SIAM J. DISC. MATH.
Vol. 3, No. 2, pp. 241–254, May 1990

REVERSIBLE TURING MACHINES AND POLYNOMIAL TIME REVERSIBLY COMPUTABLE FUNCTIONS*

G. JACOPINI†, P. MENTRASTI†, AND G. SONTACCHI‡

Abstract. The reversible Turing machine (i.e., r-machine) was introduced initially by C. H. Bennett [*IBM J. Res. Develop.*, 6 (1973), pp. 525–532]. In the first part of the paper a convenient representation of r-machines is introduced by means of diagrams. By using this method the following theorem can be proved: the invertible partial functions are exactly those that can be computed without surplus information by r-machines. Therefore the following problem is pointed out: are the invertible functions that can be computed in polynomial time also r-computable in polynomial time? In the second part of the work this open question is connected with the problem P = NP and with the problem of the existence of "one-way" bijections.

Key words. reversibility, Turing machine, one-way bijection

AMS(MOS) subject classifications. 68Q05, 68Q15, 03D10, 03D15

1. Introduction.

The notion of reversible computation was initially developed with motivations related to physics [1], [5], [6]. Reversible computation is considered a process in which, given any instantaneous state, not only the future state but also the past one is unambiguously defined.

In this paper we are interested in reversible computation only from a mathematical point of view.

In 1973 Bennett [1] defined a reversible Turing machine and proved that any Turing machine (TM) with one tape and some limitations on the input could be simulated by a reversible Turing machine with three tapes. The alphabet of the three-tape Turing machine is variable, as it depends on the number of the 5-tuples of the simulated machine. Generally the simulation produces surplus information. He then proved that the surplus can be reduced to the simple reproduction of the input.

In our paper we use only a one-tape reversible Turing machine with fixed alphabet $\{0, |\}$ (r-machine). The first result is as follows: expressive diagrams are introduced, which permits us to represent all and only the r-machines.

Then, we prove (Theorem 5.5) that the class of computable functions without superfluous information from an r-machine is identical with the class of invertible recursive functions. To obtain this result we make use of simulations in which alphabet and codification are the same (unlike Bennett) both in the simulating machine and the simulated one. The lack of simple correspondence between the 5-tuples of the two machines is not a problem. We do not represent the machines by means of 5-tuples but by means of diagrams.

Finally we prove that a polynomially honest bijection [3] is a two-way bijection if and only if it is poly-time computable with an r-machine, and that the one-way polynomial honest bijections (if they exist) are those that are poly-time computable from an ordinary Turing machine but not from one of our r-machines; their inverse would be computable in polynomial time by a nondeterministic Turing machine, but not by an ordinary Turing machine.

* Received by the editors October 17, 1988; accepted for publication (in revised form) March 9, 1989.

† Dipartimento di Matematica, Università degli Studi di Roma "La Sapienza," Piazzale Aldo Moro 5, 00185 Roma, Italy. This work was partially supported by "Ministero Pubblica Istruzione"-"Progetto Nazionale Teoria de gli Algoritmi."

‡ Dipartimento di Matematica Pura ed Applicata, Università dell'Aquila, Via Roma, 67100 L'Aquila, Italy.

2. Reversible Turing machine.

DEFINITION 2.1. A reversible Turing machine, r-machine, R is determined by a 5-tuple

$\langle Q_R, \Sigma_R, q_{0R}, D_R, T_R \rangle$ such that:

$Q_R = \{q_0, q_1, q_2, \cdots, q_m\}$ is a finite set of states;

$\Sigma_R = \{s_0, s_1, s_2, \cdots, s_n\}$ is a finite alphabet (of the tape);

$q_{0R} = q_0\ (\in Q_R)$ is the initial-final state;

D_R is a function that associates a direction of movement to each state different from q_0, i.e., $D_R : Q_R - \{q_0\} \to \{\rightarrow, \leftarrow\}$;

T_R is a set of transitions, i.e., a permutation on the set $K_R = Q_R \times \Sigma_R$, i.e., $T_R : K_R \leftrightarrow K_R$.

Let us give a short description of the behaviour of R, pointing out the differences regarding the classical TM described with 5-tuples.

A specific state is associated to the machine only in the moment in which it is halfway between two symbols. Instead, when it observes a symbol (and is replacing the symbol with another one) it does not have a specific state (but rather a pair of states: past state–future state).

$D_R(q_i) = \rightarrow (\leftarrow)$ means that while the machine moves along the tape between one symbol and another, being in the q_i state, the direction of its movement is always from left to right (from right to left).

The generic 4-tuple $\langle\langle q_i, s_j\rangle, \langle q_h, s_k\rangle\rangle \in T_R\ (i, h \neq 0)$ has the following meaning. If the machine, as it moves along the tape in state q_i, meets the symbol s_j, it replaces s_j with s_k and moves away in state q_h.

The 4-tuple $\langle\langle q_0, s_j\rangle, \langle q_h, s_k\rangle\rangle \in T_R$ has the following meaning. If the machine in the initial configuration[1] observes the symbol s_j, it replaces s_j with s_k and moves away in state q_h.

The 4-tuple $\langle\langle q_i, s_j\rangle, \langle q_0, s_k\rangle\rangle \in T_R$ has the following meaning. If the machine moves in state q_i and meets the symbol s_j, it replaces s_j with s_k and stops, as the final configuration has been reached.

The 4-tuple $\langle\langle q_0, s_j\rangle, \langle q_0, s_k\rangle\rangle \in T_R$ has the following meaning. If the machine in the initial configuration observes the symbol s_j, it replaces s_j with s_k and stops without any movement done.

