tree with a final configuration as root at the bottom, and with $C_0(w)$ as one of its leaves on top. Observe that the irreversibility of M translates precisely into the fact that G_w is a tree and not simply a line. Our idea is to design a reversible Turing machine R whose configurations on input w will form a line, which will include, among other configurations, a representation of sufficiently many configurations of G_w to reach the final configuration of M in G_w . An obvious idea to design such an R is to have R perform an Euler tour of G_w (more precisely, of the undirected tree obtained from G_w by replacing each directed arc by two undirected edges). The difficulty is to ensure reversibility of R at the transition function level.

To simplify the presentation of the machine R , we will make use of a construction developed in [CoMc87] for the purpose of showing that permutation representation problems and tree traversal problems are logspace-complete. We now recall this construction. Denote by T the following tree with nodes $\{1, 2, 3, 4, 5, 6, 7\}$ and with edges $\{a, b, c, d, e, f\}$:



Let $\Omega_T = \{e_1, e_3, f_2, f_4, b_3, b_6, c_4, c_6, d_5, d_6, a_6, a_7\}$ be the set of “edge ends”. Note that $|\Omega_T|$ equals twice the number of edges in the tree. Two permutations on Ω_T will be constructed. We first construct the “ROTATION permutation” π_T . To define π_T , fix, locally at each node N in the tree, an arbitrary ordering of the “edge ends incident with N ”. Then let π_T be the product, over each node N , of a cycle permuting the edge ends incident with N according to this ordering. In our example,

$$\pi_T = (b_3 e_3)(c_4 f_4)(a_6 b_6 c_6 d_6),$$

where we have chosen the alphabetical ordering of the incident edges as the local ordering at each node. We then construct the “SWAP permutation” σ_T , defined simply as a product of as many transpositions as there are edges in T :

$$\sigma_T = (e_1 e_3)(f_2 f_4)(b_3 b_6)(c_4 c_6)(d_5 d_6)(a_6 a_7).$$

Our interest lies in the permutation $\pi_T \sigma_T = (e_1 e_3 b_6 c_4 f_2 f_4 c_6 d_5 d_6 a_7 a_6 b_3)$. Although we have described the construction of π_T and σ_T in the context of a simple example, the construction as it applies to a general tree T should be clear. Among the properties of $\pi_T \sigma_T$ proved in [CoMc87], we will only make use of the property that $\pi_T \sigma_T$ is a single cycle which includes precisely all the elements in Ω_T . Our reversible Turing machine R will simulate the computation of $\pi_{G_w} \sigma_{G_w}$. In this way, R will traverse the computation tree of M on input w , by traversing the cycle $\pi_{G_w} \sigma_{G_w}$ (reversibly, $\pi_{G_w} \sigma_{G_w}$ being a permutation of Ω_{G_w}). Since the unique final configuration of M present in G_w is necessarily the final configuration reached by M on input w , R can reach the same final decision as that reached by M on input w .

In terms of the above construction, we now complete the description of the reversible machine R simulating the reversible machine M computing a bijective function $f : \Sigma^* \rightarrow \Sigma^*$ in linear space. We assume that R 's input tape is also R 's (only) worktape. The initial configuration of R will have $\#w\#$ on its tape, and the final configuration of R will have $\#f(w)\#$ on its tape. We assume for ease of presentation that the length of w is at least 1.

Recall that Ω_{G_w} is the set of all “edges ends” in the tree G_w (which we now view as undirected). We view each configuration of M as a string representing M 's tape content, with the local state q and head position of M recorded within this string in an amalgamated symbol $\binom{q}{h}$. Let $e_{C,C'}$ denote the edge end incident with C of the edge (C, C') . The string representing C differs from the string representing C' in at most the three adjacent positions centered at the amalgamated symbol within C : we call these three positions the active positions in C and C' with respect to $e_{C,C'}$. The reversible machine R will represent $e_{C,C'}$ internally as follows:

- the tape of R will be identical to the tape of M in configuration C ,
- the head position of R will be the position of the head of M in C , and
- the finite control of R will store the local state of M in C and the three active symbols xyz in C' (where either x or z is the amalgamed symbol representing M 's head position and local state in C').

