Some undecidable problems about the trace-subshift associated to a Turing machine

Anahí Gajardo, Nicolas Ollinger, Rodrigo Torres-Avilés

Discrete Mathematics and Theoretical Computer Science DMTCS vol. **17:2**, 2015, 267–284

Some undecidable problems about the trace-subshift associated with a Turing machine[*]

Anahí Gajardo[1†] Nicolas Ollinger[2] Rodrigo Torres-Avilés[1‡]

[1] *Departamento de Ingeniería Matemática and Centro de Investigación en Ingeniería Matemática, Universidad de Concepción, Chile*

[2] *Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, FR-45067 Orléans, France*

received 8th Sep. 2014, revised 2nd Nov. 2015, accepted 9th Nov. 2015.

We consider three problems related to dynamics of one-tape Turing machines: Existence of blocking configurations, surjectivity in the trace, and entropy positiveness. In order to address them, a reversible two-counter machine is simulated by a reversible Turing machine on the right side of its tape. By completing the machine in different ways, we prove that none of the former problems is decidable. In particular, the problems about blocking configurations and entropy are shown to be undecidable for the class of reversible Turing machines.

Keywords: Turing machine, discrete-time dynamical system, subshift, formal language, entropy

1 Introduction

In the context of dynamical systems, complexity and predictability can be studied within the framework of computational complexity, by embedding a model of computation *inside* the discrete dynamical system. The Turing machine (TM) is the most simple universal computational model. Therefore it is the natural choice for these purposes. Undecidable problems of Turing machines translate into undecidable problems about dynamical systems. Nevertheless, the problems obtained in this way are not natural in the context of dynamical systems. In a recent series of papers (Moore (1991); Kůrka (1997); Gajardo and Mazoyer (2007); Kari and Ollinger (2008); Jeandel (2014); Cassaigne et al. (2014)), Turing machines themselves are defined as dynamical systems. From these works, dynamical properties as *entropy positiveness of two tape Turing machines* in Blondel and Delvenne (2004), and *periodicity* in Kari and Ollinger (2008) and Cassaigne et al. (2014) are proved to be undecidable for Turing machines. Proving undecidability of

[*]Part of this work already appeared in the proceedings of the conference *Reversible Computation 2012* (Torres et al. (2013)).

[†]This work has been supported by CONICYT through FONDECYT #1090568, ECOS-SUD-CONICYT #C12E05 (all the authors) and BASAL project CMM, Universidad de Chile, by CI^{2}MA, Universidad de Concepción.

[‡]Corresponding author: `rtorres@ing-mat.udec.cl`

dynamical properties has the difficulty that these properties talk, in general, about the machine behaviour over *all* its configurations, while classical decision problems suppose that the machine starts on a particular class of configurations (finite, fixed initial state).

In this paper, we focus on three particular problems about Turing machines seen as dynamical systems (blocking configuration, trace-subshift surjectivity, and entropy positiveness), showing some of their interrelationships and undecidability. The common point of these three problems lies in the undecidability proof, which is based on a simulation of reversible counter machines. The present paper is an extension of a paper published in the proceedings of the conference *Reversible Computation*, held in Denmark in 2012 (Torres et al. (2013)). The undecidability of the first two problems was already developed there, together with other simpler related problems. Here we present a more detailed construction of the Turing machine that simulates a reversible counter machine, we further study the interrelationship between these problems and we prove the undecidability of the entropy positiveness problem.

The idea of "blocking mechanism" is not new in dynamical systems. For example, it is important in defining sensitivity of cellular automata. In the context of Turing machines, it consists of a finite configuration and an internal state that avoid the head from moving through it. In the Turing machines context, the head is the only way to pass *information* through the tape, thus connecting blocking in Turing machines to cellular automata. Although not directly addressed, Turing machines frequently have these mechanisms. For example, a TM that works only on the right side of the tape, starts from a blocking configuration. A TM that passes only a bounded number of times over each cell activates a blocking mechanism the last time it visits a cell. The existence of blocking mechanisms has to do with the way the TM uses its memory.

The next two problems come directly from the dynamical system approach. In this area, it is frequent to consider the *trace-subshift*. This is obtained by projecting the phase space into a finite partition, and looking at the trajectories of the system on this partition. This defines a set of semi-infinite sequences called the trace-subshift. The trace-subshift of a TM is the sequence of states and symbols that the head reads over time. It has been previously considered in Moore (1991); Oprocha (2006); Gajardo and Mazoyer (2007); Gajardo and Guillon (2010); Jeandel (2014). We are interested in the entropy of this symbolic system and the surjectivity of the shift function.

The surjectivity of the machine is a simple property; it can be directly determined from the transition function of the machine. Nevertheless, the surjectivity of the shift on the trace-subshift is not equivalent to the surjectivity of the machine and it is less easy to identify. Surjectivity of the trace-subshift implies that all the sequences of the trace can be infinitely extended to the left (the past) and, in fact, the trace-subshift can be defined as a set of bi-infinite sequences without losing any allowed pattern. In this work, we prove that, in the absence of the machine surjectivity, a blocking mechanism is necessary to assure the surjectivity of the trace-subshift. That is why surjectivity as well as the blocking property are undecidable.

Finally, we work with the entropy of the trace-subshift. In Oprocha (2006) it is proved that it coincides with the entropy of the Moving Tape Model of TMs defined by Kůrka (1997). The entropy is a non-negative number that, in a subshift, talks about the diversity of patterns that can be found on it. Blondel and Delvenne (2004) prove that the entropy of a TM with two or more tapes is uncomputable. But Jeandel (2014) gives an algorithm that approaches the entropy of a one-tape TM. This last work shows that the entropy of a TM is related to the rate of new cells that the head visits. It uses an approximation of the graph of crossing sequences (as in Hennie

(1965)). This algorithm approaches the entropy from below with an unknown convergence rate, and the author suggests that it is probably not computable. Here we prove that the entropy, in fact, cannot be distinguished from zero, even for reversible machines.

2 Definitions

2.1 Turing machines

In this paper we consider the Turing machine as a 3-tuple $T = (Q, \Sigma, \delta)$, where Q is a finite set of states, Σ is a finite set of symbols, and δ is the transition function. The model includes also an initial and a final state and a blank symbol, but we omit them here because we are interested in properties of the whole set of states. The machine works on a tape, usually bi-infinite, filled with symbols from Σ. A *configuration* is an element of $Q \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$. A *finite configuration* is an element of $Q \times \{0, 1, ..., n\} \times \Sigma^n$, for any $n \in \mathbb{N}$. The transition function δ can be considered in two forms, the *quintuple model* and the *quadruple model*, and it acts on the configurations differently depending on the model, as follows.