Example 2.2. (Reversible Turing machine). $R1 = \langle Q_{R1}, \Sigma_{R1}, q_0, D_{R1}, T_{R1}\rangle$ where:

$$Q_{R1} = \{q_0, q_1, q_2, q_3\};$$

$$\Sigma_{R1} = \{0, |\};$$

$$D_{R1}(q_1) = \leftarrow, D_{R1}(q_2) = \rightarrow, D_{R1}(q_3) = \leftarrow;$$

$$\begin{aligned} T_R = \{&\langle\langle q_0, 0\rangle, \langle q_2, |\rangle\rangle, \langle\langle q_0, |\rangle, \langle q_1, 0\rangle\rangle, \\ &\langle\langle q_1, 0\rangle, \langle q_2, 0\rangle\rangle, \langle\langle q_1, |\rangle, \langle q_3, 0\rangle\rangle, \\ &\langle\langle q_2, 0\rangle, \langle q_1, |\rangle\rangle, \langle\langle q_2, |\rangle, \langle q_3, |\rangle\rangle, \\ &\langle\langle q_3, 0\rangle, \langle q_0, 0\rangle\rangle, \langle\langle q_3, |\rangle, \langle q_0, |\rangle\rangle\};\end{aligned}$$

[1] By "configuration" we mean the content of the tape with indication of the observed cell (which will be indicated by underlining).

if the initial configuration is

$$\ldots\ldots|\quad \underline{0} \quad 0\ldots\ldots$$

the computation goes on as

$$\ldots\ldots|\quad |\overset{q_2}{\rightarrow}0\ldots\ldots$$

$$\ldots\ldots|\quad |\overset{q_1}{\leftarrow}|\ldots\ldots$$

$$\ldots\ldots|\overset{q_3}{\leftarrow}0\quad |\ldots\ldots$$

and the final configuration is

$$\ldots\ldots|\quad 0\quad |\ldots\ldots$$

DEFINITION 2.3. For each r-machine R we define the r-machine

$$R^{-1} = \langle Q_{R^{-1}}, \Sigma_{R^{-1}}, q_{0R^{-1}}, D_{R^{-1}}, T_{R^{-1}} \rangle$$

in the following manner:

$$Q_{R^{-1}} = Q_R;$$

$$\Sigma_{R^{-1}} = \Sigma_R; D_{R^{-1}}(q_i) = \rightarrow \Leftrightarrow D_R(q_i) = \leftarrow;$$

$$q_{0R^{-1}} = q_{0R} = q_0;$$

$$T_{R^{-1}} = T_R^{-1}.$$

We call R^{-1} the reverse of R.

PROPOSITION 2.4. *If R transforms a configuration $\gamma 1$ into another $\gamma 2$, then R^{-1} transforms $\gamma 2$ into $\gamma 1$, always in the same amount of time. In fact R^{-1} operates the inverse computation moving backwards through all the intermediate configurations.*

Example 2.5. The reverse machine of $R1$ of Example 2.2 is

$$R1^{-1} = \langle Q_{R1^{-1}}, \Sigma_{R1^{-1}}, q_0, D_{R1^{-1}}, T_{R1^{-1}} \rangle$$

where:

$$Q_{R1^{-1}} = \mathrm{Q}_{R1} = \{q_0, q_1, q_2, q_3\};$$

$$\Sigma_{R1^{-1}} = \Sigma_{R1} = \{0, |\};$$

$$\{D_{R1^{-1}}(q_i)\} = \{\rightarrow, \leftarrow\} - \{D_{R1}(q_i)\}, i = 1, 2, 3.$$

$$D_{R1^{-1}}(q_1) = \rightarrow, D_{R1^{-1}}(q_2) = \leftarrow, D_{R1^{-1}}(q_3) = \rightarrow.$$

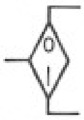
$$\begin{aligned} T_{R1^{-1}} = \{ & \langle\langle q_0, 0\rangle, \langle q_3, 0\rangle\rangle, \langle\langle q_0, |\rangle, \langle q_3, |\rangle\rangle, \\ & \langle\langle q_1, 0\rangle, \langle q_0, |\rangle\rangle, \langle\langle q_1, |\rangle, \langle q_2, 0\rangle\rangle, \\ & \langle\langle q_2, 0\rangle, \langle q_1, 0\rangle\rangle, \langle\langle q_2, |\rangle, \langle q_0, 0\rangle\rangle, \\ & \langle\langle q_3, 0\rangle, \langle q_1, |\rangle\rangle, \langle\langle q_3, |\rangle, \langle q_2, |\rangle\rangle\}. \end{aligned}$$

3. *R*-graphs.

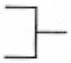
3.1. It is well known that a deterministic Turing machine on the binary alphabet $\Sigma = \{0, |\}$ can also be described by a connected H-graph (Hermes graph) with the following nodes:

0—— (START) in one and only one copy,

——0 (STOP) in one and only one copy,



(IF THEN ELSE)


(CONFLUENCE)

(where for each IF THEN ELSE there is one CONFLUENCE) and the following nodes (representing elementary machines):

—→— (A STEP TO RIGHT)

—←— (A STEP TO LEFT)

—0— (WRITE 0)

—|— (WRITE |).

Note 3.1.1. In this notation the strictly nonreversible components are:]— (it is not known from where the computation originates after a confluence) and —0—, —|— (the previous symbol is not restorable).

It is our intention to give a notation in the form of a graph also for the r-machine on the binary alphabet $\Sigma = \{0, |\}$.

DEFINITION 3.2. We define a class of connected graphs which we call r-graphs and which have the following nodes:

0—— (START) in one and only one copy,

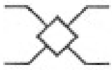
——0 (STOP) in one and only one copy,

—→— (A STEP TO RIGHT),

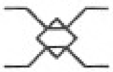
—←— (A STEP TO LEFT),

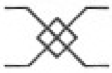
and furthermore the new nodes

—A— (ALTER)


(SPIDERETTE).

3.2.1. "A" represents the elementary reversible machine which exchanges the observed symbol and the "SPIDERETTE" means:


if the observed symbol is 0


if the observed symbol is |.

Note 3.2.2. Each component of the r-graphs is reversible. In particular, with regard to the "SPIDERETTE" we would point out that, once the symbol is known, the output edge determines in an unambiguous manner the input edge and vice versa.

