A variant of Turing machines with no control states and its connection to bounded temporal memory

Domenico Cantone and Salvatore Cristofaro

Dipartimento di Matematica e Informatica, Università di Catania
Viale Andrea Doria, 6, I-95125 Catania, Italy
{cantone,cristofaro}@dmi.unict.it

Abstract. We present a variant with no control states of the Turing machine model of computation in which, at each computation step, the operation to be performed next is determined by the symbol currently scanned and by a bounded-length suffix of the sequence of the operations already executed on the tape. We show that our variant is Turing complete, i.e., it can simulate any (standard) Turing machine. (In fact, we shall provide a *strong* simulation which replicates the same tape configurations assumed by the simulated Turing machine, without using any additional tape symbol.) As a consequence, we argue that in order to perform general computation tasks, Turing machines do not need to memorize in their control states events arbitrarily far back in the past.

Keywords: Stateless Turing machines, Turing completeness, bounded temporal memory.

1 Introduction

The notion of control states of a Turing machine (TM) is strictly related to that of *temporal memory*, i.e., the ability to remember events occurred in the past.[1] In fact, TMs can remember actions performed in the past (i.e., the operations executed on the tape and/or the symbols read off from it) by encoding the information related to such actions within their control states. Since TMs have only finitely many control states, for each TM there is a fixed bound on the amount of information that can be encoded within control states, and hence a bound on the number of past actions that it can remember. However, despite this quantitative limitation, there is conceptually no limit on how "old" the actions

V. Biló, A. Caruso (Eds.): ICTCS 2016, Proceedings of the 17th Italian Conference on Theoretical Computer Science, 73100 Lecce, Italy, September 7–9 2016, pp. 36–48 published in CEUR Workshop Proceedins Vol-1720 at http://ceur-ws.org/Vol-1720

[1] In this paper, by a Turing machine we mean a deterministic two-way single-tape Turing machine with the instructions represented by quadruples, as described in [2].

remembered by control states can be, i.e., remembered actions can be *arbitrarily old*. To clarify this point, consider for instance the case of a TM that recognizes the strings η, over a 2-symbol alphabet $\{a, b\}$, whose leftmost and rightmost symbols are different. When the machine reads the rightmost symbol of η, it needs only to remember which was the leftmost symbol. Thus, only two control states suffice for this purpose: one to remember that the leftmost symbol of η was 'a' and one to remember that it was, instead, 'b'. Notice, however, that the number of actions that the machine can perform between the reading of the leftmost and of the rightmost symbols of η can be arbitrarily large, depending on the length of η, and therefore, when the TM reads the rightmost symbol of η, the previous reading of the leftmost symbol turns out to be arbitrarily old.

The capability to remember subsequences of arbitrarily old actions is a peculiarity of TMs and, more generally, of any other device which uses control states (e.g., *finite automata*). The main aim of this paper is to investigate to what extent this property of TMs is demanded for performing general computations. To this end, we introduce a *stateless* variant of TMs (namely, with no control states or, equivalently, with only one control state), named *Stateless Bounded Temporal Memory Turing Machines* (SBTMs).[2] From a purely mechanical point of view, SBTMs behave identically to standard TMs. However, at each computation step, the operation to be performed next on the tape by the scanning head of a SBTM (i.e., printing a symbol onto the scanned cell, or moving to the right or to the left of the current position) is determined by a suffix of the sequence of the operations already executed (including the symbols read by the head after these operations and, therefore, also the symbol currently scanned),[3] whose length cannot exceed a bound depending solely on the particular machine.

