Undecidability of the Surjectivity of the Subshift Associated to a Turing Machine

Rodrigo Torres, Nicolas Ollinger, Anahi Gajardo

Undecidability of the surjectivity of the subshift associated to a Turing machine

Rodrigo Torres[1], Nicolas Ollinger[2], and Anahí Gajardo[1⋆]

[1] Departamento de Ingeniería Matemática, Centro de Investigación en Ingeniería Matemática, Centro de Modelamiento Matemático, Universidad de Concepción, Casilla 160-C, Concepción, Chile
rtorres, anahi@ing-mat.udec.cl
[2] LIFO, Université d'Orléans
BP 6759, F-45067 Orléans Cedex 2, France
Nicolas.Ollinger@univ-orleans.fr

Abstract. We consider Turing machines (TM) from a dynamical system point of view, and in this context, we associate a subshift by taking the sequence of symbols and states that the head has at each instant. Taking a subshift that select only a part of the state of a system is a classical technic in dynamical systems that plays a central role in their analysis. Surjectivity of Turing machines is equivalent to their reversibility and it can be simply identified from the machine rule. Nevertheless, the associated subshift can be surjective even if the machine is not, and the property results to be undecidable in the symbolic system.

Key words: Turing machines, discrete-time dynamical systems, subshifts, formal languages.

Relations between dynamics and computation has been looked for in several works [1–4]. In a first approach, these two concepts are very different things, roughly speaking, one can say that computation consists in obtaining an output starting from an input by means of a dynamics. The dynamics itself is not relevant, several dynamics can produce the same result. On the other hand, a complex computation cannot be obtained through a too simple dynamics. Some –weak– relations exists.

A direct way to tackle this topic consists in looking at Turing machines with the tools of dynamical systems theory. A first paper by Kürka has taken this viewpoint [4] and several others have followed [5, 6, 1, 3, 2]. There, notions such as equicontinuity, entropy and periodicity have been studied, and putted in relation with more natural properties of the machines. Some of these properties were proved to be undecidable, as is the case of *periodicity* of Turing machines in [3].

Here we continue in the line of [1, 2] that focus on a particular symbolic system (a subshift) associated with the Turing machine, that is called *t-shift*. It

⋆ This work has been supported by CONICYT FONDECYT #1090568 and BASAL project CMM, Universidad de Chile, and CI^{2}MA, Universidad de Concepción.

consists in taking the linear sequence of states and symbols that the machine reads during its evolution over a given initial configuration, and to consider afterwards the set of infinite sequences produced by all the possible initial configurations. Subshifts are key tools in the study of general dynamical systems, they give crucial information about the system (see for example [7]). In this approach, the complexity of the subshift has been related with the complexity of the machine.

In this paper, we study the *surjectivity* of the t-shift. A function T is surjective if for every y, there exists an x such that $T(x) = y$. If T is the function that defines the evolution of a Turing machine, it results to be equivalent to the reversibility of the machine and it can be characterized in a very simple way from the machine's transition rule. If the machine is surjective, so it is its associated t-shift, but the converse is not true. Thus we look for a characterization of the surjectivity of the t-shift in terms of some property of the machine.

When a subshift is surjective, every sequence can be extended by the left, in such a way that the subshift itself can be considered as a set of bi-infinite sequences, *i.e.*, sequences running over $\mathbb{Z}$. In this case, the shift action is reversible, and other properties can be considered.

Another reason to study this property is that surjectivity is a necessary condition for transitivity, which is a relevant property in the area of dynamical systems.

The following section provides definitions and concepts about symbolic dynamics and Turing machines. Section 2 gives a characterization of the t-shift surjectivity. In section 3, we establish the undecidability of some preliminary problems, to conclude with the undecidability of the property in the last section.

1 Definitions

1.1 Turing Machine

Turing machine written in quadruples. Following Morita [8], a Turing machine (TM) M is a tuple (Q, Σ, δ), where Q is a finite set of states, Σ is a finite set of symbols and $\delta \subseteq Q \times \Sigma \times \Sigma \times Q \cup Q \times \{/\} \times \{-1, 0, +1\} \times Q$ is the writing/moving relation of the machine. The machine works on a tape, usually bi-infinite, full of symbols from Σ. A *configuration* is an element (w, i, q) of $\Sigma^{\mathbb{Z}} \times \mathbb{Z} \times Q$. A writing instruction is a quadruple (q, s, s', q'); it can be applied to a configuration (w, i, q'') if $w_i = s$ and $q = q''$, leading to the configuration (w', i, q'), where $w'_i = s'$ and $w'_k = w_k$ for all $k \neq i$. A moving instruction is a quadruple $(q, /, d, q')$; it can be applied to a configuration (w, i, q'') if $q = q''$, leading to the configuration $(w, i + d, q')$.