THEOREM 3.3. *Each r-graph represents a r-machine on the alphabet $\Sigma = \{0, |\}$ and (vice versa) each r-machine on the alphabet $\Sigma = \{0, |\}$ has an r-graph which represents it.*

Proof. Let G be an r-graph with m movement nodes (i.e., $\rightarrow$, $\leftarrow$). To construct the r-machine R represented by G first at all we label 0— and —0 with q_0 and each movement node with a distinct q_i $(i = 1, 2, \cdots, m)$.

We define $Q_R = \{q_0\} \cup \{q_i\,(i = 1, 2, \cdots, m)\}$.

We define $D_R(q_i)$ equal to the symbol of the node labelled with q_i $(i = 1, 2, \cdots, m)$.

Now we define T_R as follows.

3.3.1. $\langle\langle q_i, s_j\rangle, \langle q_h, s_k\rangle\rangle \in T_R$ if and only if there is a path on G from q_i to q_h (directed to —0 and from 0—) that satisfies the following requests:

(a) just out of the node labelled q_i, the symbol observed is s_j;

(b) when an A node is passed, the symbol changes;

(c) when a SPIDERETTE node is passed, the path to be followed is established in agreement with § 3.2.1;

(d) just before passing the node labelled q_h, the observed symbol is s_k.

To verify that T_R is a permutation of $Q_R \times \Sigma_R$, the following must be verified:

(i) for each $\langle q_i, s_j\rangle \in Q_R \times \Sigma_R$ there is one and only one $\langle q_h, s_k\rangle \in Q_R \times \Sigma_R$ such that § 3.3.1 holds, and

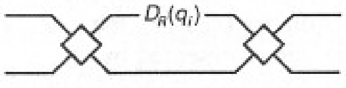
(ii) for each $\langle q_h, s_k\rangle \in Q_R \times \Sigma_R$ there is one and only one $\langle q_i, s_j\rangle \in Q_R \times \Sigma_R$ such that § 3.3.1 holds.

The uniqueness of (i) derives from the fact that, starting from any labelled node of the graph, the observed symbol determines unambiguously the future path until another labelled node has been reached; the uniqueness of (ii) is verified in a similar manner, observing that the backward path is determined symmetrically with the same rules.

The existence of (i) (and similarly for (ii)) is guaranteed by the fact that in the path from q_i to q_h (from q_h to q_i) the same edge can be covered at the most two times (the symbols of the alphabet being two).

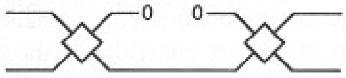
3.3.2. Vice versa, given an r-machine, to have an r-graph which represents it, we draw for each state $q_i \neq q_0$ the following:

figure F_i



and for q_0,

figure F_0



Therefore the free edges must be connected as follows:

For each element $\langle\langle q_i, s_j\rangle, \langle q_h, s_k\rangle\rangle \in T_R$ an output edge of the figure F_i of the r-graph corresponding to q_i is connected with an input edge of the figure F_h of the r-graph corresponding to q_h in the following manner:

(a′) if $s_j = 0$ and $s_k = 0$ we connect directly the upper output of F_i with the upper input of F_h;

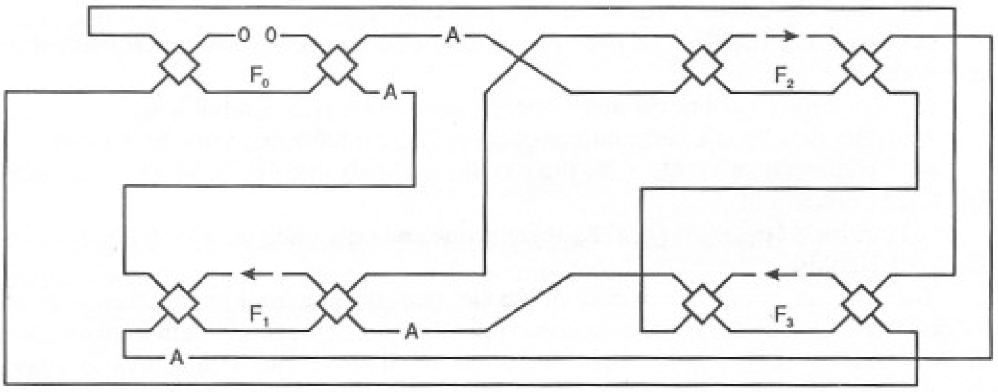
(b′) if $s_j = 0$ and $s_k = |$ we connect the upper output of F_i with the lower input of F_h placing on the path an A;

(c′) if $s_j = |$ and $s_k = 0$ we connect the lower output of F_i with the upper input of F_h placing on the path an A;

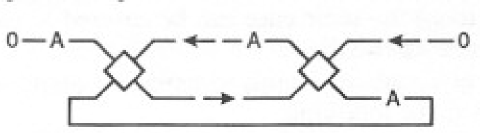
(d′) if $s_j = |$ and $s_k = |$ we connect directly the lower output of F_i with the lower input of F_h.

Note 3.3.3. The r-graph, which represents an r-machine, is not unique. If the following transformations r-graph → r-machine and r-machine → r-graph are operated one after the other generally we do not return to the initial r-graph but to a new r-graph which represents the same r-machine. Furthermore the r-graph that represents the r-machine R, obtained through the translation described in § 3.3.2, can result nonconnected, but in such a case R has inaccessible states (for any initial configuration) and it is equivalent to a machine that has fewer states, of which its translation in r-graph is the connected component of the first graph (necessarily unique) which contains 0— and —0.

Example 3.4. The translation into r-graph of $R1$ of Example 2.2 is the following:



but the following r-graph also represents $R1$.



3.5. Given an r-graph as G, we consider the r-graph, which we call G^{-1}, obtained from G with a specular symmetry (right ↔ left). G^{-1} represents an r-machine which is the reverse machine of the one represented by G.

