INFORMATIQUE THÉORIQUE ET APPLICATIONS

MAURICE MARGENSTERN

The laterality problem for non-erasing Turing machines on $\{0,1\}$ is completely solved

Informatique théorique et applications, tome 31, n° 2 (1997), p. 159-204

<http://www.numdam.org/item?id=ITA_1997__31_2_159_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 31, n° 2, 1997, pp. 159-204)

THE LATERALITY PROBLEM FOR NON-ERASING TURING MACHINES ON $\{0,1\}$ IS COMPLETELY SOLVED (*)

by Maurice MARGENSTERN ([1])

Abstract. – *In a previous work,* [2], *we defined a criterion which allowed to separate cases when all non-erasing Turing machines on* $\{0,1\}$ *have a decidable halting problem from cases where a universal non-erasing machine can be constructed. Applying a theorem which entails the just indicated frontier and analogous techniques based upon a qualitative study of the motions of the head of a Turing machine on its tape, another frontier result is here proved, based upon a new criterion, namely the number of left instructions. In this paper, a complete proof of the decidability part of the result is supplied. The case of a single left instruction with a finite alphabet in a generalized non-erasing context is also delt with. Thus, the laterality problem, raised in the early seventies, see* [9], *solved on* $\{0,1\}$ *alphabet without restriction, is now completely solved in the non-erasing case.*

Résumé. – *Dans un article antérieur,* [2], *nous définissions un critère permettant de séparer les cas où une machine de Turing non-effaçante sur* $\{0,1\}$ *possède un problème de l'arrêt décidable de ceux où l'on peut construire une machine de Turing universelle. En appliquant un théorème qui entraîne l'existence de la frontière qui vient d'être indiquée et des techniques analogues fondées sur l'étude qualitative des mouvements de la tête d'une machine de Turing sur son ruban, nous démontrons ici un autre résultat de frontière fondé sur un autre critère, à savoir le nombre d'instructions gauches. Dans cet article, nous donnons une démonstration complète de la partie "décidabilité" de ce résultat. Nous traitons aussi le cas d'une unique instruction gauche avec un alphabet fini quelconque dans un contexte non-effaçant généralisé. Posé au début des années soixante-dix, voir* [9], *le problème de la latéralité, résolu depuis sur l'alphabet* $\{0,1\}$ *sans restriction, est maintenant complètement résolu dans le cas non-effaçant.*

1. INTRODUCTION

1.1. Since Turing's work of 1936, [17], it is well known that the halting problem of a Turing machine is undecidable. In 1956, it was proved by Shannon, [16], that the same problem remains undecidable when it is restricted to deterministic machines with one head and one tape, bi-infinite,

(*) Received December 1996.
([1]) I.U.T. de Metz, L.R.I.M. (Université de Metz), et L.I.A.F.A. (CNRS, Université Paris 7).

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/97/02/$ 7.00/

on alphabet $\{0, 1\}$. Which further restrictions must be set on Turing machines in order to make the halting problem decidable for any machine of the class thus obtained?

In the sixties and the early seventies, some effort was made in that direction. To a large extent, this question appears to be more difficult if, as we shall do in our sequel, we fix our attention only on deterministic Turing machines with one head and one tape, bi-infinite. This had led to partial answers which are far from closing the matter. The state of the art may be summarized by figure 1 below.

On that figure, each point $s \times l$ represents the set of all Turing machines whose program contains s states, but the halting state(s), and uses l letters, including the blank of the tape. Let us call such a point *decidable* if the halting problem is decidable for any Turing machine in the corresponding set, and *undecidable* if the halting problem restricted to this set of machines is undecidable.

In figure 1, black squares indicate sets which contain a universal Turing machine. Consequently, the corresponding point is undecidable. By restricting the number of states or the alphabet of the machine, it is easily seen that any $x \times y$ universal machine of figure 1 says that for any couple $n \times m$, with $n \geq x$ and $m \geq y$, the corresponding point is undecidable too. White squares indicate points known to be decidable. Crosses indicate points, the status of which is not known. Indices refer to the year and to the quoted literature.

As it was pointed out recently in [12] and [14], although the number of points with an unknown status is finite, the problem of splitting points just in two classes – one containing all decidable points, the other one containing all undecidable points – is far from being closed.

It is of course a striking fact that only a few points of figure 1 outside the axises are known to be decidable. By looking closer at the methods involved for solving the halting problem about the corresponding sets of machines, we see that one way, for tackling the problem, as it was introduced in [6], consists in finding out an integer valued function c defined on the considered set M of Turing machines with the following property : there is an integer f such that the halting problem is decidable for any machine T from M such that $c(T) < f$, and for any $k \geq f$ one can always construct a universal machine U in M such that $c(U) = k$. In that case, c is called *decidability criterion* and f is called its *frontier value.*

The number of symbols of the machine alphabet provides a simple example of a decidability criterion. It is plain that any Turing machine on a one-letter

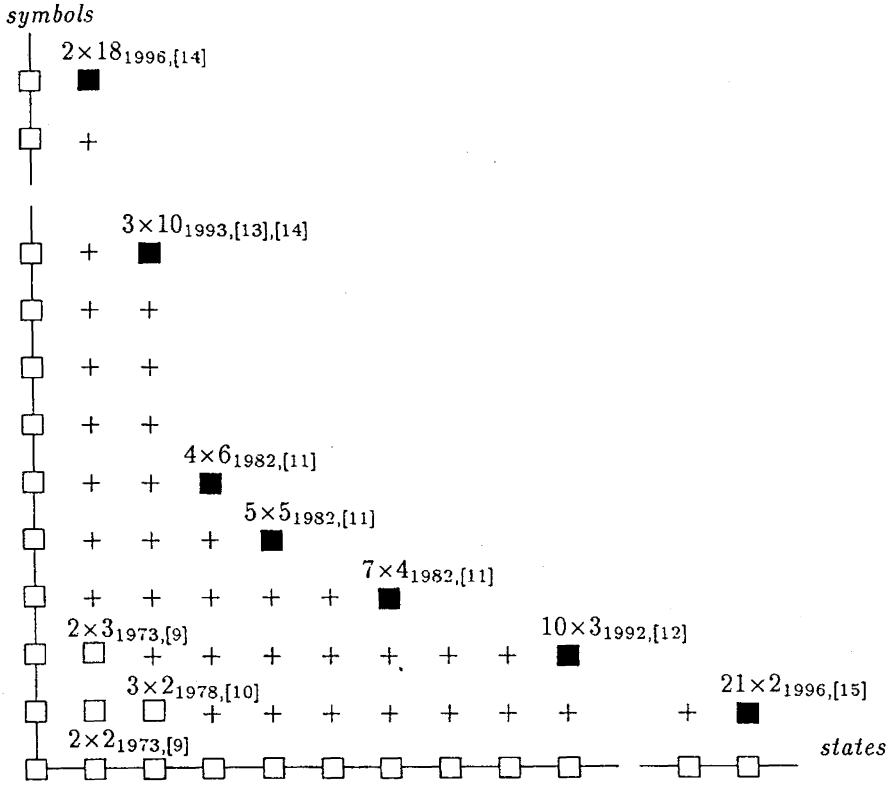


Figure 1. – The present state of the art.

alphabet has a decidable halting problem. Shannon's result, see [16], shows that starting from two letters for the machine alphabet, it is always possible to construct a universal Turing machine. The same paper shows that the number of states also provides a decidability criterion with also two as a frontier value.

1.2. In this paper, we focus our attention on one criterion, the laterality number, see later below, applied to subsets of *non-erasing* Turing machines.

Non-erasing Turing machines were first considered by Wang, see [18], as Turing machines on alphabet $\{0, 1\}$ which are not allowed to erase the non blank symbol (usually 1). Nevertheless this strong restriction, it was proved by Wang that non-erasing Turing machines turn out to also possess universal abilities. Consequently, the whole previous discussion is still meaningful for these machines, but it remained unexplored since Wang time until [2].

The non-erasing constraint has certainly connections with phenomena studied in mathematics or nature sciences where heavy conservative properties fall under observation. From this point of view, there are connections between the non-erasing constraint and monotonicity or positivity in mathematics, with entropy in physics, with conservative properties in biology, especially in genetics and embryogenesis. Indeed, following Zykin's proof of Wang theorem, see [19] and [2], in the non-erasing world, universality is proved by copying the encoding of some Turing machine configuration into the encoding of the next one since both configurations differ by at most two contiguous cells. This process reminds the conservation mechanisms of life reproduction in which evolution is made possible by mutations occurring at isolated places along a discrete time. On the other hand, the marking process involved by copying the current configuration into the next one reminds the ultimate degradation of material according to entropy law. On the tape of the simulating non-erasing machine, the evolving part of the tape may appear as the living upper part of a coral block, while the rest of the configuration corresponds to the under-water, inert and unchanged part of that same block.

Inside practical computer science, two hardware features do implement the non-erasing situation : punch-cards and CD-ROMs for which a hole can never be filled out. Wang theorem shows that the theoretic power of these computational tools is the same as the theoretic power of ordinary computer processing.

These reasons make it worth of interest to investigate the non-erasing world. In spite of the strong constraint imposed by non-erasing, results in this domain are not at all trivial as, we think, the reader will be convinced by reading carefully [2] and this paper.

1.3. Although not at all stated in these terms, Pavlotskaya's paper [9] considers the number of instructions ordering a move to the left as a possible decidability criterion. The paper just proves that any Turing machine on $\{0,1\}$ with a single instruction to the left has a decidable halting problem. The proof is not a all trivial and this function on Turing machines happened to be proved a decidability criterion only very recently, see [7], after the result of this paper was proved. That criterion result closes the laterality problem for Turing machines on $\{0,1\}$ in general. Of course, it does not therefore follow that the same problem would also be closed for any subset of Turing machines with respect to the discussion depending on the number of letters in the machine alphabet and the laterality number.

Quoting the universality result also mentionned in [6], the attention is drawn, in our sequel, on the decidability side of the laterality frontier result there stated within the frame of non-erasing machines. However, the proof given here is stronger than the proof sketched in [6] since in that paper, as well as in [2], results are proved only for machines without stationnary instructions (see the definition below, section 2). Those results are summarized as indicated in figure 2, below, where black squares indicate non-erasing Turing machine sets in which a universal machine can be constructed and white squares indicate that for the corresponding non-erasing Turing machine sets the halting problem is decidable. Call here too points decidable or undecidable, accordingly.

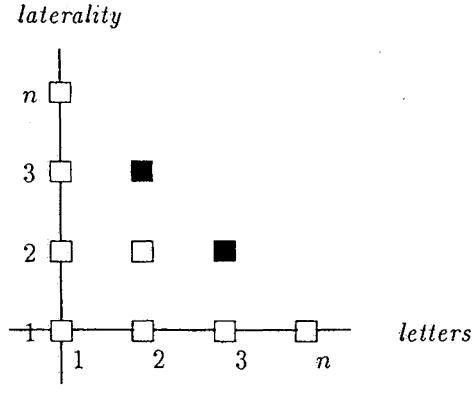


Figure 2.

Theorem 1 states that point 2×2, *i.e.* 2 letters and 2 as a laterality number, corresponds to a decidable halting problem. The undecidability of point 2×3 is the content of theorem 2, see [5]. As it will be seen in section 7, it therefore easily follows that point 3×2 also is undecidable. The decidability of all points $n \times 1$ is the content of theorem 3. As suggested by figure 2, the status of any point is known since point $x \times y$ is undecidable if so is point $u \times v$ for some u, v with $u \leq x$ and $v \leq y$.

In section 2, precise definitions are given, which are necessary to state the decidability criterion theorems : colour number theorem and laterality number theorem.

In section 3 a general analysis of the machine head motion on the tape is provided. Various lemmas proved in that section will be intensively used in

the proof of theorem 1 most often without reference. In section 4 a general pattern for proving the halting problem to be decidable when this is the case is described, and specific tools are elaborated to be used in the proof of theorem 1.

Section 5 and 6 are devoted to the proof of theorem 1 : section 5 for machines without stationnary instructions, section 6 for dropping that restriction.

In section 7, the case of a single left instruction is considered. Various extensions of the non-erasing setting are discussed and theorem 3 is there proved.

Notice that the tools elaborated in sections 3 and 4 are general and do not assume any non-erasing condition. In particular, two general results on Turing machines are proved in section 4, which state necessary and sufficient conditions for an ultimately periodical motion of the machine. One of those theorems, periodicity lemma, was already stated in [6]. The second one is a new result of this work.

2. TWO CRITERIA FOR THE HALTING PROBLEM

Remind that here, we consider only deterministic Turing machines with one head and one tape, bi-infinite, on alphabet $\{0,1\}$, 0 taken as a blank symbol.

Instructions of such a Turing machine may be represented by a quintuple (I) $\mathbf{i}\,xMy\,\mathbf{j}$ where, if the machine scans symbol x on the tape in state $\mathbf{i}$ of its finite control, then it rewrites symbol x to y on the tape, moves its head in the direction M (to the right for $M = R$, to the left for $M = L$, no move for $M = S$) and changes its state from $\mathbf{i}$ to $\mathbf{j}$ for performing the next step of the computation. Symbol M is the *move* of instruction (I), and we say, respectively, for non-halting instructions, *left*, *right* and *stationnary* instruction as $M = L$, R or S.