Quintuple model. $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, 0, 1\}$. The machine T transforms a configuration (q, i, c) into $(q', i + d, c')$, if $\delta(q, c_i) = (q', \alpha', d)$, and c' is defined by $c'_i = \alpha'$ and $c'_j = c_j$ for all $j \neq i$.

Quadruple model. $\delta : \left| \begin{array}{ccc} Q_W \times \Sigma & \to & Q \times \Sigma \\ Q_M \times \{/\} & \to & Q \times \{-1, 0, 1\} \end{array} \right.$, where $Q = Q_W \sqcup Q_M$. The machine transforms a configuration (q, i, c) into (q', i, c'), if $q \in Q_W$, $\delta(q, c_i) = (q', \alpha')$ and c' is defined by $c'_i = \alpha'$ and $c'_j = c_j$ for all $j \neq i$, but if $q \in Q_M$, then (q, i, c) is transformed into $(q', i + d, c)$, where $\delta(q, /) = (q', d)$. The first type of transformation is called *writing instruction*, and the second one *movement instruction*.

A machine in quadruple model can be directly transformed into the quintuple model, because writing instructions can be replaced by instructions with 0 movement, and movement instructions can be replaced by $|Q|$ instructions of the quintuple model. The global function T does not change with this transformation.

A machine in quintuple model can also be transformed into the quadruple model; but the global function changes, because each instruction will need to be changed by two instructions.

2.2 Dynamical systems

A *topological dynamical system* is a pair (X, T), where X is a topological space called *phase space* and $T : X \to X$ is a continuous map called *global transition function* (see for example Kůrka (2003)). For each $x \in X$, the *orbit of* x is the infinite sequence $\mathcal{O}(x) = (T^n(x))_{n \in \mathbb{N}}$. It describes the dynamics that T induces over the point x. An important tool to study a dynamical system consists in projecting X into a finite set, and considering the set of infinite sequences obtained from the set of orbits projected in this way.

2.2.1 Subshifts and languages

Given a finite set A, called *alphabet*, the set $A^{\mathbb{Z}}$ is the *two-sided full shift*, while $A^{\mathbb{N}}$ is the *one-sided full shift*. The *shift* function σ is defined both in $A^{\mathbb{Z}}$ and $A^{\mathbb{N}}$ by $\sigma(y)_i = y_{i+1}$, and it is a bijective

function in the first case. A^* denotes the set of finite sequences of elements of A, called *words*. It includes the empty word ϵ. A finite word v is said to be a *subword* of another (finite or infinite) word z, if there exist two indices i and j, such that $v = z_i z_{i+1} ... z_j$. In this case, we write: $v \sqsubseteq z$. The length of a word u is denoted by $|u|$.

Given a subset S of $A^{\mathbb{Z}}$ or $A^{\mathbb{N}}$, a formal language is defined by $\mathcal{L}(S) = \{u \in A^* \ : \ \exists z \in S, \ u \sqsubseteq z\}$. Reciprocally, given a formal language L, a set of right-infinite sequences can be defined: $\mathcal{S}_L = \{z \in A^{\mathbb{N}} \ : \ \forall u \sqsubseteq z, \ u \in L\}$. When $S \subseteq A^{\mathbb{N}}$ satisfies $\mathcal{S}_{\mathcal{L}(S)} = S$, it is called a *one-sided subshift*. Two-sided subshifts are defined in an analogous way.

We also define the operator $\cdot|_n \ : \ A^{\mathbb{N}} \to A^n$ by $z|_n = z_0 z_1 ... z_n$. It can also be defined for sets $\cdot|_n : \mathcal{P}(A^{\mathbb{N}}) \to \mathcal{P}(A^n)$ by $S|_n = \{z|_n \ : \ z \in S\}$.

When S is a subshift, it is closed for the action of σ, and the system (S, σ) is a dynamical system.

For more details about this subject see for example Kůrka (2003).

2.3 The Turing machine seen as a dynamical system

If δ is a total function, then T also defines a total function from $X = Q \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$ into itself, and (X, T) is a dynamical system. In this case we say that the machine is *complete*. The function T can be backward-deterministic (*reversible machine (RTM)*) if each configuration has *at most* one predecessor, and it can also be backward-complete (*surjective machine*) if each configuration has *at least* one predecessor.

When the machine is complete (in other words, the machine does not halt), surjectivity and reversibility are equivalent: Indeed, a complete and reversible Turing machine is surjective, as stated in Kari and Ollinger (2008). For the converse, if a Turing machine $M = (Q, \Sigma, \delta)$ is surjective, for every $q \in Q$, in the image of δ exists, or $|\Sigma|$ different pairs (state, symbol), or at least one pair (state, movement). If the machine is complete, the same happens with the pre-image of δ. Now, let us suppose that M is not reversible; then, there exists two pairs in the pre-image of δ leading to the same pair in the image. As the amount of instructions is limited, then it will exists a pair (state, symbol) or (state, movement) without a pre-image, and therefore, the machine M is not surjective, leading to a contradiction.

In this case, T is a *reversible and complete Turing machine (c-RTM)*.

We define $\pi : X \to Q \times \Sigma$ by $\pi(q, i, c) = (q, c_i)$. The *trace-subshift* associated to T is the set

$$S_T = \{(\pi(T^n(x)))_{n \in \mathbb{N}} \ : \ x \in X\} \subseteq (Q \times \Sigma)^{\mathbb{N}}.$$

It is not difficult to see that S_T is in fact a subshift Gajardo and Mazoyer (2007) and that the function $\tau : X \to S_T$, defined by $\tau(x) = (\pi(T^n(x)))_{n \in \mathbb{N}}$, satisfies that $\tau \circ T = \sigma \circ \tau$. In other words, the system (S_T, σ) is a *factor* of (X, T).

We will also consider *partial configurations* as 3-tuples of the form (q, i, u), where u is a word that specifies the values of the cells in the interval $\{0, ..., |u|-1\}$ and $0 \leq i \leq |u|-1$. The function T applied over partial configurations is not total, being undefined when the head overpasses the limit of the partial configuration. We also extend τ to partial configurations $\tau : Q \times \mathbb{Z} \times \Sigma^* \to \mathcal{L}(S_T)$ by $\tau(q, i, u) = \left(\pi(T^j(q, i, u))\right)_{j \in \{0, ..., m\}}$, where $m \in \mathbb{N}$ is the number of instructions that can be applied before the head exits the interval $\{0, ..., |u| - 1\}$.

3 Problems and concepts

3.1 Blocking states

The idea of a "blocking" configuration that avoids the head from going beyond some limit appears in several contexts as cellular automata (Blanchard and Tiseur (2000)) and subshifts (Lind and Marcus (1995)), and it is related to stability and information travel. In the context of Turing machines, there are several ways of defining it. In a first approach, one can think in a sequence of cells that acts as a "wall": if the head goes over these cells, it will read the symbols that they contain and it will "rebound". But the machine can change these symbols, then the new configuration should be also "blocking" to maintain the "wall" effect. On the other hand, the state of the machine is also important; the blocking effect can be a consequence of the combination between the state and the symbols in the tape. In order to fix this notion, we consider the following definitions.