Turing machine written in quintuples. A Turing machine M can also be written in quintuples by having the writing/moving relation δ considered as $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{-1, 0, +1\}$. A quintuple instruction (q, s, q', s', d) can be applied

to a configuration (w, i, q'') if $w_i = s$ and $q = q''$, leading to the configuration $(w', i + d, q')$, where $w'_i = s'$ and $w'_k = w_k$ for all $k \neq i$.

Turing machines, when viewed as computing model, have a particular starting state q_0, and a particular symbol called blank symbol; the computation is intended to start over a configuration $(w, 0, q_0)$, where w represents the input, a word with a finite number of non-blank symbols. The computation process stops when the machine reaches another particular state: the halting state q_F. In this paper, we are omitting these three parameters, since we do not want the machine to halt and we will study its dynamics for arbitrary initial configurations. In any case, the halting problem can be translated to the present context as the problem of deciding whether the machine reaches a particular state when starting in another particular state with an homogeneous configuration except for a finite number of cells.

We also remark that the quintuples model is the traditional one, while the quadruples model is used for reversible Turing machines. One can translate any machine written in quadruples into a machine written in quintuples in a simple way because *writing instructions* are just quintuples instructions that do not cause any movement, and *moving instructions* are those that do not modify the tape. The converse transformation is also possible but a quintuple instruction will need to be replaced by a writing instruction followed by a moving instruction, thus the set of states needs to be duplicated and the time is also multiplied by two. Therefore, both models are equivalent as computing system, but not as dynamical system.

Deterministic Turing machine. A Turing machine M is *deterministic* if, for any configuration $(w, i, q) \in X$, at most one instruction can be applied (regardless the machine is written in quadruples or quintuples). In terms of quintuples, this is equivalent to give δ as a (possibly partial) function $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, 0, +1\}$. This function δ can be projected into three components $\delta_Q : Q \times \Sigma \to Q$, $\delta_S : Q \times \Sigma \to \Sigma$ and $\delta_D : Q \times \Sigma \to \{-1, 0, +1\}$.

Complete Turing machine. In any of the two models, if no instruction can be applied, the machine halts. A Turing machine M is *complete* if for each configuration (w, i, q), at least one instruction can be applied, *i.e.*, it never halts.

Analogous notions can be defined when going backward in time.

Backward deterministic Turing machine. A Turing machine M is *backward deterministic* if each configuration comes from at most one previous configuration.

A Turing machine written in quadruples is backward deterministic (as seen in [8]) if and only if for any two different quadruples (q, s, s', q') and (q'', s'', s''', q') in δ, it holds:

$$s \neq / \wedge s'' \neq / \wedge s' \neq s'''$$

Backward complete Turing machine. A Turing machine M is *backward complete* if each configuration comes from at least one preimage.

Reversible Turing machine. A Turing machine is reversible if it is deterministic forward and backward. For a machine written in quadruples, reversing the quadruples gives the reverse machine. The reverse instruction of a writing instruction (q, s, s', q') is (q', s', s, q). The reverse instruction of a movement instruction $(q, /, d, q')$ is $(q', /, -d, q)$. It is not difficult to see that a reversible Turing machine is complete if and only if its reverse is complete.

All of these last properties are local, and they can be checked in a finite number of steps.

1.2 Dynamical System

A *dynamical system* is a pair (X, T), where X is called phase space and $T : X \to X$ is called *global transition function*. In this paper, we consider $X = \Sigma^{\mathbb{Z}} \times \mathbb{Z} \times Q$. $\Sigma^{\mathbb{Z}}$ is called the *two-sided full shift* and its elements are called *bi-infinite words*, the symbol $'$ will be used to mark the position 0; for example, $\ldots 2333'233124 \ldots$ indicates that 2 is set in the position 0. $\Sigma^{\mathbb{N}}$ is the *one-sided full shif*t and its elements are called *infinite words*.