DEFINITION: The triplet xMy is called *colour* of instruction $\mathbf{i}xMy\mathbf{j}$.

The colour of an instruction can also be viewed as a *projection* of the corresponding instruction. Applying this projection to the whole program of a Turing machine, say M, provides a set whose cardinality is, by definition, the number of colours of machine M. The notion of colour that we introduced in [1] and [2], is the simplification of a more complex notion of [9]. In fact, in [9], L. Pavlotskaya shows merely that the number of *colours* over Turing

machines on $\{0,1\}$ provides a criterion with a frontier value for the halting problem of those machines. Reformulated in terms of *colours*, Pavlotskaya's results state : up to *two* colours, the halting problem of the machine is decidable and starting from *three* ones, such an assortment of three colours can be found that a universal Turing machine can be constructed, using only colours of the assortment.

In [2] and in [1], we extend this criterion to the non-erasing case obtaining, of course, another frontier value : up to *four* colours, the halting problem is decidable. Starting from *five* colours, one can find such an assortment of colours that a universal non-erasing Turing machine can be constructed, using only the assortment colours.

Let us turn now to the laterality criterion.

We first define the *laterality* of instruction (I) as follows. If instruction (I) is, respectively, left, right, its laterality is also left, right. If instruction (I) is stationnary, notice that the next instruction to be performed by the machine is completely determined by y and $\mathbf{j}$. Consequently, either a loop of stationnary instructions is eventually performed after running (I), or a halting instruction or an instruction (J) with move R or L occurs, possibly after a sequence of stationnary instructions. In the latter case, we define the laterality of instruction (I) to be the move of (J). The case of an eventual loop of stationnary instructions is a trivial case of non-halting.

It is now possible to define our new criterion as the least number of instructions with the same laterality, for instance with left laterality. Call it the *laterality number* of the considered machine. It comes from the following consideration : in [9], where machines with possibly stationnary instructions are not considered, it is proved that if the program of a Turing machine on $\{0,1\}$ contains a *single* left instruction, but any given number of *right* ones, then the halting problem is decidable for that machine. On the over hand, if we consider the smallest universal Turing machine known up to now on $\{0,1\}$, *see* [15], then we see that a universal Turing machine on $\{0,1\}$ can be constructed if *twenty* left instruction are allowed. The question naturally arises whether this gap can be significantly shortened. The answer is *yes*, since, as indicated in the introduction, the laterality problem has been recently solved in the general $\{0,1\}$ case, see [7], where it is shown that the laterality criterion has *two* as a frontier value. In this paper, we give a precise frontier value between a decidable halting problem and universality for non-erasing machines as is stated by the following two theorems :

THEOREM 1: (cf. [3]) *The halting problem is decidable for any non-erasing Turing machine on* $\{0,1\}$ *with a laterality number not greater than two.*

THEOREM 2: (cf. [4]) *There is a non-erasing universal Turing machine on* $\{0,1\}$*, the program of which contains precisely three instructions with left laterality. Moreover, such a machine with* 218 *states can be constructed.*

In [4], we indicate with full explanations how to construct a universal non-erasing machine on $\{0,1\}$ with precisely *three* left instructions and we give the explicit program of such a machine with only 218 states, which has been checked on a computer. See also [5] for a short account of this construction.

We postpone to section 7 the statement of theorem 3 completely solving the laterality problem since this statement needs to define extensions of the non-erasing notion.

Let us now turn to section 3 for definitions and lemmas used in the sequel.

3. DESCRIBING THE MOTION OF THE MACHINE HEAD ON THE TAPE

As indicated above, the number of *colours* occurring in a Turing program appears to be a criterion with a frontier value for the halting problem of Turing machines on $\{0,1\}$.

The notion of *colour* itself gives an account of what happens *on the tape*, considering the control unit of the machine as a black box. It indicates changes of content in the cells of the tape as well as the moves of the head. It can be viewed as an *atom* of the signal defined by the trajectory of the head on a space-time diagram, as indicated by figure 3, below.

In our sequel, as we focus our attention on alphabet $\{0,1\}$, at least up to section 6, we give particular names to these colours, according to the following table :

$0R0$	:	*right blue*	$1R1$	:	*right green*
$0R1$	:	*right red*	$1R0$	:	*right yellow*
$0L0$	:	*left blue*	$1L1$	:	*left green*
$0L1$	:	*left red*	$1L0$	:	*left yellow*

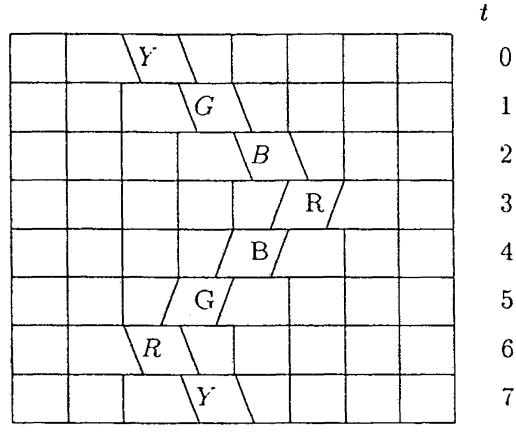
In our sequel too, we shall use a condensed representation of this signal. In order to do this, we have to go into more details of what is a *qualitative* approach of the behaviour of a Turing machine head.

Basic definitions

Following [2], fix a bijection between cells on the tape and elements of $\mathbf{Z}$, calling *address* the image of a cell. Always assume that the right neighbour

of cell c is cell c+1 and that its left neighbour is cell c-1. In this way, any interval $[a, b]$ of $\mathbf{Z}$, a or b possibly infinite, defines an aera of the tape that we call *zone* $[a, b]$. For any $d \in \mathbf{Z}$, let $[a, b] + d$ be $[a + d, b + d]$. In the finite case, we say also *segment* $[a, b]$. Cells a and b are the *ends* of the segment. An infinite zone has at most one end.

Let C_t be the current configuration at time t. Its *support* is defined inductively. For the initial configuration, the support is the smallest segment of the tape containing both data and the scanned cell and outside which the tape is empty. At time t+1, the current configuration support is the smallest segment containing both the previous current configuration support and the scanned cell at time t+1. The *basis* of the current configuration at time t is the smallest segment, possibly empty, outside which all cells are empty at time t. The *current bounds* of the current configuration are the ends of its support. A *desert of* 0*'s* is either of the infinite zones defined by the complement on the tape of the support of C_t. There is, of course, a *right* desert and a *left* one.



As the machine is here assumed to work on alphabet $\{0, 1\}$, colours are represented according to the following encoding :

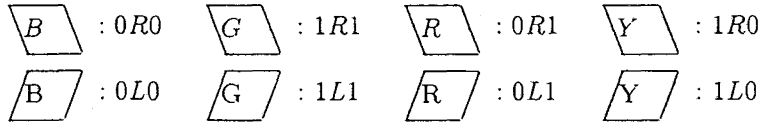


Figure 3. – Example of a space-time diagram.

The given bijection allows to define functions v, σ anf η as follows :

– $v(t)$ is the address of the cell scanned by the machine at time t;

– $\sigma(c,t)$ is the content of the cell with address c at time t ; if $c = v(t)$, $\sigma(c,t)$ is the symbol *scanned* by the machine;

– $\eta(t)$ is the state of the machine when it reads the content of the cell, scanned at time t.

We may, now, state a first property of Turing machines which says, in a sense, that its head cannot *jump* from a position to another one which would not be one of its two neighbours:

INTERMEDIATE CELL LEMMA : *If a Turing machine scans cell b at time t_1 and cell a at time t_2, with $t_1 < t_2$, then it scans at least once any cell c with $c \in [a,b]$ at time t with $t \in [t_1,t_2]$.*

The proof, for instance by induction on $t_2 - t_1$, is left to the reader.

In our sequel, t with or without indices always indicates *time*. Assuming $a \leq b$, define $[a,b]_t$ to be the *word* constituted of $\sigma(c,t)$ for c running over $[a,b]$.

Say that cell c, scanned at time t is a *half-turn* at that time if $v(t-1) = v(t+1)$ with $v(t) \neq v(t+1)$. Specify *left* or *right* half-turn according to whether $v(t+1) > v(t)$ or $v(t+1) < v(t)$. Notice that at a *left* half-turn, the machines goes back *to the right* and, symmetrically, at a *right* one, it goes back *to the left*. It is possible to define a *partial recursive* function which enumerates the half-turns of the machine. From intermediate cell lemma, it follows immediately that between two consecutive half turns the machines always goes in the same direction and that a left half-turn is thus followed by a right one and conversely.

Call *sweeping* a segment $[a,b]$ of the tape, with $a \leq b$, such that either a is a left half-turn at time t and $b+1$ the position of the next half turn, necessary a right one, or b is a right half-turn at time t and $a-1$ the position of the next half-turn, necessary a left one. The initial position of the machine and its possible halting one will also be considered as half-turns in that definition. Say that a segment $[a,b]$, $a \leq b$ is *scanned at maximal speed* if, scanning one end of the segment at time t, the machine scans the other end at time $t+b-a$. From intermediate cell lemma we see that if segment $[c,d]$, $c \leq d$ is contained in sweeping $[a,b]$ with a as left half-turn and c being scanned while the machines goes from a to b, then $[c,d]$ is scanned at maximal speed. An analogous conclusion holds if b is the right half-turn

in sweeping $[a, b]$, the machine going then from b to a : $[c, d]$ is scanned at maximal speed, cell d being scanned first.

If a left half-turn, say c, scanned at time t, is the last half-turn of the machine starting from time t, then any segment with c as left end is scanned at maximal speed starting from time t. This allows us to say that $[c, +\infty[$ is also a sweeping. We define analogously sweeping $]-\infty, c]$, where c is supposed to be a right half-turn, also the last half-turn of the machine.

The notion of sweeping allows us to introduce a *condensed* representation of the motion of the machine head in a space-time diagram. Instead of associating a line with each time t, we associate a single line with each *sweeping* of the machine. In such a way, the space-time diagram of figure 3 becomes :

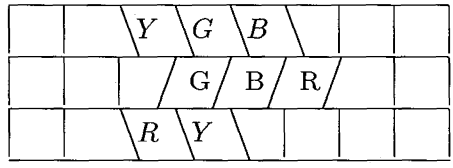


Figure 4. – Previous space-time diagram, condensed version.

An *exit* is a time at which the machine head is outside the current bounds of the just *preceding* configuration. An exit is *extremal* if and only if the bound it sets to the configuration is, at that time, a half-turn. An extremal exit is a *right* one or a *left* one according to what is the corresponding half-turn.

Parallel motion lemmas

In our sequel, we shall intensively use the following lemmas which are, intuitively, sufficient conditions for parallel motions of the machine head.

Let us formulate this more accurately.

DEFINITION: The motion of the machine on $[a, b]$ starting from t is said to be *parallel* to its motion on $[a', b']$ starting from t' if and only if:

(i) $a' = a + b' - b$

(ii) $v(t) = a$ and $v(t') = a'$

(iii) $\forall \tau \geq 0 \ (v(t+\tau) \in [a, b] \ \& \ \forall \tau' \leq \tau \ (v(t+\tau') \in [a, b])) \Rightarrow$
$(v(t'+\tau) = v(t+\tau) + b' - b \ \& \ \eta(t'+\tau) = \eta(t+\tau))$

We have the following sufficient condition for parallel motion:

REPLICATION LEMMA: *Assume that* $a' = a + b' - b$, $v(t) = a$ *and* $v(t') = a'$ *with* $[a, b]_t = [a', b']_{t'}$ *and* $\eta(t) = \eta(t')$. *Then, the motion of the machine on* $[a, b]$ *starting from* t *is parallel to its motion on* $[a', b']$ *starting from* t' *as long as the machine head remains in* $[a, b]$ *starting from* t.

The proof, for instance by induction on t_1 such that for all $\tau \leq t_1$ $v(t + \tau) \in [a, b]$ holds, is left to the reader.

A somewhat similar situation is described by the following lemma, also a sufficient condition for parallel motion:

LEFTMOST POSITION LEMMA: *Assume that there are two times* t_1 *and* t_2 *of right exits,* $t_1 < t_2$ *and an address* a *such that:*

(i) $\forall t \in [t_1, t_2]\ v(t) \geq a$

(ii) *if* $\delta = v(t_2) - v(t_1) \geq 0$, *then* $[a, v(t_1)]_{t_1} = [a + \delta, v(t_2)]_{t_2}$

(iii) $\eta(t_1) = \eta(t_2)$

Then the motion of the machine on $[a, v(t_2)]$ *starting from* t_1 *is parallel to its motion on* $[a + k.\delta, v(t_1 + (k + 1).(t_2 - t_1)]$ *for all* $k \geq 0$.

Indeed, as the basis of induction on k is contained in the induction step, it is enough to prove the lemma for $k = 1$, *i.e.* that the motion of the machine on $[a, v(t_2)]$ starting from t_1 is parallel to its motion on $[a + \delta, v(2.t_2 - t_1)]$ starting from t_2, which is obvious by induction on $t_2 - t_1$. ■

Of course, there is a symmetrical *rightmost position* lemma with an ultimately periodic motion of the machine to the left.