The reversible algorithm consists in iterating an ‘‘Euler stage’’, which we now describe. The very simple setup and termination stages will be discussed afterwards. We adopt the following conventions when describing the transitions of R :

- when we omit the tape symbol in the range of a transition, we mean that the transition leaves the tape content unmodified, and
- each local state of R having the syntactic form q^{\rightarrow} (resp. q^{\leftarrow}) indicates the presence of the right (resp. left) head motion transition $q^{\rightarrow} \rightarrow q, +1$ (resp. $q^{\leftarrow} \rightarrow q, -1$): These transitions move the head and switch local state independently from the tape content.
- as an illustrative example, the name chosen for the state $ab, q \text{ ROTATE}_{3, xyz}$ found in the ROTATION substage below is intended to convey the information that:
 - q is the local state of the machine M in the configuration of M whose tape content is the current content of R 's tape,
 - ab is some information read off from R 's tape and stored in the finite control of R temporarily,
 - 3 identifies this particular ROTATION stage local state, and
 - xyz are the three active symbols of the configuration of M identifying the end, of the current ‘‘edge end’’ represented by R , which is stored in R 's finite control.

EULER STAGE: At the beginning of an Euler stage, R internally represents some $e_{C, C'} \in \Omega_{G_w}$. The goal of the stage is to replace $e_{C, C'}$ with $\sigma_{G_w}(\pi_{G_w}(e_{C, C'}))$. This is done in two substages.

ROTATION substage: From $e_{C, C'}$ to $\pi_{G_w}(e_{C, C'})$. In the ROTATION substage, R must replace $e_{C, C'}$ with the edge end $e_{C, C''}$ coming next in the local ordering of the edge ends incident with C , where C'' is not necessarily distinct from C' . Hence,

- the tape content and head position of R are unaffected by this replacement,

- the three active positions (but generally not their contents) in C' and C'' are the same,
- C' and C'' are identical outside the three active positions,
- given the contents of the three active positions in C (where we view the contents of the active positions as including the local state and head position, although for example R internally represents the local state of M in C separately), the π_{G_w} -induced function $f_{M, C}$ mapping the contents of the three active positions in C' to the contents of the three active symbols in C'' is well defined, and
- because π_{G_w} is a permutation, $f_{M, C}$ is bijective.

Hence, the ROTATION substage consists of reading the contents of the three active positions from R 's tape, and then modifying the active symbols in the finite control. To justify reversibility at the transition function level, we propose the following transitions:

$$\begin{aligned}
q \text{ EULER}_{xyz}, b &\rightarrow b, q \text{ ROTATE}_{1, xyz}^{\leftarrow} \\
b, q \text{ ROTATE}_{1, xyz}, a &\rightarrow ab, q \text{ ROTATE}_{2, xyz}^{\rightarrow} \\
ab, q \text{ ROTATE}_{2, xyz}, b &\rightarrow ab, q \text{ ROTATE}_{3, xyz}^{\rightarrow} \\
ab, q \text{ ROTATE}_{3, xyz}, c &\rightarrow ab, q \text{ ROTATE}_{4, x'y'z'}^{\leftarrow} \\
ab, q \text{ ROTATE}_{4, x'y'z'}, b &\rightarrow a, q \text{ ROTATE}_{5, x'y'z'}^{\leftarrow} \\
a, q \text{ ROTATE}_{5, x'y'z'}, a &\rightarrow q \text{ SWAP}_{1, x'y'z'}^{\rightarrow}
\end{aligned}$$

where xyz are any legal combination of three active symbols in C' , q is any local state of M in C , $x'y'z'$ are the three active symbols in C'' given by $f_{M, C}(xyz)$, and a, b and c are any tape alphabet symbols of M . Recall that, by convention, six head motion transition schemata are also implicitly defined above.

SWAP substage: From $\pi_{G_w}(e_{C, C'})$ to $\sigma_{G_w}(\pi_{G_w}(e_{C, C'}))$.

In the SWAP substage, R must replace $e_{C_1, C_2} = \pi_{G_w}(e_{C, C'})$ with $\sigma_{G_w}(e_{C_1, C_2})$. By definition of σ_{G_w} , this simply amounts to interchanging the internal representations of C_1 and C_2 . To do this, it suffices to interchange the three active symbols of C_1 with the three active symbols of C_2 . Such an interchange is obviously reversible globally. At the transition function level, the details must deal with the chosen internal representation of e_{C_1, C_2} . According to the head position in C_2 we will distinguish two cases:

$$(C_1, C_2) = (a \binom{q}{b} c, \binom{q'}{x} y c)$$

$$\begin{aligned}
q \text{ SWAP}_{1, \binom{q'}{x} y c}, b &\rightarrow ? \binom{q}{b} \text{ SWAP}_{2, \binom{q'}{x} y c}^{\rightarrow}, b \\
? \binom{q}{b} \text{ SWAP}_{2, \binom{q'}{x} y c}, c &\rightarrow ? \binom{q}{b} \text{ SWAP}_{3, \binom{q'}{x} y}^{\leftarrow}, c \\
? \binom{q}{b} \text{ SWAP}_{3, \binom{q'}{x} y}, b &\rightarrow ? \binom{q}{b} \text{ SWAP}_{4, \binom{q'}{x}}^{\leftarrow}, y \\
? \binom{q}{b} \text{ SWAP}_{4, \binom{q'}{x}}, a &\rightarrow a \binom{q}{b} \text{ SWAP}_{5, q'}^{\leftarrow}, x \\
a \binom{q}{b} \text{ SWAP}_{5, q'}, d &\rightarrow q' \text{ EULER}_{d a \binom{q}{b}}^{\rightarrow}, d
\end{aligned}$$

$$(C_1, C_2) = (a \binom{q}{b} c, ay \binom{q'}{z})$$

$$\begin{aligned} q \text{SWAP}_{1, ay \binom{q'}{z}}, b &\rightarrow \binom{q}{b} ? \text{SWAP}_{2, ay \binom{q'}{z}}^{\leftarrow}, b \\ \binom{q}{b} ? \text{SWAP}_{2, ay \binom{q'}{z}}, a &\rightarrow \binom{q}{b} ? \text{SWAP}_{3, y \binom{q'}{z}}^{\rightarrow}, a \\ \binom{q}{b} ? \text{SWAP}_{3, y \binom{q'}{z}}, b &\rightarrow \binom{q}{b} ? \text{SWAP}_{4, \binom{q'}{z}}^{\rightarrow}, y \\ \binom{q}{b} ? \text{SWAP}_{4, \binom{q'}{z}}, c &\rightarrow \binom{q}{b} c \text{SWAP}_{5, q'}^{\rightarrow}, z \\ \binom{q}{b} c \text{SWAP}_{5, q'}, d &\rightarrow q' \text{EULER}_{5, \binom{q}{b} cd}^{\leftarrow}, d \end{aligned}$$

SETUP STAGE: The only setup required consists in having three special symbols in place of xyz in the initial R configuration $q_{0, M} \text{EULER}_{\text{\$}\text{\$}\text{\$}}$. The function $f_{M, C}$ used in defining the first ROTATION should be the identity function, and the SWAP stage must have the effect of processing a virtual transition from a virtual configuration of M into M 's initial configuration.

TERMINATION STAGE: Euler stages are repeated until the termination stage is triggered. This occurs when an Euler stage finds that the local state of M is M 's unique final local state. In that case, R halts and the desired output $f(w)$ is the sole content of R 's tape.

This completes the proof of Theorem 3.1. ■

Corollary 3.2 *Any function computable in space n can be computed in space n with a reversible TM.*

Proof. This follows directly from the proof of Theorem 3.1 because the latter does not appeal to the injectivity of f . We make the observation that the lack of injectivity simply results in the tree G_w containing more than one legal initial configuration of M : one such initial configuration corresponds to each element in $f^{-1}(f(w))$. ■

Remark. The reversible simulation in Corollary 3.2 does not keep track of the initial configurations along the Euler tour of G_w , so that, if the machine is reversed, it cannot distinguish between these. However, the Euler tour of G_w canonically orders these initial configurations. It is possible to modify our simulation in order to count the number of initial configurations encountered (at the expense of doubling the space used). If we allowed writing this count on the output tape together with $f(w)$ (in effect rendering f injective in a “minimal” way), then the reversible machine R could retrieve w from its output when run in reverse. ■

3.2 Non-linear space

Theorem 3.3 *Any function f computable irreversibly in space $S(n)$ can be computed by a reversible Turing machine in the same space.*

Proof. The simulation described in proving Theorem 3.1 can easily be generalized to the case of an irreversible multihead Turing machine where the space is delimited on every tape by special markers. This applies in particular to

a machine with a read-only input tape and a work tape. Informally, when the space bound is not linear or is a priori unknown, we will successively cycle through the configuration trees G_w^k , $k \geq 1$, for k the amount of space on the work tape between the markers. The work tape is initially set to $\text{\$}\text{\$}$, and whenever the “Euler tour” returns to the special initial configuration, the space is incremented and another tour is initiated. As soon as $k = S(n)$ is reached the simulation will find the answer and stop. Details will appear in the final version of this paper. ■

Corollary 3.4 *Any language in $DSPACE(\log n)$ is accepted by a reversible Turing machine operating in logarithmic space (and polynomial time).* ■

4 Discussion

In this paper we showed determinism to coincide with reversibility for space. It is interesting to compare this with the equivalence of deterministic time and reversible time, which is simply shown by storing the whole computational history on a separate write-only tape [Le63, Be89]. It is remarkable that this is the very same construction which proves the equivalence of nondeterministic time and symmetric² time [LePa82]. This duality of the pairs nondeterminism versus symmetry and determinism versus reversibility is tied to the question of whether transitions can be regarded as directed or as undirected: This makes no difference if the indegree of every configuration is at most one.