Subshifts. The *shift* function σ, is defined both in $\Sigma^{\mathbb{Z}}$ and $\Sigma^{\mathbb{N}}$ either by $\sigma(\ldots w_{-2}w'_{-1}w_0w_1w_2\ldots) = \ldots w_{-1}w'_0w_1w_2w_3\ldots$ or $\sigma(w_1w_2w_3\ldots) = w_2w_3\ldots$; it is a bijective function in the first case. Σ^* denotes the set of finite sequences of elements of Σ, called *finite words*. Two words $z = z_0...z_n$ and $y = y_0...y_m$ can be *concatenated* by just putting them one after the other: $zy = z_0...z_ny_0...y_m$. A word x can also be concatenated with a semi-infinite word $w = w_0w_1w_2...$: $xw = x_0...x_nw_0w_1\ldots$ A finite word z is said to be a *subword* of another (finite or infinite) word v, if there exists two indices i and j, such that $z = v_iv_{i+1}...v_j$. In this case we write: $z \sqsubseteq v$. Subsets of Σ^* are called *formal languages*. Given a subset of the full shift S, a formal language is defined:

$$\mathcal{L}(S) = \{z \in \Sigma^* \mid (\exists w \in S)\ z \sqsubseteq w\}\ .$$

Reciprocally, given a formal language L, a set of infinite sequences can be defined:

$$\mathcal{S}_L = \{w \in \Sigma^{\mathbb{M}} \mid (\forall z \sqsubseteq w)\ z \in L\}\ .$$

When S satisfies $\mathcal{S}_{\mathcal{L}(S)} = S$, it is called a *subshift*.

The t-shift. A complete and deterministic Turing machine $M = (Q, \Sigma, \delta)$ can be associated with a dynamical system (X, T), where X is the set of configurations $\Sigma^{\mathbb{Z}} \times \mathbb{Z} \times Q$, and the global transition function $T : X \to X$ consists into apply one transition of the Turing machine. We define $\pi : X \to Q \times \Sigma$ by $\pi(w, i, q) = (q, w_i)$. The *t-shift* associated to T, denoted by $S_T \subseteq (Q \times \Sigma)^{\mathbb{N}}$, is the set of orbits $\tau(x) = (\pi(T^n(x)))_{n\in\mathbb{N}}$, for $x \in X$. It is not difficult to see that S_T is in fact a subshift [1].

2 Surjectivity

As we have said, when M is deterministic and complete, T is a function. In this context, backward determinism is equivalent to injectivity of T and backward completeness corresponds to surjectivity. Through a cardinality argument, it is possible to show that, when the machine is deterministic and complete, surjectivity is equivalent to injectivity and both are easy to check from the machine's transition rule. From now on, we will work only with deterministic and complete Turing machines.

Remark 1. A Turing machine $M = (Q, \Sigma, \delta)$ written in quintuples is surjective if and only if, for every $q' \in Q$ and s', r', $t' \in \Sigma$, there is at least one $q \in Q$ and $s \in \Sigma$ such that $\delta(q, s) = (q', s', +1)$ or $\delta(q, s) = (q', r', -1)$ or $\delta(q, s) = (q', t', 0)$.

If for some q', the condition of TM surjectivity is not satisfied, we say that q' is *defective*. Thus a machine is surjective if and only if it has no defective state.

Definition 1. *A state $q' \in Q$ of a Turing machine $M = (Q, \Sigma, \delta)$ is said to be defective if:*

1. *(Quintuple model) There exist symbols s', r', $t' \in \Sigma$ such that no instruction gives: $(q', s', +1)$, $(q', r', -1)$ or $(q', t', 0)$.*
2. *(Quadruple model) There exist $s' \in \Sigma$ such that there exist no instruction $(q, /, d, q')$ nor (q, s, s', q'), for no $q \in Q$, $s \in \Sigma$, and $d \in \{-1, 0, +1\}$.*

Notice that an unreachable state is indeed defective, but we will assume that every state is reachable. If not, the subshift S_T will not be surjective in any case.

Surjectivity of T is inherited by the subshift S_T, however, if T is not surjective, the subshift can still be surjective. For example, let M be the Turing machine that simply moves to the right by always writing a 0. This machine is not surjective, but the associated subshift does.

Remark 2. A t-shift is surjective if and only if: $(\forall u \in S_T)(\exists a \in Q \times \Sigma)\ au \in S_T$

If $u = \left(\begin{smallmatrix} q_1 & q_2 & \cdots \\ s_1 & s_2 & \cdots \end{smallmatrix}\right) \in S_T$ and $a = (q, s)$, condition $au \in S_T$ says that $\delta_Q(q, s) = q_1$ and that the configuration that produces u has the symbol s at position $-\delta_D(q, s)$. If the machine does not visit position $-\delta_D(q, s)$, s can be any symbol, otherwise it is restricted to the constraint $\delta_S(q, s) = s_{i+1}$, where $i = \min\{j \mid \sum_{k=1}^{j} \delta_D(u_k) = -\delta_D(q, s)\}$.