4. GENERAL REDUCTION FOR DECIDING THE HALTING PROBLEM

In order to prove the decidability of the halting problem, as the halting of a Turing machine is definable by a recursively enumerable condition, it is enough to prove that the non-halting also is recursively enumerable

Notice first the following property. Define a segment $[a, b]$ of the tape to be a *trap zone* for machine M if and only if there is a time t_0 starting from which for all $t \geq t_0$ the head of M remains in $[a, b]$. It is well-known that:

Segment $[a, b]$ *can be decided to be a trap zone in at most* $r^L.s + 1$ *steps of the machine computation starting from* t_0, *where* $L = b - a + 1$, r *is the number of letters of the machine alphabet and* s *is the number of the machine states.*

Indeed, if the machine has not yet halted after this time, two indentical configurations must have occured.

We shall later refer to that property as *trap zone lemma.*

Notice, now, that in the non-halting situation, if the head occupies only a finite part of the tape, this leads, by trap zone lemma, to the forthcoming of two identical configurations, which is recursively enumerable. So, we may assume that the head of the machine occupies an infinite part of the tape. And this may be split to its turn into three new cases: either the whole tape is occupied, or only a *right* semi-infinite zone of it or only a *left* semi-infinite zone of it, where a semi-infinite zone is either of the form $[c, +\infty[$ or of the form $]-\infty, c]$, c finite in both cases.

Let us look at these three cases more carefully. One may express them in terms of exits in the following equivalent way:

– either there are infinitely many left exits and infinitely many right exits ;

– or there are only infinitely many left exits;

– or there are only infinitely many right exits.

We shall refer to the first case as the case of infinitely many *traversals* and we shall discuss the behaviour of the machine during traversals from the right to the left and during long enough traversals from the left to the right.

Let us consider, now, the two cases with infinitely many exits only on one side of the tape. We may, for instance, look at the case of infinitely many right exits. Either there are infinitely many *extremal* right exits or there are only finitely many of them. In the latter case, we may assume that starting from a certain time t, there are no more *extremal* exits. Indeed, consider the first exit occurring after time t. By our assumptions, it is a right exit. The next step of the machine is a right exit too. If this would not be the case, this would be a contradiction with our assumption that after time t there are no more extremal exits. So, by induction on the number of computation steps, it is plain that, after the first right exit occurring after time t, the machine goes endlessly to the right. In such a case its motion becomes ultimately periodic, which is thus recursively enumerable.

As, for our general goal, it is enough to prove that the remaining case also is recursively enumerable, we may assume that there are infinitely many extremal exits in the case when the machine head occupies only a semi-infinite zone of the tape.

We shall later on refer to this splitting as the *space splitting of a non-halting situation.*

Before turning to the proof of theorem 1, let us indicate a refinement of the case with only infinitely many extremal right exits.

Let us now look closer at the motion of the machine under this assumption. There is a recursive function S which indicates, for all i, the time of the i^{th} right extremal exit. Call *leftmost position* of the machine between $S(i)$ and $(S(i+1)$ the address of the leftmost cell scanned by its head between those times. Say, in short, *lmp* for leftmost position, and denote it by p_i. Call *lmp time* the first time when the machine head scans the content of cell p_i between $S(i)$ and $S(i+1)$ and denote it by $p(i)$. We say that *lmp* p_i is *absolute* if $v(t) \geq p_i$ for all $t \geq S(i)$. We have then the following necessary and sufficient condition for an ultimately periodic motion of the machine:

PERIODICITY LEMMA ([6]): *Let p_i the sequence of absolute lmp's, S_i the extremal right exit which just occured before lmp time corresponding to p_i and l_i the distance from p_i to $v(S_i)$. The motion of the machine is utimately periodic if and only if:*

$$\liminf_{i \to \infty} l_i < +\infty,$$

and this condition is recursively enumerable.

The condition is obviously necessary since, in view of the ultimate periodicity, all the l_i's are bounded.

Suppose, now, that we have $\liminf_{i \to \infty} l_i < +\infty$. As the l_i's are *non negative* integers, since we assume that there are infinitely many right extremal exits and no left exit, starting from a certain time, their lim inf, say λ, is reached infinitely often. So, in the sequence of those l_i's which are equal to λ, we may pick out two of them, say l_u and l_v with $u < v$, such that, denoting by t_u and t_v the *lmp* times corresponding to, respectively, p_u and p_v, we observe that $[p_u, p_u + l_u]_{t_u} = [p_v, p_v + l_v]_{t_v}$ and $\eta(t_u) = \eta(t_v)$ hold. As p_u and p_v are assumed to be *absolute lmp*'s, the desired conclusion is obtained by applying replication lemma. ■

Of course, a symmetrical notion, namely that of *rmp*, *rightmost position*, is defined for extremal *left* exits. An analogous periodicity lemma holds for that case: in the above statement, replace *lmp* by *rmp* and *extremal right exit* by *extremal left exit.*

A refinement of this situation can be formulated:

STRONG PERIODICITY LEMMA: *Let p_i the sequence of absolute lmp's, S_i the extremal right exit which just occured before lmp time corresponding to p_i and l_i the distance from p_i to $v(S_i)$. The motion of the machine is utimately*

periodic if and only if:

$$\liminf_{i\to\infty}(l_i + p_i - p_{i+1}) < +\infty,$$

and this condition is recursively enumerable.

The condition is obviously necessary since, in view of the ultimate periodicity, all the l_i's and $p_{i+1} - p_i$'s are bounded.

Suppose, now, that we have $\liminf_{i\to\infty}(l_i + p_i - p_{i+1}) < +\infty$. This means that there is an integer C such that $p_{i+1} \geq p_i + l_i - C$ infinitely often. There are at least two indices, say u and v, such that $p_{u+1} \geq p_u + l_u - C$, $p_{v+1} \geq p_v + l_v - C$, $p_{u+1} - p_u - l_u = p_{v+1} - p_v - l_v$, $[l_u - C, l_u]_{t_u} = [l_v - C, l_v]_{t_v}$ and $\eta(t_u) = \eta(t_v)$ where t_i is the first time after $p(i)$ (lmp-time of p_i) when the machine head scans cell $p_i + l_i$. By definition of lmp's, among lmp's occuring on the left of p_{i+1} inbetween t_i and $p(i+1)$ if any, at least one would be absolute which would be contrary to the *absoluteness* of lmp p_{i+1}. Consequently, the head never goes to the left of $l_u - C$ after t_u, nor to the left of $l_v - C$ after t_v. Leftmost position lemma applies to the situations at times t_u and t_v. Hence the ultimate periodicity of the machine motion. As it is clear that the just described situation is recursively enumerable, the lemma is proved. ■

5. PROOF OF THEOREM 1

As indicated in the introduction, we assume, in this section, that the considered non-erasing Turing machine on $\{0,1\}$ has no stationnary instruction.

The proof consists in splitting the problem of the non-halting into cases which will turn out to be recursively enumerable.

In a first step, we reduce the problem to the case when one of the left instructions is *blue* and the other is *green*. Then we prove some preliminary lemmas which allow, in particular conditions, to characterize motions of the machine to the left and how such a motion halts. Afterwards, the proof is split into cases according to the relations between the exit state of one left instruction and the entry state of the other one, which we combine with the space splitting of a non-halting situation. It will be seen that almost all cases are most often characterized by ultimate periodic motions to which periodicity lemma or replication lemma apply, and some of them are based on a periodic lenghtening of the space which the machine head runs over. A last case, more difficult, will be dealt with apart. In the course of the proof,

several *types of motion* for the machine head will be encountered. All these types are below illustrated in table 1 which will be used for further references, avoiding tedious repetitions. Figures of table 1 represent motions according to their successive sweepings in a way which needs no further explanation.

First reduction of the problem

A first, drastic reduction is made by the following considerations.

On alphabet $\{0,1\}$, there are at most eight colours in any program of a Turing machine. If we assume the machine to be non-erasing, then two colours are ruled out: left and right *yellow* which both replace a 1 by 0. We call *pure motion colour* any colour among the following four ones: left and right *green*, left and right *blue*. Notice that under these colours the machine reads the tape without changing the information, written on it. We proved in [1], [2] the following property, from which it is easily deduced that the halting problem is decidable for a non-erasing machine with at most four colours:

THEOREM A: *The halting problem is decidable for any non-erasing Turing machine on* $\{0,1\}$ *in the program of which at least one pure motion colour is missing.*

Let us call an instruction *stable* if its exit state is the same as its entry state, *unstable* if this is not the case. We then obtain 21 cases of couples of left instructions according to their colour and their stability, as illustrated on figure 5, below.

As among these 21 cases, one left pure motion colour is missing in 17 of them, we deduce from theorem A that the halting-problem is decidable in these cases. And so, four cases remain to be examined, all with both left colours *blue* and *green*.

We turn, now, to particular situations which will give us new tools for the cases into which the proof is split.

Preliminary lemmas

Intuitively, what is the most important is to see the possible *connections* between both left instructions of the machine. We say that instruction I *calls possibly* instruction J, or short *calls* instruction J, if the exit state of instruction I is the entry state of instruction J. For instance, a stable instruction calls itself. We shall often specify 'instruction I calls instruction

TABLE 1. – *Types of non-halting motions.*

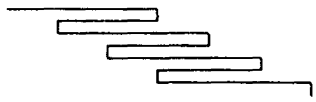


fig. *a*. simple right motion

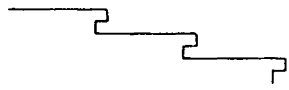


fig. *b*. degenerated right motion

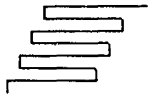


fig. *c*. simple left motion

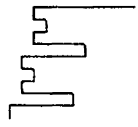


fig. *d*. two-folded left motion

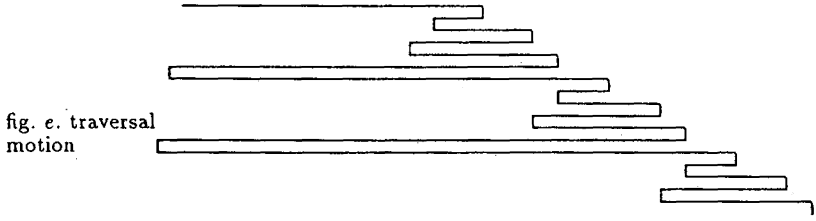


fig. *e*. traversal motion

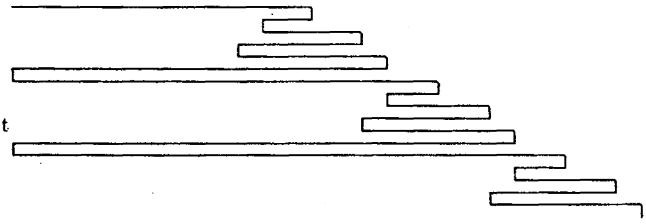


fig. *f*. fixed-point motion with right rebounds

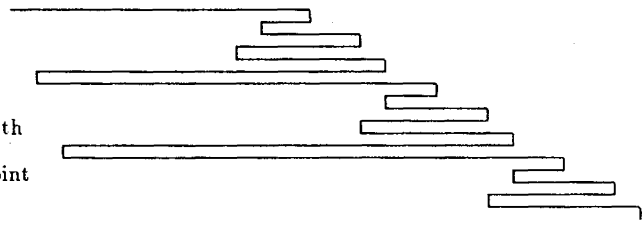


fig. *g*. motion with right rebounds without fixed-point

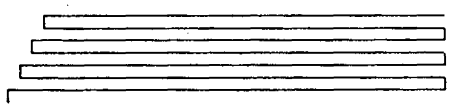


fig. *h*. fixed-point motion with left rebounds

J if a β is scanned', in order to say that instruction J is the next instruction performed by the machine after instruction I if symbol β is scanned at that time, with β either 0 in both cases or 1.

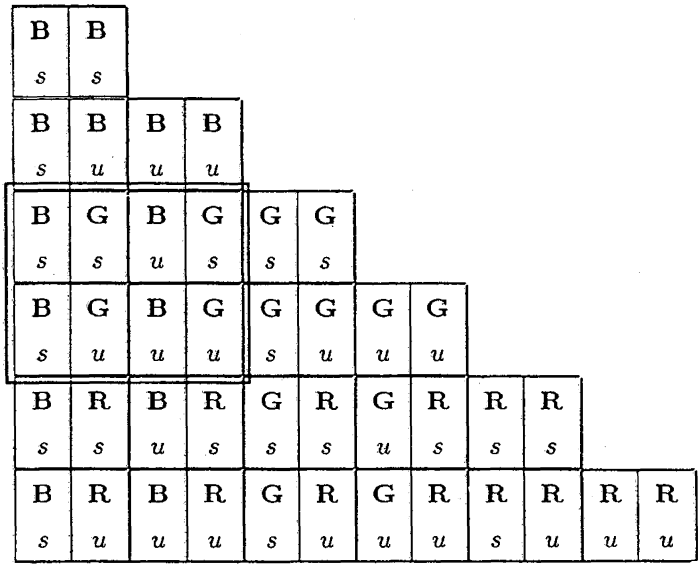


Figure 5. – The 21 cases of the problem, within the frame, the 4 difficult ones

If both left instructions of the machine call necessarily a *right* instruction, it seems intuitively clear that the machine *cannot* go to the left of the cell where the right instruction was applied. We shall now prove this carefully since it plays a great rôle in our further argumentation.