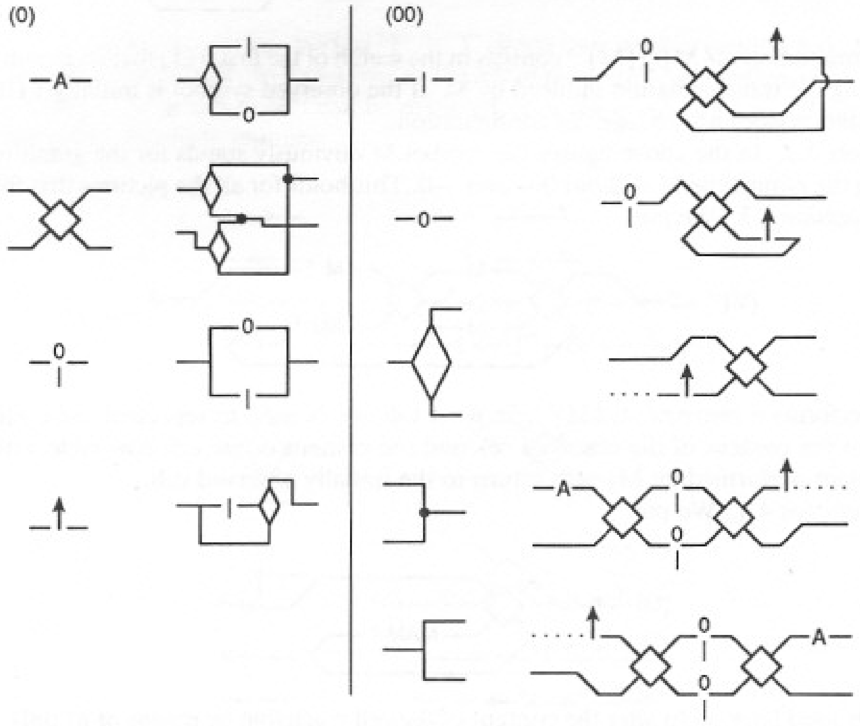
3.6. It is known that a nondeterministic Turing machine on the binary alphabet $\{0, |\}$ can be described by a connected $\bar{H}$-graph which has the same nodes as the H-graph (3.1) with the addition of the node —[, its meaning being nondeterministic choice of the next operation to perform. We designate with 0—$\substack{0\\|}$—0 the nondeterministic Turing machine, its computation consisting in the nondeterministic choice of writing 0 or | in the observed cell. We designate with 0—↑—0 the deterministic Turing machine that does not provide any output for any input configuration.

DEFINITION 3.7. We define $\bar{r}$-graph as a connected graph, its nodes being the same as those of r-graphs, with the addition of —$\substack{0\\|}$— and —↑—.

3.8. All the deterministic reversible Turing machines can be represented by r-graphs (Definition 3.2), all the ordinary Turing machines can be represented by H-graphs (§ 3.1) (but not by r-graphs). Vice versa, the nondeterministic Turing machines can be represented by $\bar{H}$-graphs or, as well, (only in a manner formally reversible) by an $\bar{r}$-graph (substantially for nondeterministic machines there is no difference between reversible and nonreversible). In fact it yields the following proposition.

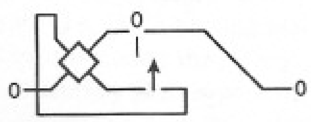
PROPOSITION 3.9. *$\bar{H}$-graphs and $\bar{r}$-graphs describe the same class of machines.*

Proof. To translate an $\bar{r}$-graph into an $\bar{H}$-graph we substitute the nodes on the left with the figures on the right in (^{0}) and to translate an $\bar{H}$-graph into an $\bar{r}$-graph we substitute the nodes on the left with the figures on the right in (^{00}):[2]



Note 3.10. If we represent the nondeterministic Turing machine M with an r-graph, its specular copy is still an r-graph that represents the machine M^{-1}, the reverse of M. We can extend what was said in Proposition 2.4 to the nondeterministic machine M. That is, if there is a computation of M that transforms an initial configuration $\gamma 1$ into a final configuration $\gamma 2$, then there is a computation of M^{-1} that operates the inverse transformation from $\gamma 2$ to $\gamma 1$ and the two computations take the same amount of time. For each nondeterministic Turing machine M there is a reverse machine M^{-1}: if M is reversible, M^{-1} is reversible also. Vice versa, a deterministic Turing machine has a reverse machine that is generally nondeterministic.

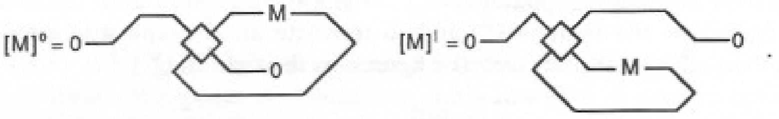
Example 3.11. The reverse machine of 0—|—0 is



[2] The dotted edges are not used and can be connected to each other (we point out that they are equal in number).

4. Simulation of Turing machine with *r*-machine. We will now introduce a few notations which we will use further on.

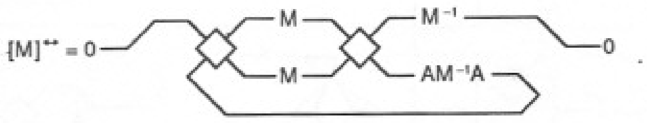
Notation 4.1. Let M be any *r*-machine; we put



The computation of $[\mathrm{M}]^0$ ($[\mathrm{M}]^|$) consists in the search of the first 0 (|) that is encountered repeating the transformation induced by M. If the observed symbol is initially | (0) the computation does not change the configuration.

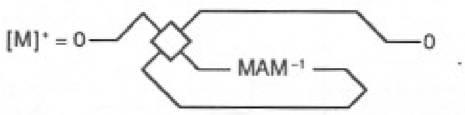
Note 4.2. In the above figures the symbol M obviously stands for the graph representing the *r*-machine M without 0— and —0. This holds for all the pictures that follow.

Notation 4.3. We put



If M performs a movement, $[\mathrm{M}]^{\leftrightarrow}$, in what follows, is used to represent the exchange between the content of the observed cell and the content of the cell reachable with the movement performed by M, with return to the initially observed cell.