In the example, the unique state of the machine is defective, it does not admit the symbol '1' at the left of the head, but position -1 is never revisited, that is why any symbol can be appended at the beginnig of u. If the state q_1 is defective and it does not admits the symbol s_{i+1} at position $-\delta(q, s)$, then $au \notin S_T$. Surjectivity will be possible when defective states avoid the head from going in to the "conflictive" positions, we develop this in the next section.

It is important to note that in the quadruples model, the surjectivity of T is held by the subshift S_T and vice versa. If we have a defective state q_1, then there exist no moving instruction leading to q_1. From the previous assertion, if q_1 is defective, then S_T is not surjective. We are interested in surjectivity only within the quintuples model.

2.1 Blocking States

We say that a state q is a *blocking state to the left (right)* if:

$$(\forall u \in S_T)(\forall s \in \Sigma)\ u_1 = (s,q) \Rightarrow \left[(\forall j \in \mathbb{N})\ \sum_{k=1}^{j} \delta_D(u_k) \neq -1(+1)\right] .$$

We also say that q is an *s-blocking state to the left (right)* for a given $s \in \Sigma$, if:

$$(\forall u \in S_T)\ u_1 = (s,q) \Rightarrow \left[(\forall j \in \mathbb{N})\ \sum_{k=1}^{j} \delta_D(u_k) \neq -1(+1)\right] .$$

Finally we say that q is just a *blocking state*, if for every $s \in \Sigma$, q is an s-blocking state either to the left or right.

A state q is said to be *reachable from the left (right)* if there exists a state q' and symbols s, s' such that $\delta(s', q') = (s, q, +1)$ (resp. -1).

The surjectivity on S_T can be characterized through these notions, in fact, S_T is surjective if and only if for each $q' \in Q$ at least one of the following holds:

1. q' is not defective: If q' is not defective, then, independently on the context, it can be reached from some configuration.
2. q' is blocking to the left (right) and it is reachable from the left (right): If q' happens to be a blocking state to the left (right), no configuration producing $u \in S_T$, with $u_1 = (q', s_1)$, is able to revisit the position -1 $(+1)$ (with respect to the initial head position), so any $(q, s) \in Q \times \Sigma$, such that $\delta_Q(q, s) = q'$ and $\delta_D(q, s) = +1(-1)$, can be appended at the beginning of u.
3. q' is blocking and it is reachable from the left and from the right: If q' happens to be a blocking state, $u \in S_T$ starts with $u_1 = (q', s_1)$ and q' is s_1-blocking to the left (right), then no configuration producing u is able to revisit the position -1 $(+1)$ (with respect to the initial head position), so any $(q, s) \in Q \times \Sigma$, satisfying $\delta_Q(q, s) = q'$ and $\delta_D(q, s) = +1(-1)$ can be appended at the beginning of u.

If we could decide when a state is blocking, we could decide surjectivity. In the next section, however, we see that checking the blocking property is not possible.

3 Undecidability of preliminary problems

In this section, we show the undecidability of several problems related with the blocking property of a state, that will serve as intermediate to finally prove the undecidability of the surjectivity on S_T in section 4.

Let us remark that the following proofs are equivalent in quadruples and quintuples model, one only has to use the usual transformation described in the section 1. The last is possible because the following problems are related to movement abilities of the head.

3.1 Undecidability of the blocking state problem

Let us consider the next three problems.

(BSl) Given a Turing machine M and a state q, decide whether q is a blocking state to the left.
(BSr) Given a Turing machine M and a state q, decide whether q is a blocking state to the right.
(BS) Given a Turing machine M and a state q, decide whether q is a blocking state.

Let us remark that (BSl) and (BSr) are Turing equivalent, *i.e.*, (BSl) reduces to (BSr) and vice versa. To see this it is enough to see that switching the movement direction on every instruction of a given machine M produces a machine M' whose states are blocking to the left if and only if the respective states of M are blocking to the right.

We prove the undecidability of these three problems by reduction from the emptiness problem, which is known to be undecidable for machines written either in quadruples or quintulples and also for machines restricted to work on a semi infinite tape. The definition of the emptiness problem is adapted to the present context as follows.