The duality mentioned above and our new results point to the question of the relationship between nondeterministic space and symmetric space. In this case however, some recent results like the inclusion of symmetric logspace in parity logspace, SC^2 , or $DSPACE(\log^{1.5} n)$ [KaWi93, Ni92, NiSzWi92] suggest that the computational power of nondeterministic space and symmetric space differ.

Acknowledgments Special thanks go to Michael Frank, who pointed out that the reversible constructibility condition imposed on $S(n)$ in a former version of Theorem 3.3 was not required. We also thank Richard Beigel, and the anonymous Complexity Theory conference referees, for their helpful comments.

References

- [Be73] C. BENNETT, Logical reversibility of computation, *IBM J. Res. Develop.* **17** (1973), pp. 525–532.
- [Be89] C. BENNETT, Time/space trade-offs for reversible computation, *SIAM J. Comput.* **81**:4 (1989), pp. 766–776.

²In a symmetric machine each transition can be done in both directions: forward and backward.

- [Br95] G. BRASSARD, A quantum jump in Computer Science, in *Computer Science Today*, ed. J. van Leeuwen, Lecture Notes in Computer Science, Volume 1000 (special anniversary volume), Springer-Verlag (1995), pp. 1–14.
- [Ne66] J. VON NEUMANN, *Theory of self-reproducing automata*, A.W. Burks, Ed., Univ. Illinois Press, Urbana (1966).
- [Co85] S.A. COOK, A taxonomy of problems with fast parallel algorithms, *Information and Control* **64** (1985), pp. 2–22.
- [CoMc87] S.A. COOK AND P. MCKENZIE, Problems complete for deterministic logarithmic space, *J. Algorithms* **8** (1987), pp. 385–394.
- [De85] D. DEUTSCH, Quantum theory, the Church-Turing principle and the universal quantum computer, *Proc. Royal Soc. London, Series A* **400** (1985), pp. 97–117.
- [HoUl79] J.E. HOPCROFT AND J.D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley (1979).
- [KaWi93] M. KARCHMER AND A. WIGDERSON, On span programs, *Proc. 8th IEEE Structure in Complexity Theory Conference* (1993), pp. 102–111.
- [La61] R. LANDAUER, Irreversibility and heat generation in the computing process, *IBM J. Res. Develop.* **5** (1961), pp. 183–191.
- [Le63] Y. LECERF, Machines de Turing réversibles. Insolubilité récursive en $n \in \mathbb{N}$ de l'équation $u = \theta^n$, où θ est un "isomorphisme de codes", *Comptes Rendus* **257** (1963), pp. 2597–2600.
- [LeSh90] R. LEVINE AND A. SHERMAN, A note on Bennett's time-space trade-off for reversible computation, *SIAM J. Comput.* **19:4** (1990), pp. 673–677.
- [LePa82] P. LEWIS AND C. PAPADIMITRIOU, Symmetric space-bounded computation, *Theoret. Comput. Sci.* **19** (1982), 161–187.
- [LiVi96a] M. LI AND P. VITANYI, Reversible simulation of irreversible computation, *Proc. 11th IEEE Conference on Computational Complexity* (1996), pp. 301–306.
- [LiVi96b] M. LI AND P. VITANYI, Reversibility and adiabatic computation: trading time and space for energy, *Proc. Royal Soc. London, Series A* **452** (1996), pp. 769–789.
- [Ni92] N. NISAN, $RL \subseteq SL$, *Proc. 24th ACM Symp. on Theory of Computing* (1992), pp. 619–623.
- [NiSzWi92] N. NISAN, E. SZEMEREDI, AND A. WIGDERSON, Undirected connectivity in $O(\log^{1.5} n)$ space, *Proc. 34th IEEE Symp. on Foundations of Computer Science* (1992), pp. 24–29.
- [Sh94] P. SHOR, Algorithms for quantum computation: Discrete log and factoring, *Proc. 35th IEEE Symp. Foundations of Comp. Sci.* (1994), pp. 124–134.
- [Si80] MICHAEL SIPSER, Halting Space-Bounded Computations, *Theoretical Computer Science* **10** (1990), pp. 335–338.