Notation 4.4. We put



$[\mathrm{M}]^+$ is used later on to alter the content of the cell reachable by means of M only if the content of the initial cell (which is also the final one) is |.

Notation 4.5. We indicate with SYM(M) the *r*-graph that is obtained from an *r*-graph representing M by substituting each → with ← and vice versa (SYM(M) and M^{-1} must not be confused).

Notation 4.6. $\rightarrow^n$ ($\leftarrow^n$) stands for → (←) repeated *n* consecutive times.

DEFINITION 4.7. We call a semitape Turing machine a Turing machine such that the final cell is the initial one and each cell to the left of the initial-final cell is never observed during the computation.

THEOREM 4.8. *Let* M *be a semi-tape Turing machine on the alphabet* $\Sigma = \{0, |\}$ *that transforms the initial generic configuration*

$$\cdots \underline{i_0}\, i_1\, i_2\, i_3 \cdots^{3}$$

in the final one

$$\cdots \underline{\omega_0}\, \omega_1\, \omega_2\, \omega_3 \cdots.$$

[3] $i_0, i_1, i_2, i_3, \cdots, \omega_0, \omega_1, \omega_2, \omega_3, \cdots, \in \{0, |\}$, the underlined symbols ($\underline{i_0}$, $\underline{\omega_0}$) indicate the observed cell (in this case the same one).

Then there exists an r-machine M^G *on the alphabet* $\Sigma = \{0, |\}$ *that transforms the initial configuration*

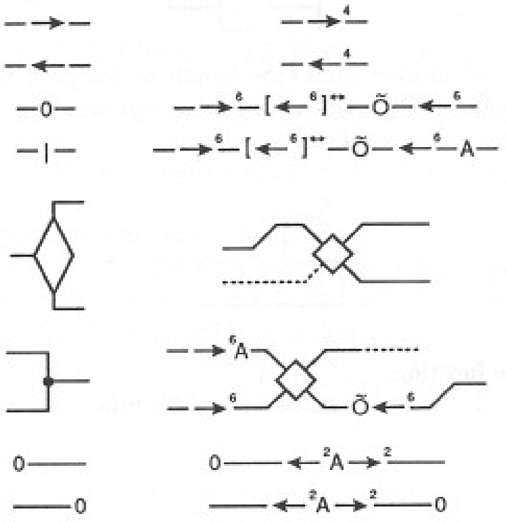
$$\cdots 0\,0\,\underline{i_0}\,0\,0\,0\,i_1\,0\,0\,0\,i_2\,0\,0\,0\,i_3\,0 \cdots$$

in the final one

$$\cdots 0\,0\,\underline{\omega_0}\,?\,0\,!\,\omega_1\,?\,0\,!\,\omega_2\,?\,0\,!\,\omega_3\,?\,0\,! \cdots,$$

where the content of the cells marked "?" or "!" denotes unspecified suitable bits. Moreover, during the computation, M^G *can visit to the left of the initial-final cell two cells at most.*

Proof. The graph of M^G can be obtained from the graph of M by substituting the nodes in the following manner:



Where

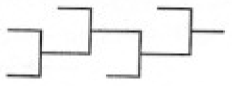
$$\tilde{O} = o\!- \leftarrow^4 A \rightarrow^4 H[H^{-1}]^{\leftrightarrow} \rightarrow^2 A \rightarrow^2 H^{-1} \leftarrow^4 A \rightarrow^4 -\!o$$

$$H = o\!- \leftarrow^4 [\leftarrow^4]^{|} \rightarrow [\rightarrow^4]^{0} \leftarrow^2 -\!o$$

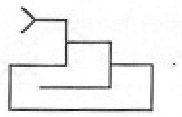
Õ has the effect of printing a 0 in the observed cell. This cannot be done "sic et simpliciter" (the information cannot be destroyed), but such a result can be obtained by exchanging the content (?) of the observed cell with that of another cell which surely contains 0 and of which the new content, which we call garbage, will no longer be used; H (the initial observed cell is the same as that of Õ) searches for the first useful cell to store the garbage. The three intercalated cells (0 0 0) with the "i's" of the input of M^G and with the "ω's" of the output (? 0 !) are used by Õ in the following manner: the first ones of each set of three cells are used, in sequence, for storing the garbage; the central ones always contain 0 (they are only temporarily altered by Õ for operative needs of H); the third ones are altered in sequence by Õ which assigns them the content |. Therefore with each intervention of Õ a new set of three 0 0 0 is transformed in ? 0 !; in the final configuration the "!'s" in the third position (of each set of the three) will be | for some left segment and then 0. To the right of the last | also the bits indicated with ? will naturally be 0.

PROPOSITION 4.9. *The computation time of* M^{G}, *when the input changes, is at most proportional to the square of the computation time of* M.

Proof. In fact, $\tilde{\mathrm{O}}$ requires a time at most proportional to the length of that part of tape observed up to then, which in turn is at most proportional to the computation time spent by M until then. The number of operations of $\tilde{\mathrm{O}}$ during the simulation corresponds to the number of nodes —0—, —|—,]— that are crossed during the simulated computation and it is at the most proportional to the computation time of M. The constant of proportionality can depend on M, but not on the particular computation. In fact, it is true that the crossing of a node]— does not correspond to a computation step, i.e., it does not involve a change of state of the simulated machine. At the beginning or at the end of a path that crosses only]— nodes in fixed number,



there must be nodes of another kind (the length of the path will only influence the constant of proportionality). Otherwise a loop made up entirely of]— would correspond necessarily to a divergent computation



5. *R*-computable functions.

DEFINITION 5.1. Let X and Y be two sets. The function $f: X \to Y$ is invertible if and only if

$$x_1, x_2 \in X, \quad f(x_1) = f(x_2) \Rightarrow x_1 = x_2.$$

Note 5.2. It is known that if f is invertible there exists $f^{-1}: Y \to X$ such that:
(5.2.1) $x \in X \Rightarrow f^{-1}(f(x)) = x$.
(5.2.2) $y \in Y \Rightarrow f(f^{-1}(y)) = y$.
(5.2.3) $(f^{-1})^{-1} = f$.