(E) Given a Turing machine M, and two states q_0 and q_F, decide whether there is an input configuration $(w, 0, q_0)$ that makes the machine to reach the state q_F in finite time.

Lemma 1. *(BSl) is undecidable.*

Proof. We prove undecidability by reduction from the emptiness problem. Let $M = (Q, \Sigma, \delta)$ be a Turing machine, and let $q_0, q_F \in Q$ be two states. We will assume, without loss of generality, that M is written in quintuples, and that starting with q_0 the head never goes to the left of position 0 (this is equivalent to say that the machine works only on the right side of the tape). Let us define M' just like M but with an additional state q_{aux}, and some small differences in its transition function δ:

$$\delta(q_F, s) = (q_{aux}, s, -1), \text{ and}$$
$$\delta(q_{aux}, s) = (q_{aux}, s, -1), \text{ for every } s \in \Sigma\ .$$

Thus, M reaches q_F for some input $(w, 0, q_0)$ if and only if the state q_0 is not a blocking state to the left for M'. ■

Theorem 1. *(BS) is undecidable.*

Proof. Let $M = (Q, \Sigma, \delta)$ be a Turing machine, and let $q_0, q_F \in Q$ be two states. Let M' be a machine defined as in the last proof. Since M works only on the right side, we have that $\delta_D(q_0, s) = +1$ for every s, thus q_0 is not s-blocking to the right for any symbol s. Therefore, q_0 is blocking to the left for M' if and only if q_0 is a blocking state for M'.

It results that the emptiness problem is satisfied for (M, q_0, q_F) if and only if the blocking problem is satisfied for (M', q_0). ■

3.2 Undecidability of the blocking state problem in complete RTMs

With the results of the last section, we discard the possibility of solving the problem of surjectivity via blocking states. However, knowing about the surjectivity of S_T for a given machine M does not help to solve the blocking problem for a particular state q of M. Thus surjectivity can still be decidable. We want to reduce the blocking state problem to the surjectivity problem, but to do so we need to produce a machine whose surjectivity depends only on the blocking property of one of its states. This can be achieved by working with reversible machines, which are modified to make one of its states defective. That is why we introduce a new problem.

(BSLrtm) Given a complete and reversible Turing machine M and a state q that is reachable from the left, decide whether q is a blocking state to the left.

We prove the undecidability of this problem by reduction from the halting problem of reversible two counter machines, which is proved undecidable in [9].

Definition 2. *A k-counter machine (k-CM) is a triple (S, k, R), where S is a finite set, $k \in \mathbb{N}$ is the number of counters, and $R \subseteq S \times \{0, +\}^k \times \{1, .., k\} \times \{-1, 0, +1\} \times S$ is the transition relation. A configuration of the machine is a pair (s, ν), where s is the current state and $\nu \in \mathbb{N}^k$ is the content of the k counters. By considering the function* sign: $\mathbb{N}^k \to \{0, +\}^k$ *defined by* $\text{sign}(\nu)_j = 0$ *if* $\nu_j = 0$ *and* $+$ *otherwise, an instruction $(s, u, i, d, t) \in R$ can be applied to a configuration (s, ν) if* $\text{sign}(\nu) = u$*, and the new configuration is (t, ν') where $\nu'_j = \nu_j$ for every $j \neq i$ and $\nu'_i = \nu_i + d$. R cannot contain the instruction $(s, u, i, -1, t)$ if $u_i = 0$.*

Just like Turing machines, a k-counter machine is said to be *deterministic (k-DCM)* if at most one instruction can be applied to each configuration. In addition, a k-CM C is said to be *reversible (k-RCM)* if it is forward and backward deterministic.

The halting problem consists in determining, given an initial configuration (s, ν), whether the machine reaches a given halting state t. It is undecidable for $k = 2$, even if the initial configuration is fixed to $(s_0, (0, 0))$.

Theorem 2. *(BSLrtm) is undecidable.*

Proof. We prove undecidability by reduction from the halting problem of 2-RCM.

Let $C = (S, 2, R)$ be a 2-RCM, with initial configuration $(s_0, (0, 0))$ and final state $t \in S$. For this proof we need a Turing machine M and a state $q = q_0$ meeting the following:

1. it simulates C on the right side of the tape,
2. it is reversible,
3. it reaches the position -1 starting at 0 from q_0 if and only if C halts (reaches t) starting from $(s_0, (0, 0))$,

4. it is complete, and
5. q_0 is reachable from the left.