SINK LEMMA: *Assume that the non-erasing machine has precisely two left instructions, one of them blue, the other one green. Assume that scanning cell c at time t, a left instruction is performed and that this instruction calls necessarily a right one. Then, if the machine has not already halted, either it performs an infinite back-and-forth motion on cells c−1 and c, or, at most* 5 *steps after time t, the machine head reaches cell c+1 without being on the left of cell c−1 starting from time t. Either of these situations is recursively enumerable.*

The proof of the lemma is split into the cases which may occur under these assumptions. We may condense them into the patterns of figure 6, below.

The property claimed by the lemma comes from the following fact. After the first application of the considered left instruction on cell c the first *right* half-turn performed afterwards on that cell, if any, is necessarily performed by an instruction with the same colour. But there is a single left instruction

with this colour, and so the right instruction which is next performed is completely determined by the symbol scanned on cell $c-1$.

Let us call *sink* either of the six cases, when segment $[c-1, c]$ is a trap zone starting from time t, represented by the first three patterns of figure 6. We shall often use the lemma in the following way: as, scanning cell c, instruction X calls necessarily a right one on cell $c-1$ and as the machine cannot fall into a sink – this is the case for an infinite motion to the right – its head scans cell $c+1$ without having scanned any cell on the left of $c-1$.

In particular, if the instruction called after that a left one has been executed, say on cell c, is necessary a right one, the machine head cannot go to the left of $c-1$: if a left instruction is performed on cell c and if the machine does not fall into a sink, then the machine scans cell $c+1$ without having scanned any cell on the left of $c-1$. Then, by induction on $d \geq c+1$, we see that on cell d, either a right instruction is performed, and the machine goes one more step to the right, or a left instruction is performed and, if there is no possible sink, then the machine goes to the left of d, without having scanned any cell to the left of $d-1$, and so it goes too one more step to the right.

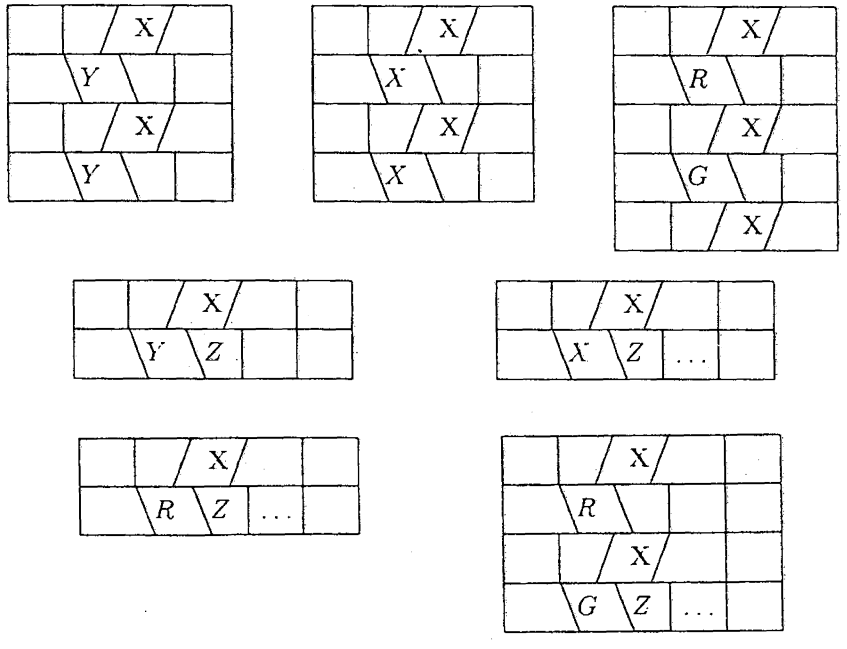


Figure 6. – The cases of sink lemma.

In these figures, letters X and Y represent B or G with $X \neq Y$. Letter Z represents any non erasing right instruction. The dots represent left instructions or right ones, as well.

Here and later, we shall state several lemmas characterizing motions to the left of the machine according to the relations between the states of the left instructions. We shall name them according to the status of the left instructions, using the following shortening, that the reader will immediately understand: $\mathbf{B}_s$, $\mathbf{B}_u$, $\mathbf{G}_s$ and $\mathbf{G}_u$.

Notice that if there is an extremal right exit on cell c at time t, then, necessarily, the instruction performed at that time is the *left blue* one.

We have the following property:

B_s-LEFT MOTION LEMMA: *Assume that the machine cannot fall into a sink, that the left blue instruction of the machine is stable and that it calls a right one if a 1 is scanned. Then, if the left blue instruction is performed on cell b at time t, if the machine head reaches cell a at time $t+T$, $b-a>2$ and if $v(t+\tau) \in [a,b]$ for all $\tau \in [0,T]$, then $[a+1,b]_{t+T} \in 0^*$.*

Proof: By the assumption, b contains 0 at time t and at time $t+1$. Let t_1 the next passage of the machine head on b during $[t,t+T]$, if any. Then, as there is a 0 in b at time t_1 and as b is a right half-turn for the machine (it never goes on the right of b during $[t,t+T]$), the left blue instruction is performed at time t_1. So there is still a 0 in b at time t_1+1. The same argument repeated for each half-turn of the head in b shows that there is a 0 in b at time $t+T$.

Consider now c the rightmost 1 in [a+1,b-1] at time $t+T$, if any. We have then 0 in $c+1$ at time $t+T$, but also at any time between t and $t+T$ *in view of the non-erasing assumption.* Let t_2 the last time at which the machine scans cell $c+1$ during $[t,t+T]$. The machine goes then from $c+1$ to c by the left blue instruction. If there were a 1 in c at time t_2+1, by sink lemma the machine head would reach $c+1$ at time at least t_2+2. And so, there is a 0 in c at time t_2+1. As t_2 is the last time at which the machine scans $c+1$, by intermediate cell lemma, the machine goes to the left of c at time t_2+1, and so there is still a 0 in c at time t_2+1. But then, by intermediate cell lemma, c is a right half turn for the machine during $[t_2+1,t+T]$, if the machine ever scans this cell again during that time interval. And so, we may repeat for c the argument used above for b. Consequently, there is a 0 in c at time $t+T$, which is contrary to the definition of c. ■

EXTREMAL RIGHT EXIT LEMMA: *Assume that the left blue instruction of the machine is stable and that this instruction calls a right one if a 1 is scanned. Then, if the machine head scans a 0 at a time t of right extremal exit and is on*

the right of a 1 written on the tape, the machine remains for ever to the right of this 1, falling into a sink or performing a periodic motion to the right starting from the time when it reaches first the rightmost 1 on the tape after time t.

Proof: Let a be the address of the rightmost 1 on the tape. By the previous lemma, the left blue instructions is performed until cell a is reached by the machine head, say at time t_1. A right instruction is then performed. The sweeping to the right leads to a new right half-turn, say on cell b. Let c be the position of the rightmost 1 written on the tape when the machine reaches the new right half-turn. We have $a \leq c < b$. On the half turn, the left blue instruction is necessarily performed. And so, the machine goes back to c, reached at time t_2, where the same right instruction is performed as previously. As on the right of c at time t_2 the tape is empty as it is on the right of a at time t_1, the motion of the machine during $[t_1, t_2]$ is parallel to its motion during $[t_2, 2.t_2 - t_1]$. ■

NO TRAVERSAL LEMMA: *Assume that the left blue instruction of the machine is stable and that it calls a right one if a 1 is scanned. Then, if the machine does not halt and occupies an infinite zone of the tape, it cannot be the whole tape.*

Proof: Indeed, the traversal from right to left starting from an extremal right exit to the consecutive extremal left exit (such an order between extremal exits does occur if there are infinitely many traversals) would be performed under the left blue instruction. By stable blue instruction lemma, as the left blue instruction is stable, either the machine head goes forever to the left, always scanning 0's and thus, occupying a left semi-infinite zone of the tape, or the motion of the machine to the left halts on a 1 on which a right instruction is called. The machine remains for ever to the right of this 1 by the previous lemma. Hence the conclusion of the lemma. ■

We may deduce, from the proof of extremal right exit lemma, that under the assumptions of no traversal lemma, the motion of the machine is an ultimately periodic one to the right if a 1 stands to the left of the machine head. Its description is very simple: the period is the time interval between two consecutive *lmp*'s, since the motion of the machine consists in alternate sweepings involving only the left blue instruction and the same finite sequence of right instructions. This motion is illustrated on table 1, figure *a*. We shall later on say simple right motion.

Let us notice the following easy corollary of sink lemma:

G_u LEFT EXIT LEMMA: *Assume that the machine cannot fall into a sink, that the left green instruction of the machine is unstable and that it always calls a*

right instruction. Then, if the left green instruction is performed on cell a at a time t of left exit, the machine never goes to the left of cell $a-1$ after time t.

We shall now see in detail the behaviour of the machine according to the four possible relations between stability and unstability with respect to the blue and green left instructions.

Case blue stable and green stable

In that case, the left instructions may be written, respectively $\boxed{\mathbf{i}0L0\mathbf{i}}$ and $\boxed{\mathbf{j}1L1\mathbf{j}}$. Notice that if $\mathbf{i} = \mathbf{j}$, we have either a constant motion of the machine to the right or, if any of the left instructions is called, a constant motion of the machine to the left. In both cases, the motion is trivially ultimately periodic and, of course, this is a recursively enumerable non-halting. Later on, we shall not consider such motions since they are trivially cases of a recursively enumerable non-halting. We shall focus our attention only on the three cases of infinite motion in space involving infinitely many extremal exits. As it will soon be seen, the only possible motions of the machine are those which are illustrated in table 1, in figures a and c.

And so, we may assume that $\mathbf{i} \neq \mathbf{j}$. Consequently, no traversal lemma applies to this case, which rules out the possibility of infinitely many traversals.

If there are only infinitely many extremal right exits, then, as $\mathbf{B}_s$-left motion lemma applies, by the discussion following no traversal lemma, we know that the motion of the machine is a simple right motion, as illustrated in table 1, in figure a, and that it is recursively enumerable.

And so we have only to study the case of infinitely many extremal left exits without right exit.

Consider the rmp which follows the first extremal left exit. As we assume a non trivial case, only the left green instruction is called during any left exit, which is, by the way, necessarily extremal. If a right blue instruction is performed just after the left green one, then, as the right neighbour of this 0, say at address a, is a 1, by intermediate cell lemma, the machine must first reach that 1 before this 0 if we want it goes to the left of a. But in order to make the head go from $a+1$ to a, the left green instruction is called and so the same right instruction as previously is called when the head scans cell a. As this rules out further exits, the green left instruction calls a right red one if a 0 is scanned.

As we suppose that there is no right exit, there is at least one absolute rmp, say on cell b. If there is a 0 on b at rmp time, then as necessarily the

blue instruction is called, then, either there is a sink, either the machine head goes to the right of b *after* rmp time, which is contrary to the absoluteness of the considered rmp. And so there is a 1 on b. Consider now the motion of the machine from b, starting from rmp time until the next extremal left exit. The following property allows to characterize this motion:

G_s-LEFT MOTION LEMMA: *Assume that the machine cannot fall into a sink, that the left green instruction of the machine is stable and that this instruction calls a right one if a 0 is scanned. Then, if the left green instruction is performed on cell b at time t, if the machine head reaches cell a at time* $t+T$, $b-a>2$ *and if* $v(t+\tau)\in[a,b]$ *for all* $\tau\in[0,T]$, *then* $[a+1,b]_{t+T}\in 1^*$.

Proof: By the assumption, b contains 1 at time t and so, this is also the case at time $t+T$ by the non-erasing assumption. Consider now c the rightmost 0 in [a+1,b-1] at time $t+T$, if any. So, there was a 0 too in c at time $t+\tau$ for all $\tau\in[0,T]$. By assumption on $c+1$, a 1 is there at time $t+T$. If t_1 is the last time at which the head scans $c+1$ during $[t,t+T]$, there is also a 1 on cell $c+1$ at time t_1+1 and at time t_1 too. Indeed, the left blue instruction could not be performed for going from $c+1$ to c, since a 0 would be left in $c+1$, and so the left green instruction made possible this move from $c+1$ to c and so there was also a 1 in $c+1$ at time t_1. But then, a right instruction is called as the machine scans cell c. By sink lemma, as there is no sink by assumption, the machine goes to the right of $c+1$ after time t_1+1, which is contrary to the definition of t_1. So there is a 1 in c at time t_1+1 and, consequently, at time $t+T$ too, by the non-erasing assumption. So there is no 0 in [a+1,b] at time $t+T$. ■

Consequently, when the machine head reaches cell a where occurs the left half-turn of the extremal left exit at time t_1, $[a+1,b]_{t_1}\in 1^*$. The right red instruction which is then performed leaves a 1 on cell a, which allows the machine to go one more step to the left for the next extremal left exit.

As cell b is an absolute rmp, the sweeping to the right, starting from cell b ends on a right half-turn on cell c with $c\leq b$ at time t_1'. It is plain that starting from that time, as the left green instruction is performed, the corresponding sweeping to the left goes downto cell $a-1$, reached at time t_2. Now, it is easy to see that the assumption of rightmost position lemma is satisfied, which, by parallel motion, entails a periodic motion starting from time t_1, the period of which is time interval $[t_1,t_2-1]$

We call this motion to the left, simple left motion as illustrated in table 1, figure c.