DEFINITION 5.3. $f \in \mathbf{I}$ if and only if f is a recursive partial invertible function of one variable.

Note 5.3.1. It is known that if $f \in \mathbf{I}$ then $f^{-1} \in \mathbf{I}$.

DEFINITION 5.4. Let $n \in \mathbf{N}$ and $n = b_0 b_1 b_2 \cdots b_{|n|-1}$ be its binary notation, $b_i \in \{0, |\}$ ($b_0 = |$ if and only if $n > 0$, $b_0 = 0$ and $|n| = 1$ if and only if $n = 0$). Further on we will say that the following tape configuration

$$\cdots 00\underline{0}b_0 \mid b_1 \mid b_2 \mid \cdots b_{|n|-1} \mid 00 \cdots,$$

which we indicate with $\langle\langle n \rangle\rangle$, represents n.

Let $X \subseteq \mathbf{N}$ and $Y \subseteq \mathbf{N}$, we say that $f: X \to Y$ is r-computable if and only if there is an r-machine R_f (that computes f) such that for each $n \in X$ transforms the initial configuration representing n in the final one representing $f(n)$ (without superfluous information).

Note 5.4.1. We point out that the "|'s" that intercalate the b_i's are necessary to indicate the end of the number represented. They would not be necessary if the alphabet of the tape contained three symbols, for example, $\{0, |, \#\}$.

5.5. Main theorem. *Let* $X \subseteq N$ *and* $Y \subseteq N$, $f : X \to Y$,

$$f \in \mathbf{I} \Leftrightarrow f \textit{ is r-computable}.$$

Proof. ($\Leftarrow$) The r-machines are Turing machines, therefore f is recursive. If for $x_1 \neq x_2, f(x_1) = f(x_2)$ then (see § 2.4) the machine R_f starting from $\langle\langle y \rangle\rangle$ would have two distinct final configurations.

($\Rightarrow$) Consider (see §§ 5.3.1 and 4.8) the r-machines $\mathrm{M}_f^{\mathrm{G}}$, $\mathrm{M}_{f^{-1}}^{\mathrm{G}}$. To define R_f we need to define two suitable r-machines, i.e., EXPANDED and MIRROR. EXPANDED transforms the initial configuration

$$\cdots ???000\underline{0}b_0 \,|\, b_1 \,|\, b_2 \,|\, \cdots b_n \,|\, 000 \cdots$$

into the final

$$\cdots ???000\underline{0}b_0 000 \,|\, 000 b_1 000 \,|\, 000 b_2 000 \,|\, \cdots 000 b_n 000 \,|\, 000 \cdots$$

without ever visiting the cells containing ?.

MIRROR writes on the left semi-tape, which is initially empty, a specular copy of the right semi-tape only with regard to every fourth cell, the others remaining 0. More precisely MIRROR transforms the initial configuration

$$\cdots 0000\underline{0}?0!b_0?0! \,|\, ?0!b_1?0! \,|\, ?0! \cdots$$

into the final

$$\cdots \,|\, 000 b_1 000 \,|\, 000 b_0 0000000\underline{0}?0!b_0?0! \,|\, ?0!b_1?0! \,|\, ?0! \cdots$$

$$\begin{aligned}\text{EXPANDED} = 0\text{—}[\to^2]^0[[\to^3]^{\leftrightarrow} \to^5 \mathrm{A}[\to^8]^0[[\to^6]^{\leftrightarrow} \leftarrow^4 \\ [\to^6]^{\leftrightarrow} \leftarrow^4]^0 \mathrm{A}[\to^6]^{\leftrightarrow} \leftarrow^4 [\to^6]^{\leftrightarrow} \leftarrow^3]^0 \leftarrow^3\text{—}0\end{aligned}$$

and

$$\begin{aligned}\text{MIRROR} = 0\text{—} \leftarrow^2 \mathrm{A} \leftarrow^4 \mathrm{A} \to^8 \mathrm{A} \to^6 [\mathrm{A} \to^2 \mathrm{A} \leftarrow^8 \mathrm{A} \to^2 [\to^6 [\leftarrow^4]^{|}[\leftarrow^4]^{|} \leftarrow^2]^{+} \to^6 \\ [\leftarrow^4]^{|}[\leftarrow^4]^{|} \mathrm{A} \leftarrow^6 \mathrm{A} \leftarrow^2 [\to^4]^{|}[\to^4]^{|} \to^6 \mathrm{A}]^0 \mathrm{A} \leftarrow^6 \mathrm{A} \to^6 \\ [\leftarrow^8]^0 \mathrm{A} \leftarrow^{10} [\leftarrow^4]^{|} \mathrm{A} \leftarrow^6 [\to^8]^0 \to^2 \mathrm{A} \to^2 \text{—}0.\end{aligned}$$