Definition 1 *Given a word $u \in \Sigma^*$, we say that the partial configuration $(q, 0, u)$ is a* blocking configuration to the left *if for every extension c of u and every time n, the position of the head in $T^n(q, 0, c)$ is greater than or equal to 0. Analogously, we say that $(q, |u| - 1, u)$ is a* blocking configuration to the right *if for every extension c of u and every time n, the position of the head in $T^n(q, |u| - 1, c)$ is less than or equal to $|u| - 1$.*

- *If $|u| = 1$ we say that (q, u_0) is a* blocking pair to the left (right).
- *If $u = \epsilon$, we just say that q is a* blocking state to the left (right).

Finally, we say that q is just a blocking state *if for every $\alpha \in \Sigma$, (q, α) is a blocking pair either to the left or to the right.*

These definitions assume that the head starts at the left (right) border of the blocking configuration. One may want a notion of blocking that works independently from the position or the state of the head, so other possibilities exist.

We consider now the problems of deciding whether a configuration, pair or state is blocking.

(BC-d) Given a Turing machine T and a partial configuration $(q, 0, c)$ (or $(q, |c|-1, c)$ for $d = 1$), decide whether the configuration is blocking in direction d.

(BP-d) Given a Turing machine T and a pair (q, α), decide whether (q, α) is a blocking pair in direction d.

(BS-d) Given a Turing machine T and a state q, decide whether q is a blocking state in direction d.

(BS) Given a Turing machine T and a state q, decide whether q is a blocking state.

It is important to note that the complexity of these problems does not depend on whether the machine is in the quadruple or quintuple model, since the blocking property talks about the long-term movements of the head.

Since (BS-d) reduces to (BS), (BC-d) and (BP-d), the undecidability of the first implies the undecidability of the other three. The undecidability of (BS-*left*) can be directly obtained by reduction from the Emptiness problem of Turing machines. The complete proof can be seen in Torres et al. (2013).

We can also consider the reversible versions of these problems. The importance of considering reversible Turing machines lies in the implications of blocking information in properties related to reversibility, such as surjectivity. This relation can be seen in the next section. The undecidability of the reversible version will be established in Section 5.1.

3.2 Surjectivity

A function $T : X \to X$ is called *surjective* if it is onto, i. e., if every point has a pre-image; in other words, if $T(X) = X$. Surjectivity of Turing machines is a local property; one can decide if a configuration has or not a pre-image just by looking at the state of the head and its surrounding symbols.

Definition 2 *A state $q \in Q$ of a Turing machine $T = (Q, \Sigma, \delta)$ is said to be defective if:*

1. *(Quintuple model) There exist symbols α, α', $\alpha'' \in \Sigma$ such that neither $(q, \alpha, +1)$, $(q, \alpha', 0)$ nor $(q, \alpha'', -1)$ belong to the image of δ.*
2. *(Quadruple model) There exist $\alpha' \in \Sigma$ such that neither $(q, 1)$, $(q, 0)$, $(q, -1)$ nor (q, α') belong to the image of δ.*

Given $d \in \{-1, 0, 1\}$, we also define the set $D_d(q) = \{\alpha \in \Sigma : (\nexists q' \in Q)(\nexists \lambda \in \Sigma)\ \delta(q', \lambda) = (q, \alpha, d)\}$.

Property 1 *A machine T is surjective if and only if it has no defective state.*

Proof: If T has no defective state, then every configuration (q, i, w) has at least one pre-image, and T is thus surjective.

If T has a defective state q, then there are configurations without pre-image, in fact, if α, α', $\alpha'' \in \Sigma$ are the symbols that make q defective in the quintuple model, then no extension of the configuration $(q, 1, \alpha\alpha'\alpha'')$ has a pre-image. In the quadruple model, if α' makes q defective, then no extension of the configuration $(q, 0, \alpha')$ has aa pre-image because q cannot be attained through a movement instruction neither through a writing instruction that writes α'. □

Remember that when δ is a total function, surjectivity of T is equivalent to its injectivity. In this work we assume that δ is total.

3.2.1 Surjectivity of (S_T, σ).

In a subshift, surjectivity simply means that every element is *extensible to the left*: A subshift $S \subseteq A^{\mathbb{N}}$ is surjective if and only if: $\forall w \in S, \exists a \in A,\ aw \in S$.

Since (S_T, σ) is a factor of (X, T), the surjectivity of T is inherited by σ in S_T. However, depending on the model, if T is not surjective, σ can still be surjective in S_T.

In the quadruple model, the surjectivity of T is held by the subshift S_T and vice versa. In fact, if we have a defective state q_0, then there exist no moving instruction leading to q_0. Thus, in the

previous step, the head is on the same cell, and if (q, α) can precede (q_0, α_0) in S_T, one needs $\delta(q, \alpha) = (q_0, \alpha_0)$; thus q_0 is not defective.

However, in the quintuple model, there exist non surjective machines with a surjective trace-subshift. Let us illustrate this in the following example.

Example 1 *Let T be the Turing machine that simply moves to the right by always writing a 0. This machine is not surjective, but its associated subshift is. 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 beginning of any $w \in S_T$.*

Surjectivity will be possible when defective states avoid the head from going into the "conflictive" positions. If $w = \left(\begin{smallmatrix} q_0 & q_1 & \cdots \\ \alpha_0 & \alpha_1 & \ldots \end{smallmatrix}\right) \in S_T$ and $a = \binom{q}{\alpha}$, condition $aw \in S_T$ says that $\delta(q, \alpha) = (q_0, \beta, d)$ and that the configuration which produces w should have the symbol β at position $-d$. If the machine does not visit the position $-d$ any more, β can be any symbol. We next give a necessary and sufficient condition for a machine to have a surjective trace-subshift.

Definition 3 *A state q is said to be* reachable from the left (right) *if there exist a state q' and symbols α, α' such that $\delta(q', \alpha') = (q, \alpha, 1)$ (resp. -1).*

Proposition 1 *The function σ is surjective on S_T if and only if for each $q \in Q$ at least one of the following holds:*

1. *q is not defective.*
2. *q is blocking to the left (right) and is reachable from the left (right).*
3. *for every $\alpha \in D_1(q)$, $\alpha' \in D_0(q)$ and $\alpha'' \in D_{-1}(q)$ either*
   - *(a) $(q, 0, \alpha'\alpha'')$ is blocking to the left and q is reachable from the left, or*
   - *(b) $(q, 1, \alpha\alpha')$ is blocking to the right and q is reachable from the right.*

Proof: We first remark that 2 implies 3; thus 2 can be suppressed from this proposition. We chose to state it this way because property 2 will be relevant in the next sections.