Case blue stable and green unstable

This second case is much similar to the previous one in many respects, and it will be seen that in this case, simple right motions are the only possible motions requiring infinite space.

As the left blue instruction is stable, what has been said in the previous case about left exits and traversals from an extremal right exit to an extremal left one holds here too. And so, in order to avoid trivialities, the only infinite motion to which we have to turn now is the case of infinitely many extremal right exits without left exit.

So, the left instructions are $\boxed{\mathbf{i}0L0\mathbf{i}}$ and $\boxed{\mathbf{j}1L1\mathbf{k}}$ with $\mathbf{k} \neq \mathbf{j}$. The case splits into three subcases according to the relations between **i**, **j** and **k**.

Case $\mathbf{i} \neq \mathbf{j}$, $\mathbf{i} \neq \mathbf{k}$:

In that case, stable blue instruction lemma applies and so we have a simple right motion as illustrated in table 1, fig. *a*.

Case $\mathbf{i} = \mathbf{j}$, $\mathbf{i} \neq \mathbf{k}$: $\boxed{\mathbf{i}0L0\mathbf{i}}$ and $\boxed{\mathbf{i}1L1\mathbf{k}}$

In this case, we cannot apply stable blue instruction lemma directly since the left blue instruction calls the green one if a 1 is scanned. But we have an *extended* $\mathbf{B}_s$-left motion lemma.

Consider again the proof of $\mathbf{B}_s$-left motion lemma.

As previously, it is proved that, at time $t + T$, there is a 0 on cell b. Let again c be the rightmost 1 in $[a, b-1]$ at time $t + T$. In $c + 1$ there is a 0 which, consequently was already there during time interval $[t, t + T]$. Let t_1 the last time at which the head scans cell $c + 1$. The head goes then from $c + 1$ to c by the left blue instruction. If there were a 0 on c at time $t_1 + 1$, as the machine never goes to the right of c during time interval $[t_1, t + T]$, c would be a right half-turn, and so only the left blue instruction could be performed on that cell, and so the 0 would remain there until time $t + T$, which is contrary to the definition of c.

So, there is a 1 in c at time $t_1 + 1$. The left green instruction is then performed, but as it calls necessarily a right instruction, as there is no sink, by sink lemma, the machine goes on cell $c + 1$ *after* time t_1. This is impossible, unless cell $c - 1$ is cell a and cell a is reached at time $t + T$.

Consequently, when an extremal right exit happens, the left blue instruction is performed until the rightmost 1 on the tape is reached. On this 1, the left green instruction is performed but, after at most five steps of computation, the machine head is again on the right of this one. Extremal right exit lemma thus applies too in this case, and so we have a simple right motion.

Case i = k, i ≠ j: $\boxed{\mathbf{i}0L0\mathbf{i}}$ and $\boxed{\mathbf{j}1L1\mathbf{i}}$

In that case, as the left blue instruction calls a right one if a 1 is scanned, $\mathbf{B}_s$-left motion lemma applies too, and so we obtain a simple right motion.

Case blue unstable and green stable

The left instructions of the machine are now, respectively, $\boxed{\mathbf{i}0L0\mathbf{k}}$ and $\boxed{\mathbf{j}1L1\mathbf{j}}$ with $\mathbf{k} \neq \mathbf{i}$. The case again splits into three subcases according to the relations between **i**, **j** and **k**. But, as we have here a left green stable instruction, the behaviour of the machine may be different from what have previously be seen. We shall see that this is the case in several subcases. All motions illustrated on table 1 are here possible, but the motion represented in figure g. In particular, in one of the cases, infinitely many traversals are possible.

Case j ≠ i, k ≠ j:

Let us notice that, in this case, if there is an extremal right exit, say on cell a, the left blue instruction is performed and as it always calls right instructions, by sink lemma, as there are no sink, the machine goes to the right of a, where begins a desert of 0's. This shows that *lmp*'s are always absolute and that if l_i is the distance between *lmp* p_i and the position of the extremal right exit which defines it, then we have $l_i = 1$ and so the assumptions of periodicity lemma are satisfied. The motion of the machine is thus an ultimately periodic one to the right. We shall say that this motion to the right is degenerated right motion, as illustrated in table 1, figure b, and it is, of course, recursively enumerable.

Notice that we could have also applied periodicity lemma for verifying the case of infinitely many extremal right exits without left exit when the left blue instruction is stable. Indeed, as the motion to the right starting from an *lmp* to the next extremal right exit consists in a single sweeping, the half-turn of the extremal exit must occur not later than s steps after the *lmp*, where s is the number of states of the machine. If this were not the case, the sweeping to the right would go on periodically. And so, we have always $l_i \leq s$. Consequently, periodicity lemma applies.

But, in that case, parallel motion lemma gives directly the characterization of the motion which simply consists in alternate sweepings between extremal exits and *lmp*'s.

Consequently if there is at least one extremal right exit, there are no more left exits. And so the case of infinitely many traversals is ruled out.

Consider, now, the case of infinitely many extremal left exits without right exit.

Notice, that in fact, any left exit is extremal: the left blue instruction always calls right ones and the left green one calls a right instruction if a 0 is scanned.

We first prove that there is a time after which any left exit is performed by the left green instruction.

Indeed, suppose that a certain extremal left exit is performed by the left blue instruction, say on cell a. As there is no sink, the machine head goes at most five steps later on cell $a + 1$, say at time t.

Assume that, at time t, there is a 0, either on cell a-1 or on cell a. If cell a contains 0, the machine goes to a-1 by the help of the left blue instruction and again, as there is no sink, the machine goes to a+1 at time t_1. If cell a still contains 0, then whatever be the content of cell a-1, it is plain that the machine will never go to the left of a-1.

As this is impossible, cell a contains 1 at time t. If cell a-1 contains 0, the next time when the machine scans cell a for going to cell a-1, the left green instruction is called and, as cell a-1 contains 0 and as we assume that there is no sink, the machine goes to a+1 at time t_1. If cell a-1 still contains 0 at time t_1, the just produced argument shows that the machine can no more go to the left of a-1.

As this is impossible, we may assume that cells a and a-1 contain 1 at time t. And so, after this time, any left exit is performed by the left green instruction.

It is now easy to see that we may apply stable green instruction lemma starting from rmp time at the first absolute rmp which must be occupied by a 1 after this time: if a 0 were on that place, the left blue instruction, unstable, would be performed, and as it always calls a right instruction and as there is no sink, the machine head would go to the right of the rmp, which would be contrary to the assumption of absolute rmp. Later on, the same argument as in the case of stable blue and green left instructions goes on. And so, the motion is a simple left one as illustrated in table 1, figure c.

Case k = j, j $\neq$ i: $\boxed{\mathbf{i}0L0\mathbf{j}}$ and $\boxed{\mathbf{j}1L1\mathbf{j}}$

In this case, the left blue instruction calls the left green one if a 1 is scanned. As the left green instruction always calls right ones, any motion to the left of the machine is of the type described by $\mathbf{G}_s$-left motion lemma or the motion begins by performing the left blue instruction on a 0, then, meeting a 1, the left green instruction is called and starting from this point, $\mathbf{G}_s$-left motion lemma applies to the situation.

As we shall see below, in that case, the basic three kinds of non-trivial infinite motions are possible, namely those illustrated in table 1 in figures e, f and h.

Cases of at least infinitely many extremal right exits

In this case, the same arguments can be used to deal with both situations when there are infinitely many extremal right exits: the case of infinitely many traversals and the case when after a certain time there are no more left exits. We shall later say no-traversal case for that latter case. The intuitive justification of that treatment is the following.

When the left green instruction is under action on a zone of 1's, the motion to the left is stopped if and only if the head scans a 0, as that instruction is stable. Call c the address of the cell where that rightmost 0 is scanned by the machine head. Notice that in such a situation, the same right instruction is called by the left green instruction. There are two possibilities. Either the right instruction is a blue one, or it is a red one. If it is a blue one, symbol 0 remains for ever in cell c which, by the non-erasing assumption and by intermediate cell lemma, is only reached by its right side while the left green instruction applies. This is the reason why the machine cannot later go to the left of c. As we shall later see, this point is then a *fixed* absolute *lmp* If the right red instruction is called, the machine possibly goes back to the right, but the zone of 1's has then increased by one cell.

As we shall soon see, in both cases, we have a *rebound* of the machine head against that position which is always an *lmp* in the case we are studying. Indeed, in the case of traversals, the extremal left exit from which a traversal from left to right starts, is an *lmp* between the right extremal exit from which the previous traversal started and the right extremal exit at which the considered traversal ends.

Let us now make things more accurate.

Say that the motion of the machine starting from extremal right exit, say at time s_i with $a_i = v(s_i)$, until position a_i is reached again before the next right exit (not necessarily an extremal one) is a *rebound* if that motion consists of two sweepings: one from a_i to *lmp* p_i defined by s_i, the second, from the *lmp* up to a_i again, or further on the right of a_i. Say that $l_i = a_i - p_i$ is the *length* of the rebound.

We claim that in both cases, starting from a certain time, the motion is a sequence of rebounds with a regular pattern in the increasing of their lengths with also possible 'local' motions to the right between consecutive rebounds. These situations are illustrated in Table 1, in figures e and f.

Before turning to the proof, let us see that, for instance, the case of infinitely many traversals is possible. Such an example is given by the

following Turing program:

	0	1
1	$1R2$	L
2	$1L1$	R

In this table, following Minsky's notation, see [8], any omitted symbol in output triple $yM\mathbf{j}$ of instruction $\mathbf{i}xMy\mathbf{j}$ is identical to the corresponding input symbol, x or $\mathbf{i}$.

an execution of which is given in figure 7, below:

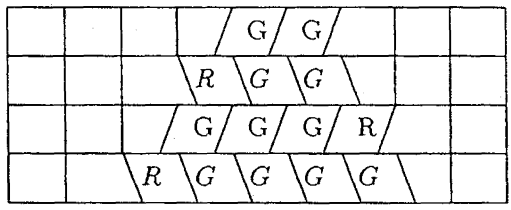


Figure 7. – Infinitely many traversals: execution of the above program

starting from the following initial configuration: ...00011000... △

Indeed, figure 8, above, is a particular case of figure e given in table 1. However, it rather faithfully shows the general behaviour of a machine for which a non-halting situation with infinitely many traversals does happen.

Notice, now, that in the case of no traversal, there is a fixed absolute lmp. Indeed, in that case, as the right instruction called by the left green one when a 0 is scanned, is a blue instruction, lmp's occur always on a 0. If this happens on infinitely many different points which are then absolute lmp's as it was already noticed, then necessarily, $p_{i+1} > p_i + \ell_i$. Strong periodicity lemma then applies and so, such a non-halting is recursively enumerable. And so, putting aside that solved situation, we may assume that there are at most finitely different points as absolute lmp's and so, starting from a certain time, there is a fixed absolute lmp.

Consider a traversal in the case of infinitely many traversals or a sweeping to the fixed absolute lmp in the case of no traversal. The motion of the machine starts from an extremal right exit, say at address a_i, and its sweeping to the left ends on lmp p_i, which is also the next extremal exit in the case of traversals. The first instruction performed during that sweeping is the left blue one which calls the left green instruction. When the lmp occurs, say at

time t_i, there is a segment of 1's on the tape outside which both deserts of 0's are lying. The right instruction called on the lmp is always the same. As we already know, it is a blue one in the case of no traversal. It is therefore a red one in the case of infinitely many traversals.

Consider, now, the sweeping to the right starting from time t_i. If this sweeping would end on a half-turn with an address b satisfying $b \leq a_i$, it is easy to see that a periodic motion to the left, a simple left motion as illustrated in table 1, figure c would occur in the case of infinitely many traversals. In the case of no traversal, the head would for ever remain within the segment $[p_i, b]$ which would thus constitute a trap zone for the machine head. As those situations cannot be the case, the sweeping ends on a half-turn with address b such that $b > a_i$. As a consequence, the considered motion of the machine is a rebound.

Notice now that if the machine scans cell a_i under state **j**, a new sweeping to the left is going on until the next lmp time is reached.

Let ℓ_i be the length of that rebound. Since we assume that there are infinitely many right exits, the above remark shows that the length of the rebounds must increase and tend towards $+\infty$. It is plain in the case of infinitely many traversals, because of the non-erasing assumption. It is also clear in the case of no traversal as we consider the case of a fixed absolute lmp. In both cases, ℓ_i characterizes the rebound from which it is the length.