We shall prove that SBTMs are Turing complete, i.e., they can simulate any standard TM. In fact, we shall see that for each TM $\mathfrak{M}$ one can construct a SBTM $\mathfrak{V}$ which *faithfully* simulates $\mathfrak{M}$, in the sense that (i) $\mathfrak{V}$ does not use any additional symbol other than those used by $\mathfrak{M}$, (ii) on any input η, $\mathfrak{V}$ halts if and only if so does $\mathfrak{M}$, and (iii) when, on input η, $\mathfrak{V}$ and $\mathfrak{M}$ halt, they reach the same tape configuration, while scanning the same cell. As a consequence, we shall also show that the above mentioned ability of TMs to remember within their control states subsequences of arbitrarily old actions is not strictly required for performing general computation tasks (see Section 3.1 for a more precise statement of this fact). The latter property entails in particular a bound on the amount of sequential data items needed to be memorized during computations, which is a relevant topic, e.g., in the field of Streaming Algorithms [5].

Some Turing complete variants of stateless TMs have already been proposed in the literature; however, these act quite differently from ours. This is the case, for instance, for the stateless TM studied in [1], named JTM. The head of a JTM

[2] Notice that restricting to one the number of control states in TMs results in a decreased computational power (see [6]).

[3] In fact, since SBTMs are devoid of the additional component of the set of control states, the only way for these devices to be aware if something happened in the past, is to trace back their own computations.

spans a window of three consecutive tape cells and can thus read/write blocks of three symbols in a single move. However, motion to the left/right of the current head position can occur only by one cell at a time. It turns out that JTMs are Turing complete. Stateless variants of several computational devices other than TMs have also been investigated in the literature, mainly *stateless automata* such as *stateless restarting automata* [3], *stateless multihead automata* [4], *stateless pushdown automata* [7], etc. However, these devices, even in their original form, have lower computational power than TMs.

The paper is organized as follows. Below we briefly review some notations and terminology which will be used in the rest of the paper. Then, in Section 2, we provide the definition of SBTMs and their semantics. Subsequently, in Section 3, we prove the Turing completeness of SBTMs, based on a simple encoding of the states of a TM, and also discuss other similar encodings. Then, in Section 3.1, we introduce a class of TMs with bounded temporal memory and prove its Turing completeness (based on the Turing completeness of SBTMs), thereby showing the independence of the computational power of TMs from the property of remembering subsequences of arbitrarily old actions within their control states. Finally, we draw our conclusions in Section 4.

Basic notations and definitions. Both in the case of TMs and SBTMs, we shall assume that *tape symbols* are drawn from the set of symbols $\mathscr{S} := \{\mathsf{s}_0, \mathsf{s}_1, \mathsf{s}_2, \ldots\}$, where s_0 is the blank. Often we shall write s_0 as $\square$. Additionally, we shall use the symbols in the set $\mathscr{A} := \{\curvearrowleft, \curvearrowright, \odot\}$ to denote certain active operations (as will be specified in the next section).

Throughout the paper, by a string we shall always mean a finite sequence of symbols belonging to the set $\mathscr{S} \cup \mathscr{A}$. In particular, for every $\mathscr{B} \subseteq \mathscr{S} \cup \mathscr{A}$, the set of all strings whose symbols belong to $\mathscr{B}$ will be denoted by $\mathscr{B}^*$. We shall write ε for the *empty string* and $|\alpha|$ for the *length* of the string α. Notice that we do *not* distinguish between a symbol and the string of length 1 consisting only of that symbol. The *concatenation* of two strings α and β is denoted by $\alpha.\beta$ or, more simply, by $\alpha\beta$. For every string α and each $n \geqslant 0$, α^n denotes the string of length $n|\alpha|$ consisting of the concatenation of n copies of α; thus, in particular, we put $\alpha^0 := \varepsilon$. A string α is a *suffix* (resp., *prefix*) of a string β, and in such a case we write $\alpha \sqsupseteq \beta$ (resp., $\alpha \sqsubseteq \beta$), if $\beta = \gamma\alpha$ (resp., $\beta = \alpha\gamma$), for some string γ; α is a *factor* of β, if $\beta = \lambda\alpha\rho$, for some strings λ and ρ.