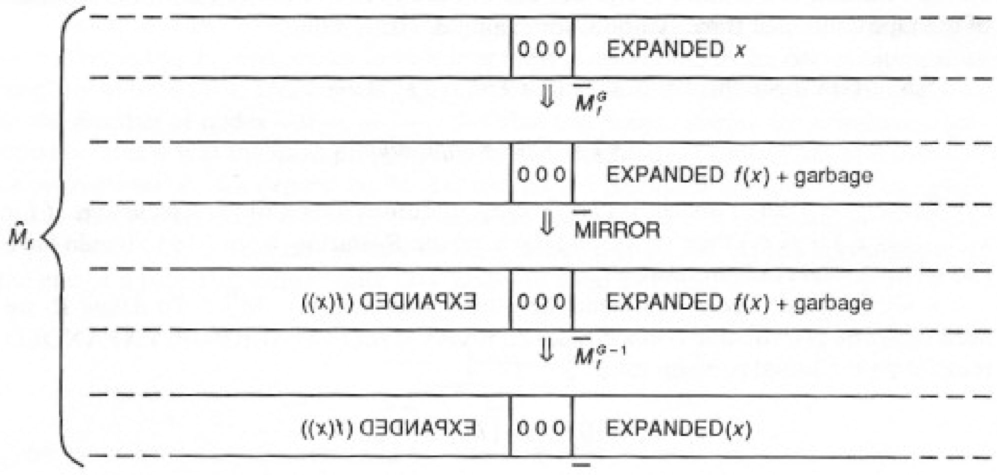
Suppose

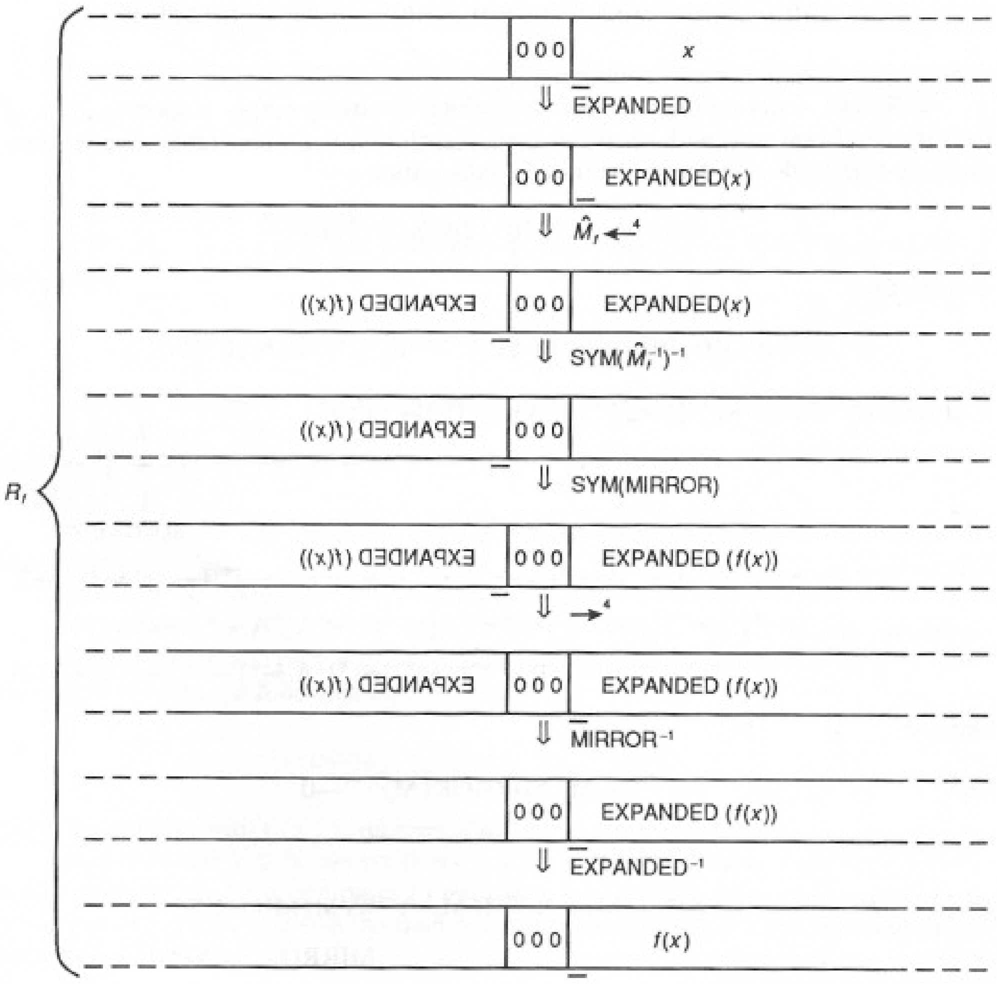
$$\tilde{\mathrm{M}}_f = 0\text{—}\mathrm{M}_f^{\mathrm{G}}\,\text{MIRROR}(\mathrm{M}_f^{\mathrm{G}})^{-1}\text{—}0. \tag{5.5.1}$$

We can define

$$\begin{aligned}R_f = 0\text{—EXPANDED } \tilde{\mathrm{M}}_f \leftarrow^4 \text{SYM}(\tilde{\mathrm{M}}_{f^{-1}})^{-1}\text{SYM(MIRROR)} \to^4 \\ \text{MIRROR}^{-1}\text{EXPANDED}^{-1}\text{—}0.\end{aligned} \tag{5.5.2}$$

We can represent $\hat{M}_f$ and R_f in the following manner:




R_f
000
x
$\overline{\text{EXPANDED}}$
EXPANDED(x)
$\hat{M}_f \leftarrow^4$
EXPANDED ($f(x)$)
SYM($\hat{M}_f^{-1}$)$^{-1}$
SYM(MIRROR)
EXPANDED ($f(x)$)
$\rightarrow^4$
$\overline{\text{MIRROR}}^{-1}$
$\overline{\text{EXPANDED}}^{-1}$
$f(x)$

Note 5.6. If we construct the reversible machine that computes f as suggested in (5.5.1) and (5.5.2) the computation time of R_f is about two times the computation time of M_f^G and $M_{f^{-1}}^G$ (plus the computation time of EXPANDED and MIRROR at the most proportional to $|x|^2$ or $|f(x)|^2$).

6. Existence of one-way bijections. We will first define the HB set of the polynomially honest bijections.

DEFINITION 6.1. $f \in \mathrm{HB} \Leftrightarrow f \in \mathbf{I}$, f is total and there is a polynomial p such that

(6.1.1.) $$|f(x)| \leq p(|x|), p(|x|) \leq |f(x)| \text{ for each } x \in \mathbf{N}.$$

Notation 6.1.2. If $X \subseteq \mathrm{HB}$, with X^{-1} we indicate the set of inverse bijections:

$$f^{-1} \in X^{-1} \Leftrightarrow f \in X.$$

We now define three subsets of HB: HBP, HBRP, HBNP.