($\Rightarrow$) Let us suppose that σ is surjective in S_T, and let us suppose that neither 1 nor 3 hold for some q. Let us suppose, of course, that q is reachable (i.e. from the left, the right or though a transition without movement); otherwise σ cannot be surjective. Let α, α' and α'' be on D_1, D_0 and D_{-1}, respectively, such that they make statement 3 false.

Let us suppose first that $\delta(q, \alpha') = (q', \gamma, +1)$ for some $\gamma \in \Sigma$ and $q' \in Q$. Now, let $x = (q, 0, ...\alpha\alpha'\alpha''...)$, with α' at position 0, and $w = \left(\begin{smallmatrix} q & q' & \cdots \\ \alpha' & \alpha'' & \ldots \end{smallmatrix}\right) = \tau(x)$. We can assume that q is reachable from the left; otherwise, w cannot have a pre-image because, by hypothesis, α' and α'' cannot be written at the position where they are. As statement 3 does not hold for q, $(q, 0, \alpha'\alpha'')$ is not blocking to the left. We can take x as the extension of $(q, 0, \alpha'\alpha'')$ that makes it not blocking, and put α at position -1. We can see that w cannot be extensible because, if x is produced with a movement from the left, α should be written. Therefore, S_T is not surjective and we have a contradiction.

Now, if $\delta(q, \alpha') = (q', \gamma, -1)$, the proof is analogous.

The last case is $\delta(q, \alpha') = (q', \gamma, 0)$. We have two possibilities: 1) The head eventually moves in some direction; in this case, we proceed exactly as before in this direction. 2) The head remains in the same cell forever; we have a contradiction again because, in this case, (q, α') is a blocking pair in both directions, and then q is only reachable from the center. Now, let x be a extension of $(q, 0, \alpha')$, and let $w = \left(\begin{smallmatrix} q & q' & \cdots \\ \alpha' & \gamma & \cdots \end{smallmatrix}\right) = \tau(x)$. If $w' = \left(\begin{smallmatrix} p & q & q' & \cdots \\ \beta & \alpha' & \gamma & \cdots \end{smallmatrix}\right)$ is a pre-image of w, then $\delta(p, \beta) = (q', \alpha', 0)$, which is impossible because $\alpha' \in D_0(q)$.

($\Leftarrow$) Let $w = \left(\begin{smallmatrix} q & q' & \cdots \\ \alpha_0 & \alpha_1 & \cdots \end{smallmatrix}\right)$ be an element of S_T, generated by a configuration x.
Let us suppose that 1, 2 or 3 hold for q.

1. If q is not defective, then, independently on the context, x can be reached from some configuration.

2. If q happens to be a blocking state to the left (right), no configuration producing w is able to revisit the position -1 $(+1)$ (with respect to the initial head position), so any $(q'', \alpha) \in Q \times \Sigma$, such that $\delta(q'', \alpha) = (q, \alpha', +1)$ $((q, \alpha', -1))$ for some α', can be appended at the beginning of w, and such pair exists because q is reachable from the left.

3. Let us suppose that $x = (q, 0, ...v_{-1}v_0v_1)$ (with $v_0 = \alpha_0$). If $v_d \notin D_{-d}(q)$, for some $d \in \{-1, 0, 1\}$, then we can extend w to the left with a pair (p, λ) such that $\delta(p, \lambda) = (q, v_d, -d)$. If $v_d \in D_{-d}(q)$, for all $d \in \{-1, 0, 1\}$, then, due to 3, we can extend w to the left with any $(q, \lambda) \in Q \times \Sigma$, such that $\delta(q'', \lambda) = (q, \lambda', d')$ for some $\lambda' \in \Sigma$, with d' depending on whether v_{-1},v_0 and v_1 satisfy 3.*a* or 3.*b*.

Therefore, w has a preimage by σ. Since w is arbitrary, σ is surjective on S_T. □

We consider the following problem.

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

(Surj) is related to (BS-d) and (BP-d) as can be appreciated in Proposition 1. They are all undecidable, as we will see in Section 5.

3.3 Positive entropy

The *entropy* of a subshift S is given by the following limit,

$$H(S) = \lim_n \frac{1}{n} \log(\#(S|_n))$$

This definition is based on the *words* of the trace-subshift, which correspond to partial developments of the machine dynamics. The interesting work of Brudno (1983) proves that the entropy of a subshift can be computed as a maximum over its elements, i. e., over complete traces; more precisely, a maximum of the *prefix-free Kolgomorov complexity* (as now on called just Kolgomorov complexity for simplicity) of the elements (see for example Downey and Hirschfeldt (2010)). The Kolgomorov complexity $K(u)$ is defined as the length of a shortest program that computes the finite word u. For an infinite sequence $w \in S$, the *upper complexity* of w is defined by $\overline{K}(w) = \lim_n \sup \frac{K(w|_n)}{n}$. Brudno proves that the entropy of a given subshift S is given by the next equality.

Theorem 1 (Brudno (1983))

$$H(S) = \max_{w \in S} \overline{K}(w)$$

There is a rich theory about Kolmogorov complexity, a good reference on the subject is the book of Downey and Hirschfeldt (2010). Here we will only use two classical properties.

Property 2 *1. There exists a constant C such that, for all finite words v, v', $K(vv') \leq K(v) + K(v') + C$.*

2. For every $\alpha \in \Sigma$ and $n \in \mathbb{N}$, $K(\alpha^n) = O(log(n))$.

These allow us to prove the next simple lemma.

Lemma 1 *For any finite word u and symbol α, it holds that $\overline{K}(u\alpha^{\mathbb{N}}) = 0$.*

Proof: If we define $w = u\alpha^{\mathbb{N}}$, from the last properties, we have

$$\overline{K}(w) = \lim_n sup \frac{K(w_{|n})}{n} \leq \lim_n sup \frac{K(u) + K(\alpha^{n-|u|}) + C}{n} = 0.$$

□

In Jeandel (2014), an algorithm that approximates the entropy of the trace-subshift of a one-dimensional Turing machine is developed. His result is strongly based on the next two properties.

Property 3 (Jeandel (2014)) *1. A positive upper complexity is reached on configurations on which the head visits each position of the tape finitely many times.*

2. From the last assertion, we know that some of the configurations with maximal upper complexity never cross the origin.

We will show that, although computable by approximation, the entropy of a trace-subshift cannot be distinguished from 0 in finite time. In other words, we will prove that the next problem is undecidable.

(PE) Given a deterministic and complete Turing machine T, decide whether $H(S_T) > 0$.