If the machine meets the above objectives, q_0 will not be blocking to the left if and only if, starting from $(s_0, (0, 0))$, C halts.

The first 3 objectives can be viewed in detail in the appendix; however, here we sketch them briefly. For simplicity, we define M in the quadruple form. For the first objective, we make a traditional simulation. Let $M = (Q, \Sigma, \delta)$ be a Turing machine. Starting with q_0, the machine writes "$< \mid >$" into the tape, and goes to state s_0 (as seen in appendix, Part 1). This corresponds to the initial configuration $(s_0, (0, 0))$ of the counter machine. The simulation will correctly work if the tape initially contains only 1s. In general, each configuration $(s, (n, m))$ of the counter machine will be represented in the Turing machine by the configuration $(...' < 1^n|1^m > ..., 0, s)$. This will be achieved by simulating each instructions of C, as seen in the appendix, Part 2. In this way, M simulates C. It is noteworthy that, throughout the simulation, the machine does not reach the left side of the tape.

Providing simple safeguards, each counter machine instruction can be simulated by M in a reversible way. However, when reaching a state $s \in S$, while the counter machine knows whether each of its register is empty or not, the Turing machine does not, thus it may not be reversible in such situations. In order to solve this, the symbols of the form (d, d') are added to Σ, for each $d, d' \in \{0, +\}$; and we better represent C configurations by $(...'(d, d')1^n|1^m > ..., 0, s)$.

Now, for the third goal, when M reaches the halting state t of C, we add an extra transition to move to the left: $(t, /, -1, t_{aux})$. Thus M is able to reach the -1 starting from q_0 if and only if C halts when it starts from $(s, (0, 0))$.

Next, for the fourth objective, we use an idea from [3]. Create a new machine $M' = (Q', \Sigma, \delta')$, with $Q' = Q \cup \{+, -\}$. States of the form $(q, +)$ act in the same way that in the M machine, and states of the form $(q, -)$ makes the reverse transitions. Each transition not defined for M, switch $+$ by $-$; analogously, transitions not defined in the reverse makes $-$ to become $+$. This makes M' complete.

The fifth goal is attained by modifying M' in only one instruction:
$((q_0, -), /, 0, (q_0, +))$ is switched by $((q_0, -), /, +1, (q_0, +))$.

Thus, q_0 is a blocking state to the left, reachable from the left, for the complete reversible TM M', if and only if C does not halt (reaches the t state) from $(s_0, (0, 0))$. ■

Remark 3. The same machine can be used to proof that the emptiness problem for reversible and complete TM is undecidable, We simply make $q = q_0$ and $q' = t$.:

(E_{RCTM}) Given a reversible and complete TM $M = (Q, \Sigma, \delta)$ and two states $q, q' \in Q$, decide if there exists a configuration $(w, 0, q)$ such that the state q' is attained in finite time.

4 Undecidability of the surjectivity of the subshift associated to a Turing machine

(Surj) Given a deterministic and complete Turing machine written in quintuples, decide whether its t-shift is surjective.

Theorem 3. *(Surj) is undecidable.*

Proof. We prove the undecidability by reduction from (BSLrtm). Let $M = (Q, \Sigma, \delta)$ be a complete and reversible Turing machine. Let q' be a state that is reachable from the left, and let q, s and s' be such that $\delta(q, s) = (q', s', +1)$. We know that this machine is surjective. We assume that δ is written as a function.

Now let us define M' as M, but with an additional state q_{aux} and the following new instructions:

$$(\forall t \in Q)\ \delta(q_{aux}, t) = (q', t, 0)\ .$$

and changing $\delta(q, s) = (q', s', +1)$ by $\delta(q, s) = (q_{aux}, s', +1)$.

M' is not surjective, because the configuration (w, i, q_{aux}) has not preimage if $w_{i-1} \neq s'$. So q_{aux} is the unique defective state of M'. In this way, if q' is a blocking state to the left for M, then so is q_{aux} and since it is also reachable from the left, the t-shift of M' is surjective. On the other hand, if this t-shift is surjective, q_{aux} must be blocking. As q_{aux} is only reachable from the left, it must be blocking to the left. This is possible only if q' is blocking to the left. ■

5 Conclusions

Surjectivity of the t-shift of a Turing machine resulted to be not equivalent to the surjectivity of the machine it self. The last property was characterized in terms of another property of the Turing machine, the blocking property of its states: in the absence of the surjectivity of the Turing machine, some of its states must be blocking.