Given a nonempty string α, we denote with $\mathsf{Head}(\alpha)$ (resp., $\mathsf{Last}(\alpha)$) the leftmost (resp., rightmost) symbol of α, and denote with $\mathsf{Init}(\alpha)$ (resp., $\mathsf{Tail}(\alpha)$) the string of length $|\alpha| - 1$ obtained by deleting the rightmost (resp., leftmost) symbol of α; we also put $\mathsf{Head}(\varepsilon) := \mathsf{Last}(\varepsilon) := \square$ and $\mathsf{Init}(\varepsilon) := \mathsf{Tail}(\varepsilon) := \varepsilon$.

2 SBTMs and their semantics

Informal description. A SBTM is equipped with the very same hardware components as a TM (see [2]), namely, a *linear tape*, infinite in both directions, divided into *cells*, with a *head* that at any given instant of time is positioned

over a particular cell (the *scanned cell*), and a finite *control box* that determines the operations that the head performs on the tape. As for TMs, such operations are of the following four types: (a) reading the symbol in the scanned cell, (b) printing a symbol onto the scanned cell, (c) moving to the left by one cell, and (d) moving to the right by one cell. However, only the operations of type (b), (c), and (d)–the *active operations*–are actually determined by the control box, since the *read* operation takes place automatically after each active operation and at the very beginning of each computation by a SBTM (see next).

Before a SBTM begins its computation with a string η as input, the symbols of η are placed (from left to right) in consecutive cells of the tape (the *input cells*), one symbol per cell, whereas the remaining cells of the tape are left "empty", i.e., they contain a special *blank* symbol; the head is positioned over (i.e., scans) the *initial cell*, namely, the cell immediately to the left of the input cell containing the first (i.e., leftmost) symbol of η. This is the *initial configuration* for a SBTM with input η. Then, starting from an initial configuration, SBTMs execute their computation steps at discrete instants of time, beginning at time $i = 0$. Each computation step, but the initial one, consists of the execution of an active operation, of one of the types (b), (c), or (d), immediately followed by a *read* operation from the cell being scanned by the head. In particular, for $i > 0$, the active operation O performed at $\boldsymbol{Step}_i$ is determined by the control box of the SBTM as a function of a suffix $\mathcal{S}$ of the sequence of the computation steps executed up to $\boldsymbol{Step}_{i-1}$ (included). In this case, we say that the SBTM has executed the *computation rule* (or *c-rule*) $\mathcal{S} \rightarrowtriangle O$. Concerning $\boldsymbol{Step}_0$, it is also convenient to regard it as consisting of the execution of an active operation followed by a *read* operation, as the remaining steps. Thus, we think of $\boldsymbol{Step}_0$ as consisting of the initial activation of the SBTM-control box–the zero operation–immediately followed by a read operation (of the blank symbol contained in the initial cell). After executing a computation step, a SBTM proceeds to the next step, and so on, until it possibly *halts*; this happens when none of the c-rules $\mathcal{S} \rightarrowtriangle O$ of the SBTM can be executed, for any suffix $\mathcal{S}$ of the sequence of computation steps executed up to then. Any finite sequence of consecutively executed computation steps constitutes a *(computation) trace* of the SBTM, whereas a *complete trace* is a trace starting at the initial computation $\boldsymbol{Step}_0$. The *computation history* of a SBTM $\mathfrak{V}$ is the whole sequence of configurations assumed by the tape of $\mathfrak{V}$ (namely, tape inscriptions along with head position) during the execution of the computation steps of $\mathfrak{V}$.

From the above description, it emerges that a SBTM is essentially a (finite) collection of c-rules. In particular, a *deterministic* SBTM is a *suffix-free* collection of c-rules, namely a collection of c-rules containing no two distinct c-rules $\mathcal{S}' \rightarrowtriangle O'$ and $\mathcal{S}'' \rightarrowtriangle O''$ such that $\mathcal{S}'$ is a suffix of $\mathcal{S}''$.[4] (Thus, intuitively, at any given computation step, the active operation O that the head of a deterministic SBTM can perform, if any, is uniquely determined by the previous computation steps.)