Remark 1 *It is important to point out that, if we transform a TM from the quintuple model to the quadruple model, the trace-subshift and its entropy change, but only by a factor of two. Thus, its positiveness is not affected. Thus the complexity of (PE) does not depend on the model in which T is expressed.*

4 Simulating counter machines

The main difficulty when proving undecidability of a TM property that comes from dynamical systems theory is that it has to do with the whole set of trajectories; it is not restricted to trajectories that start at particular points. The three undecidability proofs that we present here are based on a machine that persistently simulates a counter machine from a particular state. In this way, some of the dynamical properties of our Turing machine will talk about the behaviour of the counter machine over its initial state, inheriting, in this way, some of its undecidable properties.

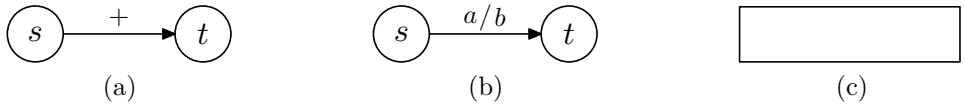


Fig. 1: (a): Instruction $\delta(s, /) = (t, +)$. (b): Instruction $\delta(s, a) = (t, b)$. (c): Subroutine.

Definition 4 *A k-counter machine (k-CM) is a triple* (Ω, k, R)*, where* Ω *is a finite set,* $k \in \mathbb{N}$ *is the number of counters, and* $R \subseteq \Omega \times \{0, +\}^k \times \{1, .., k\} \times \{-1, 0, +1\} \times \Omega$ *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* $(s, \theta, i, d, t) \in R$ *can be applied to a configuration* (s, ν) *if* $\text{sign}(\nu) = \theta$*, 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* $(s, \theta, i, -1, t)$ *if* $\theta_i = 0$*.*

Since the transition relation is not necessarily a function, the defined system can be nondeterministic, or not defined for some configurations. We will restrict our attention to deterministic and one-to-one 2-counter machines (this will be called 2-RCM, for reversible 2-counter machine). Morita (1996) proves that a two-counter machine is reversible if for every state $s \in \Omega$ and $\theta \in \{0, +\}^2$, the machine can reach a configuration (s, ν), with $sign(\nu) = \theta$, from at most one state.

In the next section, we define a reversible Turing machine that simulates a given arbitrary 2-RCM. The simulation is fairly standard, using special symbols to separate each counter, which are represented by series of cells containing 1s. In a first stage we define an incomplete RTM, but in Section 4.2, we recall one of the methods presented in Kari and Ollinger (2008) that allows to complete any reversible transition rule.

4.1 Construction of the reversible Turing machine that simulates a 2-RCM.

Given a 2-RCM $C = (\Omega, 2, R)$, we construct a reversible Turing machine $T_C = (Q_C, \Sigma, \delta)$ defined in quadruples. Its state set is $Q_C = \Omega \cup Q_0 \cup \Omega_1 \cup ... \cup \Omega_{|R|}$, where Q_0 is the set of states destined for initialisation, and Ω_i is the set of states needed to simulate the i-th instruction of R. The symbol set is $\Sigma = \{<, |, >, 1\} \cup \{0, +\}^2$. The first set recreates the counters on the machine, and the second contains auxiliary symbols indicating the status of the counters. We describe the transition function of T_C with figures; the used notation is specified in Figure 1.

The idea behind this simulation is that each configuration $(s, (n, m))$ of the counter machine will be represented in the Turing machine by the configuration $(s, 0, ...(sign(n), sign(m))1^n|1^m > ...)$, with the symbol $(sign(n), sign(m))$ at position 0. The machine simulates C starting from configuration $(s_0, (0, 0))$ by writing "<|>" on the tape. State $q_0 \in Q_0$ initiates a subroutine (see Figure 2) that first writes " < | > " on the tape, and then comes back to replace " < " by "(0, 0)" and to pass to state s_0. Subsequently, the machine adds and removes 1's from the tape, according to the instructions of C.

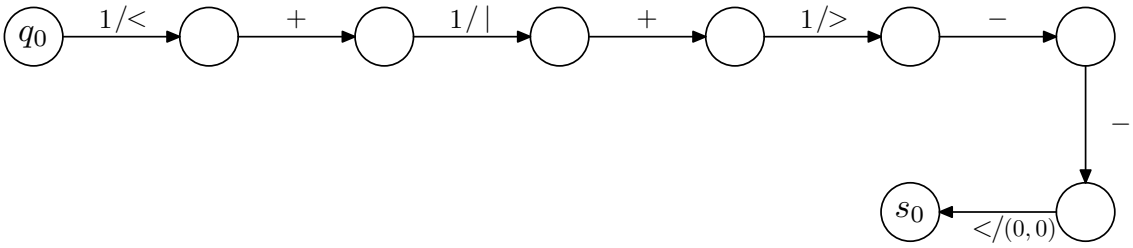


Fig. 2: The routine that writes the sequence "< | >" in the tape.

Each sub-routine uses an exclusive set of states. The state set Ω_i is dedicated to perform the i-th instruction of R. There are several cases, depending on the action over the counters, the affected counter and its *sign*. There are eight addition instructions represented by four subroutines that can be found in Table 1. There are only four subtraction instructions, since the counter must be non-empty if we want to subtract from it. Subtractions are represented in Table 2. Instructions with no action on the counter are simpler; instruction $(s, (d, d'), i, 0, t)$ is simply represented by the transition $\delta(s, (d, d')) = (t, (d, d'), 0)$.

The machine will work as long as the background is full of 1's (the new visited cells). If it encounters any other symbol, it halts. 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 instruction of the counter machine is the next to be applied. If T_C is in a state of C, reading a sign (d, d') will call a subroutine which executes the corresponding instruction of R. This is illustrated in Figure 3.