But the blocking property resulted to be undecidable as well as the surjectivity of the t-shift. A rather simple problem in Turing machines can not be decided in t-shift, so we think that others more complicated problems (for example, transitivity) can not be decided too, but this will require a more deep investigation.

References

1. Gajardo, A., Mazoyer, J.: One head machines from a symbolic approach. Theor. Comput. Sci. **370** (2007) 34–47
2. Gajardo, A., Guillon, P.: Zigzags in Turing machines. In Ablayev, F.M., Mayr, E.W., eds.: CSR. Volume 6072 of Lecture Notes in Computer Science., Springer (2010) 109–119

3. Kari, J., Ollinger, N.: Periodicity and immortality in reversible computing. In Ochmanski, E., Tyszkiewicz, J., eds.: MFCS. Volume 5162 of Lecture Notes in Computer Science., Springer (2008) 419–430
4. Kůrka, P.: On topological dynamics of Turing machines. Theoret. Comput. Sci. **174**(1-2) (1997) 203–216
5. Blondel, V.D., Cassaigne, J., Nichitiu, C.: On the presence of periodic configurations in Turing machines and in counter machines. Theoret. Comput. Sci. **289** (2002) 573–590
6. Oprocha, P.: On entropy and turing machine with moving tape dynamical model. Nonlinearity **19** (October 2006) 2475–2487
7. Kůrka, P.: Topological and Symbolic Dynamics. Société Mathématique de France, Paris, France (2003)
8. Morita, K., Shirasaki, A., Gono, Y.: A 1-tape 2-symbol reversible Turing machine. IEICE TRANSACTIONS **E72-E**(3) (1989) 223–228
9. Morita, K.: Universality of a reversible two-counter machine. Theor. Comput. Sci. **168**(2) (1996) 303–320

Appendix

Construction of the Turing machine shown in the proof or Theorem 2. We will construct a reversible Turing machine $M = (Q, \Sigma, \delta)$ to simulate a 2-reversible counter machine $C = (S, 2, R)$. $Q = S \cup Q_0 \cup S_1 \cup ... \cup S_{|R|}$, where Q_0 is the set of states required for writing "$< \mid >$" on the tape, including q_0, and S_i the set of states needed to simulate the instruction i of R.

$\Sigma = \{<, |, >, 1\} \cup \{0, +\}^2$. The first set recreates the counters on the machine and the second contains auxiliary symbols indicating the status of the counters when reaching a new state.

The transition function is given by a graph, the notation is described in figure 1. It is noteworthy that this machine simulates arbitrary counter machine, so we will describe only generic instructions.

In general, each configuration $(s, (n, m))$ of the counting machine will be represented in the Turing machine by the configuration $(< 1^n | 1^m >, 0, s)$. The machine simulates C from configuration $(s_0, (0, 0))$, by writing "$< \mid >$" on the tape. Subsequently, the machine adds and removes 1's from the tape, accordingly to the instructions of C. The machine will work as long as the background is full of 1's (the new visited cells), if it encounter any other symbol, it stops. It is important to note that, before reaching any state of C, the machine replaces the symbol "$<$" by the pair $(d, d') \in \{0, +\}^2$, in order to indicate the *sign* of each counter at the end of each instruction. In this way, the machine knows which of the instructions of the counter machine is the next to be applied. And this makes the reversibility of the counter machine to be inherited by the Turing machine.

Part 1: Initial configuration. The first action is to write "$< \mid >$" in the tape, see figure 2.

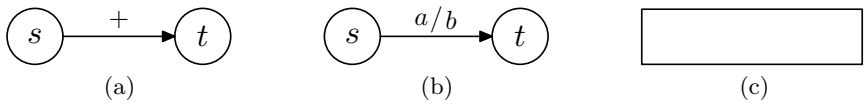

Fig. 1: (a): Instruction $(s, /, +, t)$. (b): Instruction (s, a, b, t). (c): Subroutine.

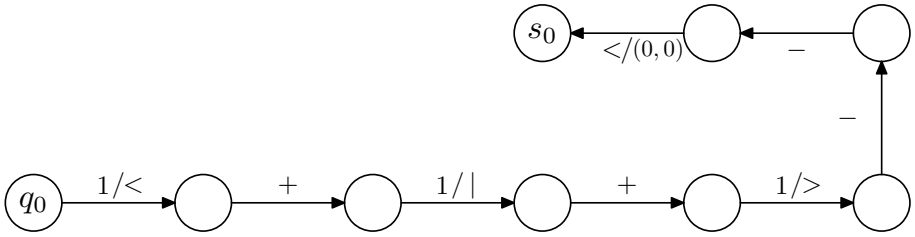

Fig. 2: The routine that writes the sequence "$< | >$" in the tape.