[4] In this paper we are interested only in deterministic SBTMs.

Formal definition. To formally define SBTMs, we need a convenient representation of c-rules and their components (i.e., traces and active operations). If we represent the zero operation with the symbol $\odot$, the left and right head motions with the symbols $\curvearrowleft$ and $\curvearrowright$, respectively, and, for $x \in \mathscr{S}$, we represent the operation of printing x onto the scanned cell with the very same symbol x, then every trace $\mathcal{T}$ can be handily represented as a string (in the alphabet $\mathscr{S} \cup \mathscr{A}$) as follows. First of all, we designate each computation step $\boldsymbol{S}$ by a 2-symbol string $\mathscr{o}x$, where $\mathscr{o} \in \mathscr{S} \cup \mathscr{A}$ is the symbol denoting the active operation O performed by $\boldsymbol{S}$ (as explained earlier) and x is the symbol subsequently read from the tape by step $\boldsymbol{S}$, immediately after the execution of O. Then a trace $\mathcal{T}$ can be represented with the string resulting from concatenating the 2-symbol strings representing the computation steps in $\mathcal{T}$, in the same order in which they occur in it. For instance, let $\mathcal{T}$ consist of the two computation steps $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, where: (i) $\boldsymbol{S}_1$ consists of moving the head one cell to the right and then reading the symbol $\Box$ from the tape; and (ii) $\boldsymbol{S}_2$ consists of printing s_1 onto the scanned cell and then reading the same symbol s_1. Then, $\mathcal{T}$ is represented by the string $\curvearrowright\Box\mathsf{s}_1\mathsf{s}_1$. Observe also that any complete trace of a SBTM, i.e., a trace starting at the initial $\boldsymbol{Step}_0$, is represented by a string with prefix $\odot\Box$. Finally, we represent a c-rule $\mathcal{S} \twoheadrightarrow O$ as the ordered pair $(\sigma, \mathscr{o})$, where σ is the string representing the trace $\mathcal{S}$ (as described above), and $\mathscr{o}$ is the symbol denoting the active operation O.

At this point, a SBTM could be formally defined simply as the set of the ordered pairs $(\sigma, \mathscr{o})$ representing its c-rules $\mathcal{S} \twoheadrightarrow O$. However, such an approach would require particular care to avoid circularity, since the notion of traces of SBTMs is defined in terms of the very same notion of c-rules which we want to define. For the sake of minimality, we shall circumvent this circularity problem by simply admitting, among the ordered pairs representing 'genuine' c-rules, even those pairs $(\sigma, \mathscr{o})$ in which the string σ could possibly represent no valid trace. Thus we give the following definitions.

Definition 1. *A* c-rule *is an ordered pair $(\sigma, \mathscr{o})$, also written as $\sigma \twoheadrightarrow \mathscr{o}$, where σ is any string in the alphabet $\mathscr{S} \cup \mathscr{A}$ and $\mathscr{o} \in \mathscr{S} \cup \{\curvearrowleft, \curvearrowright\}$. A set of c-rules is* suffix-free*, if it contains no two distinct c-rules $\sigma' \twoheadrightarrow \mathscr{o}'$ and $\sigma'' \twoheadrightarrow \mathscr{o}''$ such that $\sigma' \sqsupseteq \sigma''$.*

Definition 2. *A* (deterministic) SBTM *is any finite, suffix-free set of c-rules. The* alphabet of a SBTM $\mathfrak{V}$ *is the set $\mathscr{S}_{\mathfrak{V}}$ of all the tape symbols, but the blank $\Box$, occurring in any of its c-rules.*