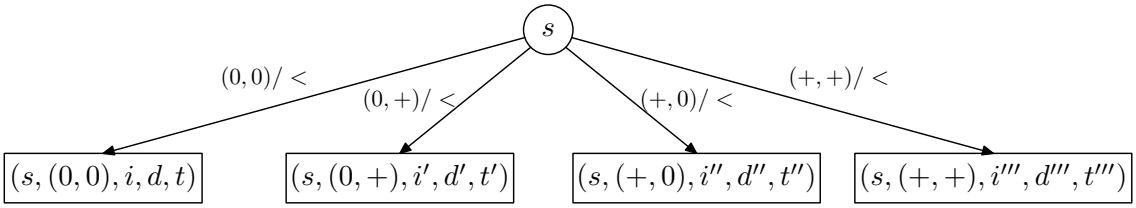
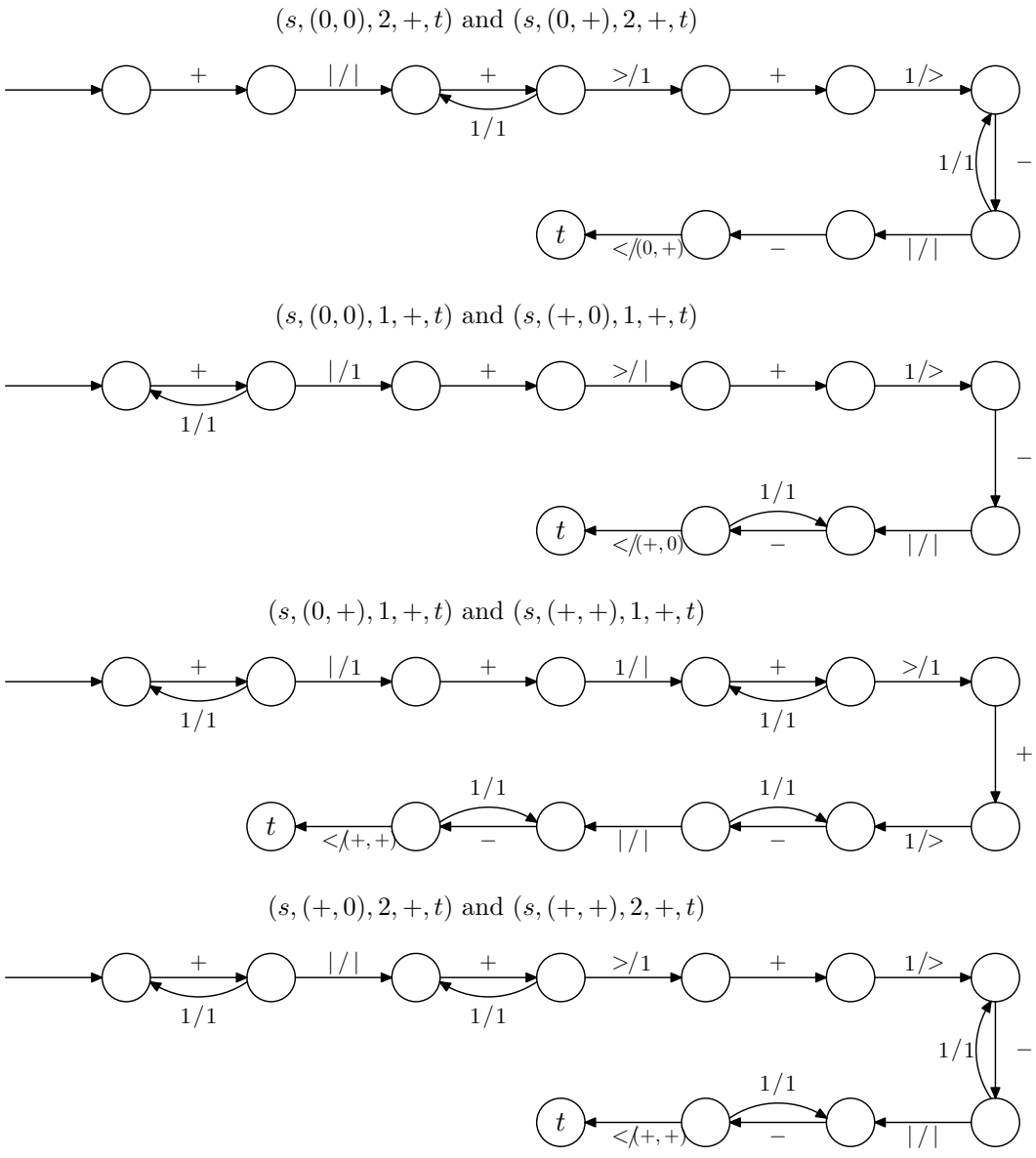


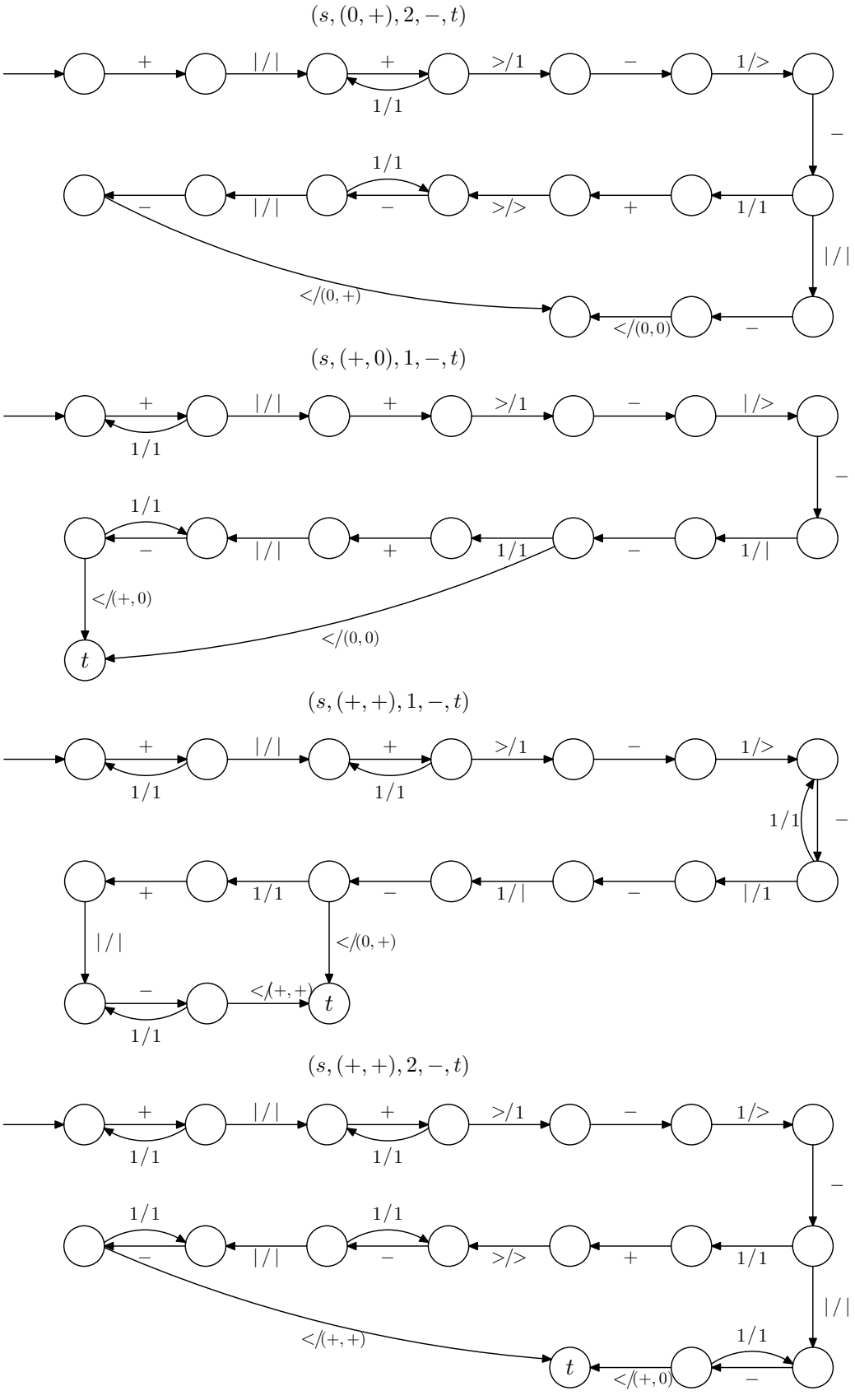
Fig. 3: Depending on the *sign* of the counters, the machine performs the instruction $(s, (0, 0), i, d, t)$, $(s, (0, +), i', d', t')$, $(s, (+, 0), i'', d'', t'')$ or $(s, (+, +), i''', d''', t''')$.