Let now ℓ_i be the lengths of the successive rebounds in the motion of the machine. When ℓ_i is big enough, the sweeping to the right of the rebound becomes ultimately periodic and it is always the same motion, since the same right instruction is called when lmp p_i is reached. Let p be the length of the shortest period of that motion. With each ℓ_i, associate $\ell_i \bmod p$. As the number of the corresponding possible values is finite, there are two rebounds, say ℓ_u and ℓ_v such that $\ell_u \equiv \ell_v \bmod p$. Let t_u and t_v be the times when the sweeping to the right of the rebound reaches again a_u, respectively a_v. The condition $\ell_i \equiv \ell_u \bmod p$ entails that $\eta(t_u) = \eta(t_v)$. It is then plain that the motion of the machine starting from t_u is parallel to its motion starting from t_v as long as the head of the machine remains on the right of a_u-1. In particular, when the head reaches a_α-1 for the first time after t_α, say at time t'_α, $\alpha = u$ or v, we have a 1 in a_α-1 and, by parallelism, $\eta(t'_u) = \eta(t'_v)$. The state of the machine is then the state of the left green instruction as both left instructions have the same exit state. New rebounds are in fact in action, which are ℓ_{u+1} and ℓ_{v+1}, the next rebound after, respectively, ℓ_u and ℓ_v. But, by parallelism, and as consecutive lmp's are

on the same position in the case of no traversal, or shifted by on cell to the left in the case of infinitely many traversals, we get $\ell_{u+1} - \ell_u = \ell_{v+1} - \ell_v$. Consequently, $\ell_{u+1} \equiv \ell_{v+1} \bmod p$. By induction, it is now easy to conclude that if we observe two traversals ℓ_u and ℓ_v such that $\ell_u \equiv \ell_v \bmod p$, we shall have $\ell_{u+k} \equiv \ell_{v+k} \bmod p$ and $\ell_{v+k+1} - \ell_{v+k} = \ell_{u+k+1} - \ell_{u+k}$. As we may assume that $u < v$, we get then that

$$(*) \qquad \ell_{u+k} = \ell_u + [k/(v-u)].\sum_{k=u}^{v-1}(\ell_{k+1} - \ell_k) + \sum_{k=u}^{u+r-1}(\ell_{k+1} - \ell_k),$$

where $k = [k/(v-u)].(v-u) + r$, with $0 \leq r < v-u$, and $[x]$ satisfies $[x] \leq x < [x]+1$ for any rational x.

Notice that in the case of no traversal, when the machine head is on the right desert of 0's after it has just performed a rebound, it only writes 1's on the tape until it goes back to the left. If this would not be the case, any 0 written outside the rebound segment would be an absolute *lmp*, which is impossible, according to our assumption. In the case of infinitely many traversal, the head may write a few 0's on the right side of the rebound segment. This gives rise to local *lmp*'s which are one by one erased by each sweeping to the left.

Notice that p is easily determined in the following way. Let **q** be the exit state of the right instruction called by the left green one if a 0 is scanned. Then, start the machine on the left end of a segment of 1's with length $2s$, where s is the number of states of the machine, under state **q**. The machine is then in the same conditions as at the beginning of the sweeping to the right which characterizes the rebound of length ℓ. It is plain that the period is at most s and that the aperiodic part of the motion needs at most s steps.

It is now not difficult to describe completely the motion of the machine. If τ_u is the *lmp*-time corresponding to p_u, and so $t_u = \tau_u + \ell_u$, formula $(*)$ allows to compute the further times of right extremal exits, provided that all the distances $\ell_{u+k+1} - \ell_{u+k}$ are known for $k \in \{0, \ldots, v-u-1\}$. From that observation, it is easy to compute $v(t)$ for $t \geq \tau_u$: we leave that to the reader. Those considerations also entail that the observation of $\ell_u \equiv \ell_v \bmod p$ is a sufficient condition for ensuring that the motion of the machine will endlessly go on. That condition on ℓ_i's is clearly recursively enumerable.

This proves that with such left instructions, the case of infinitely many traversals is recursively enumerable as well as the case of infinitely many extremal right exits without left exit.

Let us call *traversal* the just described motion in the case of infinitely many traversals. It is illustrated in table 1, figure e. Let us call *fixed-point motion with right rebounds* the described motion in the case of no traversal, an illustration of which is given in table 1, figure f.

Case of infinitely many extremal left exits without right exit

In that case, the possible motions of the machine are somewhat symmetrical to those of the previous case.

Notice first that the right instruction called by the left green instruction is necessarily right red. If this were not the case, there would be at most a single extremal left exit, which is contrary to the assumption.

Consider the rmp occurring after the first extremal left exit. We have two cases, according to the content of the rmp cell at rmp time. If it is a 1, then it is plain that a simple left motion does occur.

Suppose, now, that it is a 0. If the next rmp is occupied by a 1 at rmp time, it is clear that, starting from this time, a simple left motion appears. And so, we may assume that all rmp cells are occupied by a 0 at rmp time. Consequently, the left blue instruction is called at that time, and so the 0 remains at this place. Consequently, if we focus our attention on absolute rmp's, which exist since there is no right exit, the corresponding cells always contain 0.

Let again p_i be the positions of the absolute rmp's and l_i the distance between p_i and the position of the latest extremal left exit before its rmp time. Let $a = p_1$, the position of the first absolute rmp.

Starting from rmp time, the machine head goes to the left since, on a, the left blue instruction is called. If cell a-1 contained a 0 at rmp time, by sink lemma, the machine would go to the right of a, which is contrary to the assumption of absolute rmp. And so, there is a 1 in a-1 at rmp time. Consequently, after it has been performed on a, the left blue instruction calls the left green one which is performed on a-1 and, starting from that place, downto the next extremal left exit which is by a single cell to the left of the former extremal left exit since the left green instruction calls a right one if a 0 is scanned.

Consequently, we have, typically a *rebound.* Starting from this new extremal left exit we have again a rebound up to a since a simple left motion is excluded by assumption, and, for this new rebound, we have $l_2 = l_1 + 1$. This argument can be repeated and so, by induction, we obtain

that the motion from one extremal left exit to the following one is a rebound up to a, that all rmp's are absolute, that $p_i = a$ for all i and that $l_{i+1} = l_i + 1$.

Consequently, starting at most from $2.s + 1$, where s is the number of states of the machine, the sweeping to the right, in each rebound, begins with an aperiodic part on at most s steps and goes on further periodically with a period of length p. Let $I_1, \ldots, I_p$ the sequence of instructions involved by the period. We obtain, as l_i increases by 1 at each rebound, that the machine control happens to be successively in each exit state of the instructions I_j when it scans cell a at the end of each sweeping to the right. As the motion goes on endlessly, each of this state calls the left blue instruction when a 0 is scanned. Of course, this condition is also sufficient for the rebounds to go on endlessly.

Consequently, this situation of non halting is recursively enumerable. Let us call it, later on, fixed-point motion with left rebounds, an illustrative sample of which is given in table 1, figure h.

Case i = j, i $\neq$ k: $\boxed{\mathbf{i}0L0\mathbf{k}}$ and $\boxed{\mathbf{i}1L1\mathbf{i}}$

Notice that, here, the left blue instruction calls always a right one. Consequently, if there is an extremal right exit, which is necessarily performed by the left blue instruction, a right instruction is called and so, $l_i = 1$ for any lmp which, by the way, is also absolute since sink lemma applies to the situation of an extremal right exit.

Consequently, if an extremal right exit occurs, there are no more left exits. This rules out the case of infinitely many traversals.

For the case of infinitely many extremal right exits without left exit, what we have said about l_i's shows immediately that periodicity lemma applies, and so, we have here a degenerated right motion, see table 1, figure b.

Consider, now, the case of infinitely many extremal left exits without right exit. By what has just be seen, there cannot be a 0 on any absolute rmp at rmp time. And so there is always 1 in each absolute rmp at that time.

But here, we cannot apply directly $\mathbf{G}_s$-left motion lemma, since the left green instruction calls a right one if a 0 is scanned. In fact, we have an *extended* $\mathbf{G}_s$-left motion lemma in the following meaning:

If the machine goes from b at time t to a at time $t + T$, with $b - a \geq 2$, without going out of $[a, b]$ during time interval $[t, t + T]$ and with a 1 in b at time t, then $[a + 2, b]_{t+T} \in 1^*$.

Proof: As in the proof of $\mathbf{G}_s$-left motion lemma, consider the rightmost 0 at time $t + T$ in $[a, b]$, say at position c. By considering the last time t_1 in

$[t, t+T]$ at which the machine head scans cell $c+1$, we see that necessarily, $c \leq a+1$. If this were not the case, the left blue instruction would be called on c at time t_1+1 and, by sink lemma, the machine would scan again cell $c+1$ at a time later than t_1+1, which is impossible by the definition of t_1. ■

Consequently, when the machine goes to the left from the first absolute rmp, say p_1, it reaches the leftmost 1, say on c, at time t_1, then performs the left blue instruction on $c-1$ and performs a right one on $c-2$.

Consider, now, the first time t'_1 when the machine reaches cell c-3. There is there a 0. The machine head was in c-2 at time t'_1-1. If there is a 0 on cell c-2 at time t'_1-1, the instruction performed at that time is necessarily the left blue one. But this instruction is necessary called by the left green instruction because the head could not be on c-3 before t_1. And so, there is necessarily a 1 on c-1 at time t'_1-2 and $[c-1, p_1]_{t'_1-2} \in 1^*$.

If there is a 1 on cell c-2 at time t'_1-1, the instruction performed at that time is the left green one. As the machine head could not be on cell c-3 at time t'_1-2, it was in cell c-1 where, necessarily the left green instruction was performed at time t'_1-2 since the left blue one calls always a right instruction. And so, there is necessarily a 1 in cell c-1 at time t'_1-2 and $[c-2, p_1]_{t'_1-1} \in 1^*$.

Let $t_2 = t'_1$-1 in the latter case and $t_2 = t'_1$-2 in the former. We have that, at time t_2 the machine is in the same state as it was at time t_1, it has never gone to the right of p_1 between t_1 and t_2, there is a desert of 0's on the left of the scanned cell and there are 1's from it up to p_1. Consequently, the rightmost position lemma applies and we have a motion very similar to a simple left motion. However, call this motion, two-folded left motion, an illustrative sample of which is given below and in table 1, figure d.

Indeed, a somewhat more complicated motion as in simple left motion may here occur. It suffices that the machine should not immediately put a 1 in cell $c-1$ in the above situation. For instance, the following machine:

	0	1
1	$L\,2$	L
2	$1\,R\,1$	$R\,3$
3	$1\,R\,4$	$R\,1$
4		$R\,5$
5		$R\,1$

performs the motion of figure 8, below:

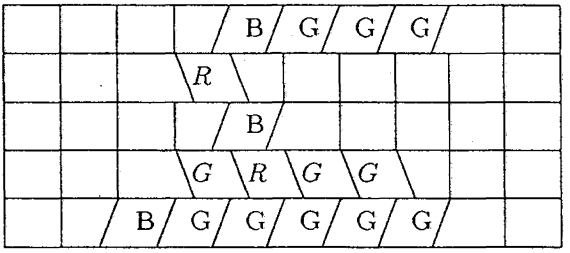


Figure 8. – Two-folded left motion: a sample

Case blue unstable and green unstable

The left instructions of the machine are now, respectively, $\boxed{\mathbf{i}0L0\mathbf{h}}$ and $\boxed{\mathbf{j}1L1\mathbf{k}}$ with $\mathbf{h} \neq \mathbf{i}$ and $\mathbf{k} \neq \mathbf{j}$. This case splits again into *four* subcases according to the relations between **i**, **j**, **h** and **k**.

If the *calling* relations between the left instructions are not too tight as it will be the case in the below three first cases, it will be seen that, roughly speaking, the motions are essentially motions to the right. In the last case, all motions illustrated on table 1 are possible, and it will be seen that any other motion is ruled out.

Case $\mathbf{h} \neq \mathbf{j}$, $\mathbf{k} \neq \mathbf{i}$:

In this case, each left instruction calls always a right one and so, sink lemma always applies to them. Consequently, more than one left exit is impossible, which rules out the case with infinitely many extremal left exits without right exit and, *a fortiori*, the case with infinitely many traversals.

The case with infinitely many extremal right exits without left exit remains as the single possible one. As, after a right extremal exit the left blue instruction is called, by sink lemma, as there is no sink, any *lmp* is at a distance equal to 1 from its defining exit. And so, periodicity lemma applies, which shows that the motion of the machine is the degenerated right one, see table 1, figure *b*.

Case $\mathbf{h} = \mathbf{j}$, $\mathbf{k} \neq \mathbf{i}$: $\boxed{\mathbf{i}0L0\mathbf{j}}$ and $\boxed{\mathbf{j}1L1\mathbf{k}}$

Here, the left blue instruction calls the left green one and sink lemma applies always to the left green instruction since it always calls a right one.

Consider an extremal right exit, if any, say on cell a. The left blue instruction is performed on a, then if there is a 1 on a-1, the left green

instruction is performed but, as we assume no sink, sink lemma shows that the machine heads goes to cell a, without having been on the left of a-2. If there is a 0 on a-1, a right instruction is called there and so, the machine head goes to cell a+1 without having been on the left of a-1. Thus we have here that $l_i \leq 2$ and so, periodicity lemma applies again and the motion of the machine is a simple right one, very near to a degenerated right motion.

This analysis shows that more than two left exits is ruled out and so, both cases involving infinitely many extremal left exits are ruled out.

Case h $\neq$ j, k = i: $\boxed{\mathbf{i}0L0\mathbf{h}}$ and $\boxed{\mathbf{j}1L1\mathbf{i}}$

This time, the left green instruction calls the left blue instruction which, on its turn, always calls right instructions.

This property of the left blue instruction entails, as in a previous case, that starting from cell a, as we assume no sink, the machine head goes eventually on cell a+1. Consequently, an extremal right exit rules out further left exits and in the case of infinitely many right exits, the motion is a degenerated right one.