2.1 Formal semantics of SBTMs

Let $\mathfrak{V}$ be a SBTM. Suppose that $\mathfrak{V}$ is ran with some given input string $\eta \in (\mathscr{S}_{\mathfrak{V}})^*$. For any time instant $i \geqslant 0$, we denote with $\mathcal{T}_i^{\mathfrak{V}}(\eta)$ the (string representing the) complete trace of $\mathfrak{V}$ from $\boldsymbol{Step}_0$ up to $\boldsymbol{Step}_i$, and with $\mathcal{C}_i^{\mathfrak{V}}(\eta)$ the tape configuration reached at the end of $\boldsymbol{Step}_i$. Letting C be the cell scanned by the head at the end of $\boldsymbol{Step}_i$, we represent $\mathcal{C}_i^{\mathfrak{V}}(\eta)$ as the triple (λ, s, ρ) in which:

(a) λ is the string consisting of the symbols contained, at the end of $\boldsymbol{Step}_i$, in the (possibly empty) portion of the tape to the left of C from the leftmost cell that has been scanned in any of the computation steps up to $\boldsymbol{Step}_i$; (b) s is the symbol contained in C at the end of $\boldsymbol{Step}_i$; and (c) ρ is the string consisting of the symbols contained, at the end of $\boldsymbol{Step}_i$, in the (possibly empty) portion of the tape to the right of C up to the rightmost cell that either has been scanned in any of the computation steps up to $\boldsymbol{Step}_i$ or is an input cell.

We say that a trace $\mathcal{T}_i^{\mathfrak{V}}(\eta)$ *is terminal (relative to* $\mathfrak{V}$*)*, if there is no c-rule $\sigma \twoheadrightarrow \mathscr{o}$ in $\mathfrak{V}$ such that σ is a suffix of $\mathcal{T}_i^{\mathfrak{V}}(\eta)$.

The formal definitions of $\mathcal{C}_i^{\mathfrak{V}}(\eta)$ and $\mathcal{T}_i^{\mathfrak{V}}(\eta)$, for $i \geqslant 0$, are provided recursively as follows. Initially, for $i = 0$, we put

$$\mathcal{C}_0^{\mathfrak{V}}(\eta) := (\varepsilon, \square, \eta) \quad \text{and} \quad \mathcal{T}_0^{\mathfrak{V}}(\eta) := \odot\square\,.$$

For $i > 0$, let $\mathcal{C}_{i-1}^{\mathfrak{V}}(\eta) := (\lambda, s, \rho)$. Then, if $\mathcal{T}_{i-1}^{\mathfrak{V}}(\eta)$ is terminal, we put $\mathcal{C}_i^{\mathfrak{V}}(\eta) := \mathcal{C}_{i-1}^{\mathfrak{V}}(\eta)$ and $\mathcal{T}_i^{\mathfrak{V}}(\eta) := \mathcal{T}_{i-1}^{\mathfrak{V}}(\eta)$. Otherwise, let $\sigma \twoheadrightarrow \mathscr{o}$ be *the* c-rule of $\mathfrak{V}$ such that σ is a suffix of $\mathcal{T}_{i-1}^{\mathfrak{V}}(\eta)$.[5] Then we put recursively:

$$\mathcal{C}_i^{\mathfrak{V}}(\eta) \;/\; \mathcal{T}_i^{\mathfrak{V}}(\eta) := \begin{cases} (\lambda, \mathscr{o}, \rho) \;/\; \mathcal{T}_{i-1}^{\mathfrak{V}}(\eta)\mathscr{o}\mathscr{o}, & \text{if } \mathscr{o} \in \mathscr{S} \\ (\mathsf{Init}(\lambda), \mathsf{Last}(\lambda), s\,.\,\rho) \;/\; \mathcal{T}_{i-1}^{\mathfrak{V}}(\eta)\curvearrowleft\mathsf{Last}(\lambda), & \text{if } \mathscr{o} = \curvearrowleft \\ (\lambda\,.\,s, \mathsf{Head}(\rho), \mathsf{Tail}(\rho)) \;/\; \mathcal{T}_{i-1}^{\mathfrak{V}}(\eta)\curvearrowright\mathsf{Head}(\rho), & \text{if } \mathscr{o} = \curvearrowright\,. \end{cases}$$