DEFINITION 6.2. One bijection $f \in \mathrm{HB}$ belongs also to HBP [HBRP, HBNP] if there is an ordinary [reversible, nondeterministic] Turing machine M_f and a polynomial p such that:

(1) for each $\langle x, y\rangle \in \mathbf{N}^2$ there is a computation C of M_f that transforms the input $\langle\langle x\rangle\rangle$ in the output $\langle\langle y\rangle\rangle$ if and only if $y = f(x)$;

(2) the computation time is less than or equal to $p(|x|)$ for at least a computation C.

Note 6.2.1. $\langle\langle x\rangle\rangle$ is to be understood as defined in 5.4. The "if and only if" at point (1) indicates that M_f (even if nondeterministic) must not compute two distinct values starting from the same x. The condition "for at least one computation" at point (2) is necessary (meaningful) only if M_f is nondeterministic.

Obviously we have

(6.3.1) $\mathrm{HBRP} \subseteq \mathrm{HBP}$,

(6.3.2) $\mathrm{HBP} \subseteq \mathrm{HBNP}$,

and furthermore

(6.3.3) $\mathrm{HBRP} = \mathrm{HBRP}^{-1}$.

(6.3.4) $\mathrm{HBNP} = \mathrm{HBNP}^{-1}$

(in fact $M_{f^{-1}} = M_f^{-1}$ if M_f is a reversible or nondeterministic machine, see Proposition 2.4, and for Definition 6.1 there is no difference between polynomiality in regard to the range and the value of f.)

On the other hand, we do not know whether

(6.3.5) $\mathrm{HBP} = \mathrm{HBP}^{-1}$?

The set $\mathrm{B2V} = \mathrm{HBP} \cap \mathrm{HBP}^{-1}$ is known as the class of the two-way bijections, i.e., computable in two directions in polynomial time.

$\mathrm{B1V} = \mathrm{HBP} - \mathrm{HBP}^{-1}$ is known as the class of one-way bijections, i.e., computable in polynomial time in one direction but not in the other.

THEOREM 6.4. B2V = HBRP.

The two-way bijections are all and only those computable in polynomial time by a reversible Turing machine.

Proof. if $f \in \mathrm{HBRP}$ for 6.3.1 and 6.3.3 then $f \in \mathrm{HBP} \cap \mathrm{HBP}^{-1}$; if $f \in \mathrm{HBP} \cap \mathrm{HBP}^{-1} = \mathrm{B2V}$, for § 5.5, Note 5.6, and Proposition 4.9 then $f \in \mathrm{HBRP}$.

The problem of the existence of one-way bijections ($\mathrm{B1V} \neq \varnothing$?) is still open (it is also connected to the problem $\mathrm{P} = \mathrm{NP}$). For us it is not solved, but can be reduced to the following: "HBP $\neq$ HBRP?" (Are there polynomially honest bijections polytime computable by an ordinary Turing machine but not by a reversible Turing machine?), which in turn is equivalent to the problem (6.3.5).

The practical meaning of Theorem 6.4 can be illustrated as follows: let us suppose that f is a bijection easy to compute in one direction, obviously $f \in$ HBP, but difficult to compute (at least intuitively) in the other direction, so that one doubts its bipolynomiality: $f^{-1} \in$ HBP? We therefore do not know whether $f \in$ B2V or $f \in$ B1V. To try to establish it, it is not necessary to deal with the difficult inverse problem, i.e., to establish whether $f^{-1} \in$ HBP or $f^{-1} \in$ HBNP $-$ HBP. In fact to try to prove that $f \in$ B2V all that is necessary is to try to program the direct problem by means of the construction[4] of a reversible Turing machine that computes it in polynomial time. If one succeeds in this, apart from having resolved such a problem, it is possible to obtain with only the aid of a mirror a polynomial procedure for the inverse problem. Obviously to prove the contrary ($f \in$ B1V) it will be necessary to establish that a function f, computable in polynomial time by a deterministic Turing machine, is not computable, always in polynomial time, by any reversible Turing machine. This would no doubt be more difficult, but if it were done the result would be striking. In fact if it resulted

$$f \in \text{HBP} - \text{HBRP} \qquad (f \in \text{B1V} \neq \varnothing),^5$$

furthermore for problems 6.3.2 and 6.3.4 and Theorem 6.4, one would have

$$f^{-1} \in \text{HBNP} - \text{HBP}.$$

Let Π be the predicate so defined

$$\Pi[(2x+1)2^y] = \begin{cases}\text{true}\\ \text{false}\end{cases} \Leftrightarrow \lfloor f^{-1}(x)/2y \rfloor \text{ is } \begin{cases}\text{odd}\\ \text{even.}\end{cases}$$

It would result $\Pi \in \text{NP} \cap \text{CoNP} - \text{P}$ (result even stronger than $\Pi \in \text{NP} - \text{P}$, i.e., $\text{P} \neq \text{NP}$). In fact, it is easy to prove

$$f^{-1} \in \text{HBNP} \Rightarrow \Pi \in \text{NP} \cap \text{CoNP}$$

$$\Pi \in \text{P} \Rightarrow f^{-1} \in \text{HBP}.$$

REFERENCES

[1] C. H. Bennett, *Logical reversibility of computation*, IBM J. Res. Develop., 6 (1973), pp. 525–532.

[2] H. Hermes, *Enumerability, Decidability, Computability*, Springer-Verlag, Heidelberg, New York, 1969.

[3] K. Ko, *On some natural complete operators*, TCS, 37 (1985), pp. 1–30.

[4] H. R. Lewis and C. H. Papadimitriou, *Element of the Theory of Computation*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[5] R. P. Feynman, *Quantum Mechanical Computers*, California Institute of Technology, Pasadena, CA, 1984.

[6] T. Toffoli, *Computation and construction universality of reversible cellular automata*, J. Comput. System Sci., 15 (1977), pp. 213–231.

[4] Obviously one cannot use the general method of the proof of the Theorem 5.5, because it assumes the knowledge of $M^G_{f^{-1}}$, but one must directly construct R_f.

[5] It is known that $\text{B1V} \neq \varnothing \Rightarrow \text{P} \neq \text{NP} \cap \text{CoNP}$.