By the way, infinitely many left exits are clearly ruled out too, in particular, infinitely many traversals.

Case h = j, k = i: $\boxed{\mathbf{i}0L0\mathbf{j}}$ and $\boxed{\mathbf{j}1L1\mathbf{i}}$

This case is the most difficult to deal with. Intuitively, it comes from the fact that this case is very near to the case in which a universal non-erasing Turing machine can be constructed: it is enough to add a left green instruction which calls the other left green one if a 1 is scanned, or to add a stable left green instruction with the same entry state as the left blue one. The latter solution allows the machine to cross over large zones of 1's separated by a single 0 and to stop this motion to the left by putting two consecutive 0's on the tape, as is shown, for instance, in [5].

In that case, all the motions previously encountered are possible plus a new one. In order to get some insight into the reason why, we shall characterize motions to the left.

Intuitively, the matching conditions on the allowed colours make a motion to the left possible only if the underlying configuration allows to call both left instructions, the one after another. We have, precisely:

B_u-G_u-LEFT MOTION LEMMA: *Assume that the machine cannot fall into a sink, that both left instructions of the machine are unstable and that each instruction calls the other if a symbol, different from its entry one, is scanned. Then, if the machine head is on cell b at time t, scanning a 0, if it reaches*

cell a at time $t+T$, $b-a>2$ and if $v(t+\tau)\in[a,b]$ for all $\tau\in[0,T]$, then $[a+1,b]_{t+T}\in(10)^*+0(10)^*$.

Proof: Notice first that the 0 which occurs in b at time t remains at this place until, at least, time $t+T$: the reason why is the fact that the first applied instruction is the left blue one. This instruction leaves the 0 for time $t+1$ and so, later on, at any time within $[t,t+T]$ at which the machine head scans b, a right half-turn occurs and so, the left blue instruction is always called. As the machine does not go on the right of b and as there is no sink, if there is no 1 on cell b-1 at time $t+1$, the head must write it at this time, as shown by the patterns of figure 6.

Let c the rightmost cell in $[a,b]$, if any, such that we have $[c+1,b]_{t+T}\in(10)^*+0(10)^*$ but that this property fails for $[c,b]_{t+T}$.

Suppose that we have 0 on $c+1$ at time $t+T$ and that we also have 0 in cell c at that time. Let t_1 the last time in $[t,t+T]$ at which the head scans cell $c+1$. It goes then from $c+1$ to c by the left blue instruction since the 0 which is there was always there before, by the non-erasing assumption. But on c, as a 0 is scanned, a right instruction is called and the head is again on $c+1$ later than t_1. This is impossible, unless $c=a$. So, if $c>a$, there is a 1 on c and so $[c,b]_{t+T}\in(10)^*$, which is contrary to the definition of c and so, $c=a$.

Suppose, now, that we have 1 on $c+1$ at time $t+T$ and that we also have 1 in cell c at this time. Let t_1 the last time in $[t,t+T]$ at which the head scans cell $c+1$. As the machine goes from $c+1$ to c, it must be performed by a left instruction, and as it leaves a 1 on cell $c+1$, it must be performed by the left green one. If there is a 1 on cell c at that time, then a right instruction is called and the machine is again on cell $c+1$ at a time later than t_1, unless $c=a$. If $c>a$, then there is a 0 on cell c at time t_1+1. As the machine never goes on the right of c during the time interval $[t_1+1,t+T]$, c is a position of right half-turn for the machine. Consequently, as the left blue instruction is called for the first step to the left at time t_1+1, the argument showing that 0 remains on cell b until $t+T$ also applies here for showing that we have 0 in c at time $t+T$. But this is a contradiction with the definition of c and so, $c=a$. ■

Consider, now, the main three cases of a non-halting situation. We shall follow as close as it will be possible the already discussed case when the left instructions are, respectively, blue instable and green stable, the blue one calling the green one if a 1 is scanned, *i.e.* when left instructions are $\boxed{\mathbf{i}0L0\mathbf{j}}$ and $\boxed{\mathbf{j}1L1\mathbf{j}}$ with $\mathbf{j}\neq\mathbf{i}$. Later on, this case will be called the *reference* case.

Call *regular* any segment of the tape of the form $(10)^+$ at current time.

Case of infinitely many traversals

In that case, we can repeat *mutatis mutandis* the argument which we used in the reference case.

Indeed, we consider again the first traversal from right to left. When the machine performs the extremal right exit with which the traversal begins, say at address b, the left blue instruction is performed. The motion to the left goes at least until the next left exit, when the first 0 of the left desert of 0's is scanned by a right instruction. By $\mathbf{B}_u$-$\mathbf{G}_u$-left motion lemma, this means that the motion to the left is stopped at this point by meeting two adjacent 0's, the rightmost one being scanned by the left blue instruction, say at address a+1. Notice that, as the motion goes on later on the left of a, this entails that the left blue instruction calls a right red one if a 0 is scanned.

Notice that left exits always occur in that way, by meeting 00 on the tape, and so, the sweeping to the right which follows the left exit starts always in the same state. By the way, it is then easy to see that if that sweeping ends on cell c, with $c \leq b$, the assumption of rightmost position lemma will be statisfied at the time of the next left exit and so, we obtain that the motion is a simple right one, which is contrary to the assumption of infinitely many traversals.

Consequently, the sweeping to the right ends on c with $c > b$. The discussion now follows as in the reference case since what is there important is that the state of the machine at left exits should be the same and that the regular zone of lenght l_i should not be altered: this is the case since a 1 written in a zero-place of the regular zone would prevent further motions on the left of this 1. And later, the periodic motion insures that the right instructions involved during the sweeping to the right do preserve regularness.

And so, we obtain that the motion is of type traversal and that this situation of non-halting is recursively enumerable.

Case of infinitely many extremal right exits without left exit

We cannot, here, follow as closely as previously the reference case. Indeed, in the present case, *two kinds* of *lmp*'s may occur. If we consider a motion to the left as described by $\mathbf{B}_u\mathbf{G}_u$-left motion lemma, and assume that it is performed by a single sweeping, it is stopped either by meeting 00 on the tape, or by meeting 11. According to the pattern, actually met by the head,

we shall speak, later on, of *lmp*'s of type 00 and of *lmp*'s of type 11. And so, on the contrary of what happens in the reference case, the machine has here, in principle, the possiblity to *alter* regular zones.

Notice first that *lmp*'s of type 11 are always absolute in view of the non-erasing assumption. For the same reason, notice that *lmp*'s of type 00 cannot be written in a regular zone of the tape but outside it, especially, in our case, on the right of it.

This latter property allows us to deal first with type 00 *lmp*'s:

Case of infinitely many absolute lmp's of type 00

For each absolute *lmp* of type 00, there is a first time when the machine head scans the first 0 of the *lmp* pattern while the head is moving on the right desert of 0's, after going on the right of a previous extremal right exit. It ensues that if p_i denotes again the sequence of abolute *lmp*'s, the relation $p_i + l_i \leq p_{i+1}$ holds. And so, $l_i + p_i - p_{i+1} \leq 0$ for all i. Consequently, by strong periodicity lemma, this motion is ultimately periodic and so, this non-halting situation also is recursively enumerable.

The corresponding motion may be a bit more complex than, strictly speaking, the simple right motion considered in former cases: there may be here several kinds of sweepings to the right. It looks like the parallel parts in the motions illustrated in figures *e*, *f* and *g* of table 1. Later call this motion right periodic motion.

Case of finitely many absolute lmp's of type 00

In that case, which is the alternative case to the previous one, we may assume that starting from a certain time t, there are only absolute *lmp*'s of type 11. Consequently, we assume that p_i is now the sequence of all *lmp*'s of type 11 after time t and l_i is defined as in periodicity lemma.

We have now two cases, according to whether $\liminf_{i\to\infty} l_i < +\infty$ or not. In the first case, we know from periodicity lemma that the motion of the machine is a right periodic one and so the non-halting is recursively enumerable.

In the second case, $\liminf_{i\to\infty} l_i = +\infty$ or, which is the same thing, $\lim_{i\to\infty} l_i = +\infty$. We shall see that, here, we have a motion, recalling the fixed-point motion with right rebounds seen in the reference case. But here, the proof is a bit more elaborated since, contrary to the reference case, the machine may alter the content of the tape during the motion to the right

which brings the head on the right of the preceding right extremal exit. And as will be seen later, this may be the case.

Remind that we may now assume that $l_i \geq I_R + 1$, where I_R is the number of right instructions of the machine. It is easy to see that $I_R = 2.(s-1)$, where s is the number of states of the machine.

Consider the first sweeping starting from *lmp* time on cell p_i. Its first instruction is always the same for all *lmp*'s of type 11. If this sweeping ends on at most I_R steps, and if it has not altered the tape, segment $[p_i, p_i + I_R]$ is then a trap zone for the machine, which is contrary to our assumption. So, either the sweeping has performed at least $I_R + 1$ steps and then a periodic motion has appeared (at most I_R possible right instructions), or the machine has altered the tape. In the latter case, at least a 1 has been written in a zero-place of the regular segment over which the head is running. The following sweeping to the left will stop on that rightmost 1 and the *same* sweeping to the right will then start. And so, in all cases we obtain a periodic motion, the period of which occupies a zone of length at most I_R.

Let us see that this second possibility is ruled out by our assumption on l_i's. Indeed, this periodic motion leads the machine on the right of the previous extremal right exit with position $a_i = p_i + l_i$. Starting from a_i, the head performs at most s steps to the right and then the head goes back to the left. Consequently, the next extremal right exit is at most at s steps of a_i. As in the period of the motion to the right the machine head writes at least one 1 on the tape, p_{i+1} is at most at I_R on the left of a_i. And so, $l_{i+1} \leq I_R + s$. But this holds for any *lmp* of type 11 starting from the time after which l_i are supposed to be big enough. This is contrary to the assumption $\lim_{i \to \infty} l_i = +\infty$.

Consequently, the motion to the right starting from *lmp* time in the case of an *lmp* of type 11 is a single sweeping to the right which brings the machine head on the right of the previous extremal right exit. We shall say again that we have then a *rebound* although in that case, the sweeping to the right may alter the regular zone that it crosses over.

In each rebound, the sweeping to the right starts in its aperiodic part, at most on I_R steps, and then enters the periodic part of lenght p with $p \leq I_R$, and from the previous analysis, the length of the aperiodic part added to p is at most I_R, so that there is at least one full period of this motion performed in all rebound that we shall consider now.

As any type 11 *lmp* is absolute, strong periodicity lemma allows to assume that $p_{i+1} < p_i + l_i$. Either no 1 is written in a zero-place of the regular part of

the zone crossed over during a sweeping to the right, or at leat one is written.

In the former case, we have the fixed-point motion with right rebounds described in the refercence case.

In the second case, notice that the rightmost 1 is written either during the starting aperiodic part of the rebound or during each period occuring during the considered sweeping. If a 1 is written during the periodic part, say of lenght p, we get then $p_{i+1} \geq p_i + l_i - p$. Consequently, strong periodicity lemma applies to this situation and so, the machine motion is a right periodic one.

And so, the considered 1 is written only during the starting aperiodic part of the sweeping.

As there are infinitely many extremal right exits, we shall find two absolute lmp's, say p_u and p_v, of course of type 11, with $l_u \equiv l_v \bmod p$. Let $a_u = p_u + l_u$, respectively $a_v = p_v + l_v$, and let t_u, t_v the respective times when the machine head scans cell a_u, a_v after lmp time. As, by the condition on $l_u, l_v \bmod p$, the state of the machine is the same at time t_u and t_v and as there is there a 0 and a desert of 0's on its right, the motion of the machine starting from time t_u is parallel to its motion starting from t_v, as long as the head remains on the right of a_u starting from t_u.

By parallelism of the motion after time t_u to the corresponding one after t_v as long as the head remains on the right of a_u, we have $l_{u+1} - l_u = l_{v+1} - l_v$ and so, $l_{u+1} \equiv l_{v+1} \bmod p$ since $l_{u+1} = l_u + a_{u+1} - a_u - d$, where $p_u + d$ is the position of the rightmost 11 pattern on the right of p_u at time t_u. Consequently, as $p_{u+1} < a_u$, the same argument applies to the further rebounds, until a_v is reached.

Consequently, we have

$$l_v - l_u = -d.(v - u) + \sum_{k=u}^{v-1} (a_{k+1} - a_k)$$

and, by the same way,

$$l_w - l_v = -d.(w - v) + \sum_{k=v}^{w-1} (a_{k+1} - a_k);$$

where a_w is the extremal right exit reached at a time t_w, with $a_w - a_v = a_v - a_u$, p_w is the corresponding lmp of type 11 and l_w defined with respect to p_w as usually.

As $v - u = w - v$, since the number of rebounds between t_u and t_v is the same as between t_v and t_w, and $a_{u+k+1} - a_{u+k} = a_{v+k+1} - a_{v+k}$ for all $k = 0, \ldots v - u - 1$ by parallelism, as shown previously, we obtain that $l_w - l_v = l_v - l_u$. It is clear now that, reasoning in the same way as in the reference case, those observations are enough for insuring that the motion described in this way will later go on by repeating peridocically the same number of rebounds followed, between two of them, by fixed sequences of motions from a certain position a back to this position, but without any step on the left of a in the meanwhile.