The sequence $\mathcal{C}_0^{\mathfrak{V}}(\eta), \mathcal{C}_1^{\mathfrak{V}}(\eta), \mathcal{C}_2^{\mathfrak{V}}(\eta), \ldots$ is the *computation history of* $\mathfrak{V}$ *with input* η. We say that $\mathfrak{V}$ *with input* η *halts, and produce as output a string* $\omega \in (\mathscr{S}_{\mathfrak{V}})^*$ (and write $\mathfrak{V}(\eta) \downarrow \omega$), if, for some $i \geqslant 0$, we have that $\mathcal{T}_i^{\mathfrak{V}}(\eta)$ is terminal and ω is the string obtained from $\lambda s \rho$ by deleting all occurrences of the symbol $\square$, where $(\lambda, s, \rho) := \mathcal{C}_i^{\mathfrak{V}}(\eta)$.

As in the case of TMs, SBTMs can be used in three different modalities: (a) to compute *partial functions*, (b) as *string generators*, and (c) as *language acceptors*. Specifically, given a SBTM $\mathfrak{V}$, we say that:

(a) $\mathfrak{V}$ *computes a (partial) string function* f over $(\mathscr{S}_{\mathfrak{V}})^*$, if, for all $\eta, \omega \in (\mathscr{S}_{\mathfrak{V}})^*$,

$$\mathfrak{V}(\eta) \downarrow \omega \quad \text{iff} \quad \omega = f(\eta)\,.$$

(b) $\mathfrak{V}$ *generates a string* $\omega \in (\mathscr{S}_{\mathfrak{V}})^*$, if $\mathfrak{V}(\varepsilon) \downarrow \omega$.

Finally, in order to use SBTMs as language acceptors, we must first define what we mean for a SBTM to *accept/reject* its input. Since SBTMs have no control states, a possibility could be the following one. We extend the definition of a SBTM by including two new distinguished symbols, say the symbols $\mathbb{y}$ (for "Yes") and $\mathbb{n}$ (for "No"), such that for no c-rule $\sigma \twoheadrightarrow \mathscr{o}$ in the SBTM it is the case that $\mathbb{y}$ or $\mathbb{n}$ occurs in σ (but possibly we can have that $\mathscr{o} = \mathbb{y}$ or $\mathscr{o} = \mathbb{n}$).[6] Then we say that

[5] Observe that, according to Definition 2 there is in fact exactly one such c-rule $\sigma \twoheadrightarrow \mathscr{o}$.

[6] Thus, if the SBTM prints $\mathbb{y}$ or $\mathbb{n}$ then it halts.

(c) a SBTM $\mathfrak{V}$ *accepts* (resp., *rejects*) *an input string* $\eta \in (\mathscr{S}_{\mathfrak{V}} \setminus \{\mathtt{y}, \mathtt{n}\})^*$, if there is a time instant $i \geqslant 0$ such that $\mathsf{Last}(\mathcal{T}_i^{\mathfrak{V}}(\eta)) = \mathtt{y}$ (resp., $\mathsf{Last}(\mathcal{T}_i^{\mathfrak{V}}(\eta)) = \mathtt{n}$).