The head will always come back to the position where the symbol "<" was placed while the computation is simulated correctly. T_C will halt if at some moment it encounters an unexpected symbol. It will also halt if it attains a halting state of C.

Remark 2 *The Turing machine T_C that simulates 2-RCM machine C is reversible.*
In fact, in the quadruple model, a machine is reversible if for every two different instructions



Tab. 1: Sub-routines corresponding to the different adding instructions.



Tab. 2: Sub-routines corresponding to subtraction instructions.

going to the same state $\delta(q_1, \alpha) = (q, \alpha'_1), \delta(q_2, \alpha_2) = (q, \alpha'_2)$, we have that $\alpha'_1 \neq \alpha'_2$ and $\alpha_1 \neq /, \alpha_2 \neq /$. From Tables 1 and 2, and Figures 2 and 3 it can be directly verified that every state in $Q_0 \cup \Omega_1 \cup ... \cup \Omega_{|R|}$ satisfies these requirements. We need to be more careful with the states in Ω. Different sub-routines can arrive to the same state in Ω, but they always write the sign of the counters on the tape. Therefore, since C is reversible, T_C will always write different signs when arriving to the same state from different subroutines, which imply that T_C is reversible.

4.2 Reversing the computation

Kari and Ollinger (2008) present a technique that allows to define a complete and reversible Turing machine starting with a reversible Turing machine, by just adding instructions. We recall it here. Given a reversible Turing machine $T = (Q, \Sigma, \delta)$ written in quadruples, a new machine $T' = (Q \times \{\triangleleft, \triangleright\}, \Sigma, \delta')$ is defined as follows.

$$\begin{aligned} \delta(q, /) = (q', d) &\Rightarrow \delta'((q, \triangleright), /) = ((q', \triangleright), d) \text{ and } \delta'((q', \triangleleft), /) = ((q, \triangleleft), -d) \\ \delta(q, \alpha) = (q', \alpha') &\Rightarrow \delta'((q, \triangleright), \alpha) = ((q', \triangleright), \alpha') \text{ and } \delta'((q', \triangleleft), \alpha') = ((q, \triangleleft), \alpha) \end{aligned}$$

In other words, $(q, \triangleright)$ and $(q, \triangleleft)$ represent T in state q running forwards or backwards, respectively.

Moreover, if at some iteration no instruction of T can be applied, the "time direction" is switched. This is performed by adding the next instructions.

$$\begin{aligned} \text{no transition defined for } q &\Rightarrow \delta'((q, \triangleright), /) = ((q, \triangleleft), 0) \\ \text{else if neither } \delta(q, /) \text{ nor } \delta(q, \alpha) \text{ are defined} &\Rightarrow \delta'((q, \triangleright), \alpha) = ((q, \triangleleft), \alpha) \\ \text{no transition arrives to state } q &\Rightarrow \delta'((q, \triangleleft), /) = ((q, \triangleright), 0) \\ \text{else if neither } (q, \alpha), (q, -1), (q, 0) \text{ nor } (q, 1) \text{ are in the image of } \delta &\Rightarrow \delta'((q, \triangleleft), \alpha) = ((q, \triangleright), \alpha) \end{aligned}$$

We will apply this technique not only to machine T_C but also to some modified versions of it.

5 Undecidability of the problems

5.1 Undecidability of the blocking state problem in complete RTMs

The problem we study in this section is not exactly the one described in Section 3.1. It is a restriction of the reversible version of (BS-*left*), thus it directly reduces to (BS-*left*). The interest behind this restriction will be clear in Section 5.2.

(BS-left Artm) Given a complete and reversible Turing machine T 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 Morita (1996). The halting problem in this

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

(Halt2rcm) Given a 2-RCM C and two states s_0 and s_f, decide whether C arrives to s_f when starting from configuration $(s_0, (0, 0))$.

Theorem 1 *(BS-left Artm) is undecidable.*

Proof: Let $C = (\Omega, 2, R)$ be a 2-RCM, with initial configuration $(s_0, (0, 0))$ and final state $s_f \in \Omega$. We will define a Turing machine T and a state $q = q_0$ meeting the next conditions:

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 s_f) 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, C halts when it starts from $(s_0, (0, 0))$.

The machine T_C defined in Section 4.1 satisfies the first 2 objectives.

Now, in order to reach the third goal, we add an extra state and an extra rule to T_C: $\delta(s_f, /) = (q_{aux}, -1)$. In this way, T_C is able to reach the position -1, starting from q_0, if and only if C halts when it starts from $(s, (0, 0))$. It is noteworthy to say that, as s_f is a final state, the machine is still reversible.

We next apply the technique *reversing the computation* described in Section 4.2 to obtain a complete and reversible Turing machine $T'_C = (Q', \Sigma, \delta')$, with $Q' = Q_C \times \{\triangleright, \triangleleft\}$.

The fifth goal is attained by modifying T'_C in only one instruction:

$\delta'((q_0, \triangleleft), /) = ((q_0, \triangleright), 0)$ is switched to $\delta'((q_0, \triangleleft), /) = ((q_0, \triangleright), 1)$.

This instruction exists because q_0 is not attained by δ.

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

□

We would like to remark that in this proof we assume T in the *quadruple* model; but T_C can be transformed to the *quintuple* model at the end of the proof, the result remaining the same.

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

We will use problem (BS-left Artm) to prove the undecidability of the surjectivity on the trace-subshift.

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

Theorem 2 *(Surj) is undecidable.*

Proof: We prove this by reduction from (BS-left Artm). Let $T = (Q, \Sigma, \delta)$ be a complete and reversible Turing machine written in quintuples. Let q' be a state that is reachable from the left, and let q, α and α' be such that $\delta(q, \alpha) = (q', \alpha', +1)$. We know that this machine is surjective.