We shall here say that the motion is a motion with right rebounds without fixed-point as illustrated in table 1, figure g. Of course, what is here proved shows that the non-halting is also recursively enumerable in this case.

Notice that in the case we just studied, the case of infinitely many lmp's of type 00 is possible, but they are, of course, never absolute. Indeed, such lmp's may be created by the motion which happens between two consecutive rebounds, but the same motion 'erases' them during 'local' motions to the left, which implies, for instance, that the right instruction called by the left blue one if a 0 is scanned, should be a right red one.

Case of infinitely many extremal left exits without right exit

This case is very similar to the corresponding situation in the reference case. Indeed, as we consider here left exits, as already noticed in the case of infinitely many traversals, any left exit is an extremal one and the right instruction is called by the left blue one as a 0 is scanned. It is easy to notice that, in this case, sweepings to the right starting from the left exit do not alter the regular zone which lies between the left exit and the rmp points, ends excepted.

Consequently, it is easy to see that we have two cases: either a simple left motion if we ever find an absolute rmp which is a point of a regular zone, or fixed-point motion with left rebounds in the opposite case. Both kinds of motions are possible. In the case of a fixed-point motion with left rebounds, the motion is a sequence of consecutive rebounds without other motion in between. Moreover, there are two cases for such motions: one with all rmp's on the same position of type 00, the other with all rmp's on the same position of type 11.

As in all subcases of case blue unstable and green unstable the non-halting is recursively enumerable,the situation of non-halting is recursively enumerable too in the global case.

This completes the proof or theorem 1.

6. THE CASE OF POSSIBLE STATIONNARY INSTRUCTIONS

Notice, first, that the colour results used in the proof of theorem 1 have to be re-examined since works [9], [2] and [1] on which the proof is based assume that the considered machines have no stationnary instructions.

However, those results remain true in our new context. It is clear enough for [9]. For [2], it is enough to examine the proof of theorem A since in the new context eight colours may occur instead of five ones. Now, a close examination of the proof of that theorem shows that the same arguments hold even if stationnary instructions are allowed. Indeed, those instructions yield new colours which are taken into account by the colour criterion. Moreover, the motions on which is based the analysis produced in the proof of theorem A do not assume that stationnary instructions are ruled out. Consequently, cases when one of the two instructions with left laterality is stationnary is immediately delt with by theorem A since the remaining single left instruction yields at most one left pure motion colour. So we have to consider the case when the two instructions with left laterality are genuine left instructions blue and green, as above.

In the new context, definitions of section 3 remain unchanged. Lemmas of that section remain true since they are general lemmas on deterministic Turing machines which do not consider the laterality nor the move of the instructions.

The same conclusion is true for definition and lemmas of section 4, especially periodicity lemma and strong periodicity lemma which are also general results on deterministic Turing machines.

The first reduction of the problem goes as in section 5 since theorem A holds in the new context.

In fact all lemmas and arguments of section 5 remain valid. Only sink lemma has to be slightly reformulated: instead of 5 steps after time t, we have K steps, where K is a machine dependent constant depending on the number of stationnary instructions possibly called by the left ones or the right ones involved in the proof of sink lemma in section 5. The basic six cases are the same up to the fact that the trivial non-halting with a loop of stationnary instructions may occur. But this latter case is typically a *sink*. And so, as the lemma is used in section 5 under the assumption that there is no sink, the new lemma can be used in the same way in our new context.

The other lemmas about motions to the left have also to be slightly reformulated. When a lemma assumes that a right instruction is called, his

new version assumes that an instruction with right laterality is called. Notice, in particular, that proofs of left motion lemmas in section 5 are formulated in a way which allows possible stationnary instructions. With this slight modification, the whole argumentation of section 5 applies to the extended context.

7. COMPLETING THE SOLUTION OF THE LATERALITY PROBLEM FOR NON-ERASING MACHINES

In this section, we first introduce the extended definition of non-erasing machines on finite alphabets with, possibly, more than two letters. Then we state theorem 3 which is summarized in figure 2 given in the introduction. The second part of this section is devoted to the proof of theorem 3.

Extending the non-erasing assumption to any finite number of symbols

Notice that the non-erasing condition on $\{0,1\}$ induces an *order* relation on O and 1. This can easily be extended to any finite alphabet with at least two letters as done in the following definition:

DEFINITION: *Let Σ be a finite alphabet with at least two letters. Assume that a symbol of Σ is chosen as a blank symbol and that an order $\leq$ is defined on Σ such that the blank symbol is the least symbol. Say then that a Turing machine on Σ is non-erasing if and only if for any instruction $\mathbf{i}xMy\mathbf{j}$, the relation $x \leq y$ always holds.*

On alphabet $\{0,1\}$, this definition coincide with the previous one. It is now possible to state:

THEOREM 3: *The halting problem is decidable for any non-erasing Turing machine on a finite alphabet Σ satisfying the assumptions of the previous definition the program of which contains a single left instruction.*

This theorem solves the case of a single left instruction and any number of symbols for the alphabet. From figure 2 and theorems 1 and 2, a case remains to be examined: the case of non-erasing machines with two instructions having the left laterality on an alphabet with three letters, fitted with an order such that the chosen blank symbol is the minimum according to that order. It is then not difficult to see that the construction supplied in [4] of a universal non-erasing Turing machine on $\{0,1\}$ quoted in [6] can be extended to the case of three letters. Indeed, in the machine constructed in [4], there is a single motion to the left involving the three left instructions possessed by the machine program, and this motion is always stopped by meeting the same

00 pattern (two contiguous cells containing the blank symbol). If we have an extra-letter, it can be used to stop the motion to the left which, in that case, only needs two left instructions. If the extra-symbol is the greatest symbol on the three-letter alphabet, the construction of [4]can be a bit simplified: the extra-symbol can be used to prevent further motions to the left to again cross over already 'erased' letters, *i.e.* letters consisting only of 1's.

Consequently, the laterality problem in the non-erasing context whichever way the non-erasing condition is extended to finite alphabets with at least three letters is completely solved.

Extended non-erasing machines with a single left instruction

Let us now turn to the proof of theorem 3.

Let $I = \mathbf{i}xLy\mathbf{j}$ be the single left instruction of the considered non-erasing Turing machines, say, M. Let S be the number of instructions of M.

Assume first that $x \neq y$. Let instruction I be applied on cell a at time t. It may be assumed that at time t+1, the machine applies a right instruction, going back to a which now contains y. On $y \neq x$, a right instruction is eventually called, unless a loop of stationnary instructions appears since, as $x < y$, by the non-erasing assumption, letter x cannot be scanned again in cell a. By intermediate cell lemma, the machine can no more go to the left of cell a+1. And so, if x is not the blank symbol, starting from a certain time, the machine endlessly goes to the right. If x is the blank symbol, then there is at most a single left exit which is then an extremal one and in the case of infinitely many extremal right exits, the just supplied analysis shows that $\ell_i \leq S$ and so, a recursively enumerable non-halting does occur.

Assume now that $x = y$. If instruction I is unstable, *i.e.* $\mathbf{i} \neq \mathbf{j}$, the same argument as in sink lemma extended to stationnary instructions applies. And so, infinite in space motions of the machine goes endlessly to the right with possible extremal right exits for which $l_i \leq 1$ which entails, by periodicity lemma, a recursively enumerable non-halting motion.

Assume now $x = y$ and $\mathbf{i} = \mathbf{j}$.

Assume that there are infinitely many extremal right exits. Symbol x is then the blank symbol. As there are infinitely many right exits and as left exits would trivially be endlessly repeated by stability of instruction I, the case of infinitely many traversals is ruled out. If s_i is the position of the ith extremal right exit, it is plain that $s_{i+1} < s_i + S$. By stability of instruction I, any *lmp* contains a non-blank symbol at *lmp* time and so, all *lmp*'s are absolute. As x is the blank symbol, at *lmp* time, there is a

desert of 0's and so, two such lmp's scanned at lmp-time under the same state will occur. Leftmost position lemma applies, and so this non-halting situation is recursively enumerable.

Assume now that there are infinitely many extremal left exits. As instruction I is stable, necessarily symbol x is not blank. Notice that this fact rules out any extremal right exit. Observe that in each rmp, symbol x must be present at rmp time, unless the machine cannot go back to the left. As there are infinitely many extremal *left* exits, the motion from an rmp to the next extremal left exit leaves a zone of x's only. Sweepings to the right which happen after the extremal left exits do not change x into another symbol and so, this situation is clearly the same as the simple left motion studied in section 5. Consequently, this case is also a recursively enumerable non-halting motion.

As all cases have been examined, theorem 3 is proved. ■

CONCLUSION

We think that the material given here convinced the reader that the approach to the study of Turing machines based on a qualitative analysis of the machine head, is a promising one. It can be noticed, in particular, that the notion of colour plays an important rôle in the proof that the laterality number is a decidability criterion for the halting problem of non-erasing Turing machines.

Other criteria should be explored in order to better know the frontier between decidability and undecidability. This is a still wide open field, and we think that we here introduced a tool of possible use in that research.

ACKNOWLEDGEMENTS

The author thanks Serge Grigorieff for his constant attention and his endless encouragings. He thanks too gratefully Hubert Comon, Étienne Grandjean and Josef Gruska for their attention to this work.

REFERENCES

1. M. Margenstern, Sur la frontière entre machines de Turing à arrêt décidable et machines de Turing universelles. Rapport de recherche LITP N° 92.83, (Institut Blaise Pascal), 1992.

2. M. MARGENSTERN, Turing machines: on the frontier between a decidable halting problem and universality. Lecture Notes in Computer Science, *710*, pp. 375-385, in *Proceedings of FCT'93*, 1993.
3. M. MARGENSTERN, Décidabilité du problème de l'arrêt pour les machines de Turing non-effaçante sur $\{0,1\}$ et à deux instructions gauches. Rapport de recherche LITP N° 94.03 (Institut Blaise Pascal), 1994.
4. M. MARGENSTERN, Une machine de Turing universelle sur $\{0,1\}$, non-effaçante et à trois instructions gauches. Rapport de recherche LITP N° 94.08 (Institut Blaise Pascal), 1994.
5. M. MARGENSTERN, Une machine de Turing universelle non-effaçante à exactement trois instructions gauches. *C.R.A.S., Paris*, 1995, *320*, I, pp. 101-106.
6. M. MARGENSTERN, Non-erasing Turing machines: a new frontier between a decidable halting problem and universality. Lecture Notes in Computer Science, 1995, *911*, pp. 386-397, in *Proceedings of LATIN'95*).
7. M. MARGENSTERN and L.PAVLOTSKAYA, Deux machines de Turing universelles à au plus deux instructions gauches. *C.R.A.S., Paris*, 1995, *320*, I, pp. 1395-1400.
8. M.L.MINSKY, Computation: Finite and Infinite Machines. Prentice Hall, Englewood Cliffs, N.J., 1967.
9. L.PAVLOTSKAYA, Razreshimost' problemy ostanovki dlja nekotorykh klassov mashin T'juringa. Matematicheskie Zametki, Ijun' 1973, *13*, (6), pp. 899-909, pp. 899-909. (transl. Solvability of the halting problem for certain classes of Turing machines, *Notes of the Acad. Sci. USSR*, Nov.1973, 13, (6), pp. 537-541.
10. L.PAVLOTSKAYA, Dostatochnye uslovija razreshimosti problemy ostanovki dlja mashin T'juring, Avtomaty i mashiny, 1978, pp. 91-118 (Sufficient conditions for halting problem decidability of Turing machines) (in Russian).
11. Ju.V.ROGOZHIN, universal'nykh mashin T'juringa. Matematicheskie Issledovanija, 1982, *69*, pp. 76-90, (Seven universal Turing machines) (in Russian).
12. Ju.V.ROGOZHIN, Universal'naja mashina T'juringa s 10 sostojanijami i 3 simvolami. Matematicheskie Issledovanija, 1992, *69*, pp. 76-90, (A universal Turing machine with 10 states and 3 symbols) (in Russian).
13. Ju.V.ROGOZHIN, About Shannon's problem for Turing machines. Comput. Sci. J. Moldova, 1993 *1* 3(3), pp. 108-111.
14. Ju.V.ROGOZHIN, Small universal Turing machines, Theoretical Computer Science, *168-2*, special issue on Universal Machines and Computations, 1996, pp. 215-240.
15. Ju.V.ROGOZHIN, A universal Turing machine with 21 states and 2 symbols, private communication, 1996.
16. C.E.SHANNON, A universal Turing machine with two internal states., *Ann. of Math. Studies*, 1956, *34*, pp. 157-165.
17. A.M.TURING, On computable real numbers, with an application to the Entscheidungsproblem., *Proc. Lond. Math. Soc.*, 1936, ser. 2, *42*, pp. 230-265.
18. H. A. WANG, variant to Turing's theory of computing machines, *J. Assoc. Comput. Mach.*, 1957, *4*, 1, pp. 63-92.
19. G.P.ZYKIN, Zamechanie ob odnoj teoreme Khao Vana, *Algebra i Logika*, 1963, *2*, 1, pp. 33-35, (Remark on a theorem of Hao Wang) (in Russian).