Part 2: Executing instructions. Let us suppose that the following instructions are in R: $(s, (0,0), i, d, t)$, $(s, (0,+), i', d', t')$, $(s, (+,0), i'', d'', t'')$ y $(s, (+,+), i''', d''', t''')$. They are simulated with the routine depicted in figure 3.

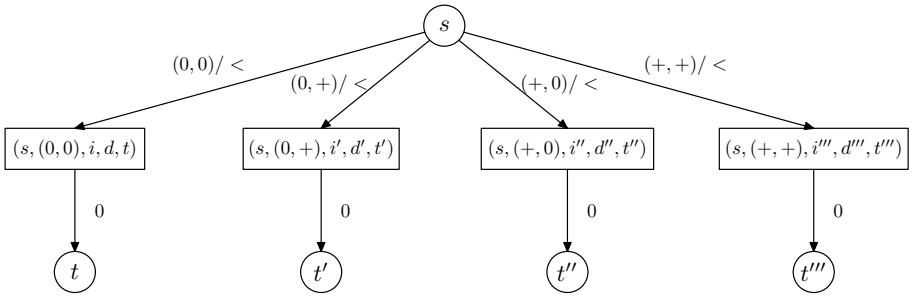

Fig. 3: Depending on the *sign* of the counters, the machine performs the instruction $(s, (0,0), i, d, t)$, $(s, (0,+), i', d', t')$, $(s, (+,0), i'', d'', t'')$ or $(s, (+,+), i''', d''', t''')$.

Part 3: Addition/Subtraction. Each sub-routine uses an exclusive set of states. There are several cases, depending on the *sign* of each counter, but they are all similar, thus we present only two examples in figures 4 and 5.

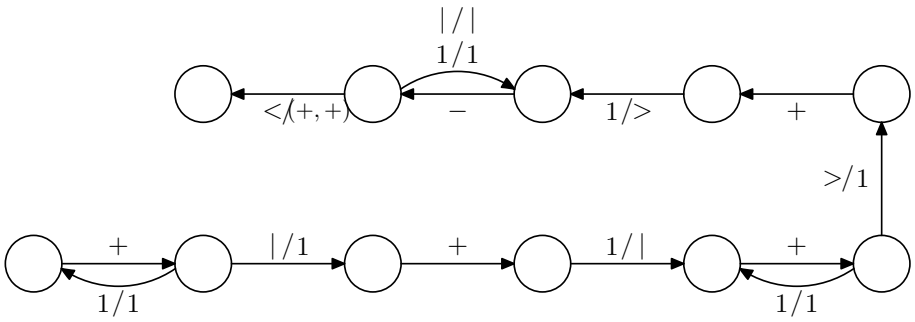

Fig. 4: Sub-routine corresponding to instruction $(s, (+,+), 1, +, t''')$, it adds one unit to counter 1, assuming counter 2 non empty.

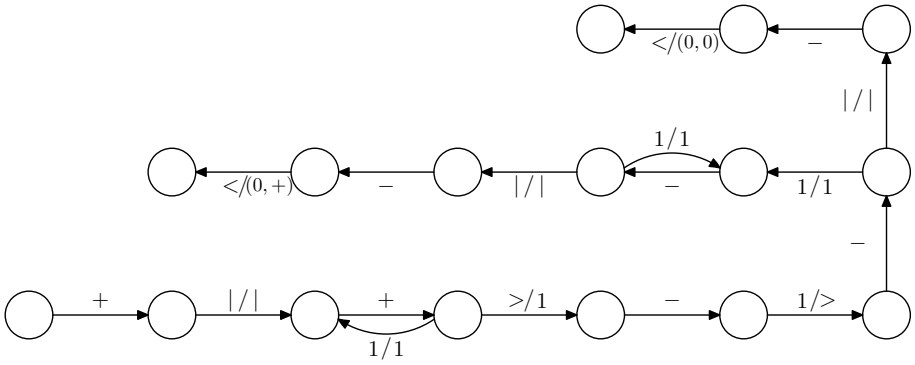

Fig. 5: Sub-routine corresponding to instruction $(s, (0,+), 2, -, t')$, it subtracts from counter 2, assuming counter 1 empty.