Now let us define T' as T, but with an additional state q_{aux} and the following new instructions:

$$(\forall \beta \in \Sigma)\ \delta(q_{aux}, \beta) = (q', \beta, 0)\ . \tag{1}$$

And changing

$$\delta(q, \alpha) = (q', \alpha', +1) \text{ by } \delta(q, \alpha) = (q_{aux}, \alpha', +1). \tag{2}$$

T' is not surjective, because the configuration (q_{aux}, i, w) has not preimage if $w_{i-1} \neq \alpha'$. So q_{aux} is the unique defective state of T', with $D_1(q_{aux}) = \Sigma - \{\alpha'\}$ and $D_0(q_{aux}) = D_{-1}(q_{aux}) = \Sigma$.

In this way, if q' is a blocking state to the left for T, then so is q_{aux} for T', and it is also reachable from the left. Therefore, by Proposition 1, σ is surjective in $S_{T'}$ if and only if q' is a blocking state to the left for T.

□

5.3 Undecidability of the entropy positiveness on reversible one-tape Turing machines

(PE) Given a deterministic and reversible complete Turing machine, decide whether its entropy is 0.

Theorem 3 *(PE) is undecidable.*

Proof: We prove it by reduction from the halting problem with empty counters of 2-RCM.

Let $C = (\Omega, 2, R)$ be a 2-RCM, with initial configuration $(s_0, (0, 0))$ and final state $s_f \in \Omega$. For this proof we use the Turing machine T_C defined in Section 4.1. This machine:

1. simulates C on the right side of the tape,
2. is reversible.

We distinguish four types of configurations:

1. Configurations where T_C finds error(s) in its computation and halts.
2. Configurations where T_C goes to unbounded searches of symbols like $<, |, >$.
3. Configurations where T_C has a successful infinite computation of C.
4. Configurations where T_C has a successful finite computation of C.

We slightly modify T_C by adding the rules depicted in Figure 4. These rules allow a new computation to start if the initial one is achieved. A second simulation will correspond to the machine C starting from $(s_0, (0, 0))$.

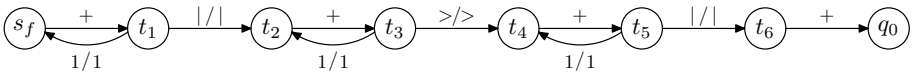


Fig. 4: Sequence of states added to T_C.

Finally, we use again the technique *reversing the computation*, as described in Section 4.2, to obtain a complete machine T'_C. In this new machine, configurations of type 1 simply loop into a periodic behaviour, always visiting the same cells. Its upper complexity is 0.

The upper complexity of configurations of type 2 is 0 by Lemma 1, since all the searches are done over symbol 1. Configurations of type 3 persistently visit the starting cell. By a remark from Jeandel (2014), its upper complexity is 0. Finally, configurations of type 4 will reach state s_f and start a search of symbol | to start a simulation of C from $(s_0, (0, 0))$.

If the machine C halts from $(s_0, (0, 0))$, T'_C can repeat simulations of C several times. More precisely, we can consider a configuration $(s_0, 0, w)$, where $w \in (u + v)^{\mathbb{Z}}$, with $u = 1^r|$ and $v = 1^{|\tau((q_0,0,u))|}$. If r is taken as the minimum space needed to perform a complete simulation of C from $(s_0, (0, 0))$, T_C will make a simulation of C with the same frequency as the word u appears in w. We thus have a lower bound for the entropy of $S_{T'_C}$:

$$H(T'_C) = \lim_n \frac{1}{n} \log(\#(S_{T'_C}|_n)) \geq \lim_k \frac{\log(2^k)}{k(|\tau((q_0, 0, u))|)} = \frac{\log 2}{|\tau((q_0, 0, u))|} > 0$$

On the other hand, if the machine C does not halt from $(s_0, (0, 0))$, even if a configuration of type 4 reaches s_f, T'_C will start a search of symbol "|". Three cases appear:

- Symbol | is found. In this case, the machine starts a new computation of C from $(s_0, (0, 0))$, the configuration becomes of type 1 or 3, and its upper complexity is 0.
- Symbol | is not found. We are over a configuration of type 2. By Lemma 1, its upper complexity is 0.
- A symbol different from | and 1 is found. The time is "reversed", and the machine comes back to the initial cell. Then, we know from Jeandel (2014) that we can discard it.

We conclude that C halts on $(s_0, 0, 0)$ if and only if $S_{T'_C}$ has positive entropy. □

References

F. Blanchard and P. Tiseur. Some properties of cellular automata with equicontinuity points. *Annales de l'Institute Henri Poincaré*, 36(5):562–582, 2000.

V. Blondel and J. Delvenne. Quasi-periodic configurations and undecidable dynamics for tilings, infinite words and Turing machines. *Theoret. Comput. Sci.*, 319:127–143, 2004.

A. Brudno. Entropy and the complexity of the trajectories of a dynamical system. *Transactions of the Moscow Mathematical Society*, 44(2):127–151, 1983.

J. Cassaigne, N. Ollinger, and R. Torres. A Small Minimal Aperiodic Reversible Turing Machine (submitted). Apr. 2014. URL `http://hal.archives-ouvertes.fr/hal-00975244`.

R. Downey and D. Hirschfeldt. *Algorithmic randomness and complexity*. Springer, 2010.

A. Gajardo and P. Guillon. Zigzags in Turing machines. In E. Ablayev, F. anf Mayr, editor, *Computer Science Symposium in Russia (CSR 2010)*, volume 6072 of *LNCS*, pages 109–119, 2010.

A. Gajardo and J. Mazoyer. One head machines from a symbolic approach. *Theor. Comput. Sci.*, 370:34–47, 2007.

F. Hennie. One-tape, off-line Turing machine computations. *Information and Control*, 8:553–578, 1965.

E. Jeandel. Computability of the entropy of one-tape Turing machines. In E. Mayr and N. Portier, editors, *Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25, pages 421–432, 2014.

J. Kari and N. Ollinger. Periodicity and immortality in reversible computing. In E. Ochmanski and J. Tyszkiewicz, editors, *Mathematical Foundations of Computer Science (MFCS 2008)*, volume 5162 of *LNCS*, pages 419–430, 2008.

P. Kůrka. On topological dynamics of Turing machines. *Theoret. Comput. Sci.*, 174(1-2):203–216, 1997.

P. Kůrka. *Topological and Symbolic Dynamics*. Société Mathématique de France, Paris, France, 2003.

D. Lind and B. Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, New York, NY, USA, 1995. ISBN 0-521-55900-6.

C. Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(2):199, 1991.

K. Morita. Universality of a reversible two-counter machine. *Theor. Comput. Sci.*, 168(2):303–320, 1996.

P. Oprocha. On entropy and Turing machine with moving tape dynamical model. *Nonlinearity*, 19:2475–2487, 2006.

R. Torres, N. Ollinger, and A. Gajardo. Undecidability of the surjectivity of the subshift associated to a Turing machine. *LNCS*, (7581):44–56, 2013.