Example 1. Let us consider the SBTM $\mathfrak{V}$, consisting of the following c-rules

(1) $\odot\Box \rightarrow \curvearrowright$ (2) $\odot\Box\curvearrowright\Box \rightarrow \mathtt{n}$ (3) $\odot\Box\curvearrowright s \rightarrow s$ (4) $ss\curvearrowright t \rightarrow s$
(5) $\curvearrowright stt \rightarrow \curvearrowright$ (6) $\curvearrowright uss\curvearrowright\Box \rightarrow \mathtt{y}$ (7) $\curvearrowright sss\curvearrowright\Box \rightarrow \mathtt{n}$,

where $s, t, u \in \{\mathtt{s}_1, \mathtt{s}_2\}$, with $s \neq u$. Then $\mathfrak{V}$ accepts exactly the nonempty strings in $\{\mathtt{s}_1, \mathtt{s}_2\}^*$ whose leftmost and rightmost symbols are different, while rejecting all remaining strings in $\{\mathtt{s}_1, \mathtt{s}_2\}^*$. The SBTM $\mathfrak{V}$ behaves as follows. Starting with an input string η on its tape, $\mathfrak{V}$ initially moves one cell to the right (cf. c-rule (1)) and checks whether the newly scanned cell C is empty; if C is empty (which means that η is the empty string), then $\mathfrak{V}$ prints the symbol $\mathtt{n}$ onto C (cf. c-rule (2)) and halts, thus rejecting η; otherwise, if C contains a symbol $s \in \{\mathtt{s}_1, \mathtt{s}_2\}$, then $\mathfrak{V}$ prints back s onto C (cf. c-rule (3)). Then, each time $\mathfrak{V}$ scans a nonempty cell D, i.e., a cell containing a symbol $t \in \{\mathtt{s}_1, \mathtt{s}_2\}$, it checks which of the following two conditions holds, namely: (i) the cell D has just been reached after a right head motion preceded by a printing action of a symbol $s \in \{\mathtt{s}_1, \mathtt{s}_2\}$; (ii) the cell D has just been involved in a printing action preceded by a right head motion. In case (i), $\mathfrak{V}$ prints the symbol s over D (cf. c-rule (4)), otherwise, in case (ii), $\mathfrak{V}$ moves one cell to the right of D (cf. c-rule (5)). Finally, when an empty cell E is encountered (following a right head motion), it is checked whether the two symbols s and u previously read from the two adjacent cells to the left of E are different. If this is the case, $\mathfrak{V}$ prints the symbol $\mathtt{y}$ onto E (cf. c-rule (6)) and halts, thus accepting η; otherwise, if s and u are equal, $\mathfrak{V}$ prints the symbol $\mathtt{n}$ (cf. c-rule (7)) and halts, thus rejecting η. Observe that, during the computation, the symbols of the input string η, but the leftmost one, are in turn replaced by the leftmost symbol of η; hence, at any step, the leftmost symbol of η is remembered by the last printed symbol. □

3 Turing completeness of SBTMs

We prove the Turing completeness of SBTMs by constructing for every TM $\mathfrak{M}$ a SBTM $\langle\mathfrak{M}\rangle$ such that $\mathfrak{M}$ and $\langle\mathfrak{M}\rangle$ are equivalent in the following strong sense, namely, for each input string η: (a) $\mathfrak{M}$ halts iff $\langle\mathfrak{M}\rangle$ halts; and (b) when $\mathfrak{M}$ and $\langle\mathfrak{M}\rangle$ halt, they do with the same tape configuration, i.e., with the same tape content and the same head position.[7] (In fact, we shall see informally that the computations of $\mathfrak{M}$ and $\langle\mathfrak{M}\rangle$ are synchronized in an even stronger way.)

To begin with, let us review some useful notations and concepts pertaining to TMs. We assume that the control states that any TM $\mathfrak{M}$ can assume belong to the set $\mathscr{Q} = \{\mathtt{q}_0, \mathtt{q}_1, \mathtt{q}_2, \ldots\}$, where $\mathtt{q}_0$ is bound to denote the *initial state* of $\mathfrak{M}$. Turing machine's instructions are represented as quadruples of the form $(q, x, \mathscr{o}, p)$, where

[7] In Section 3.1 we will see, as well, how to construct, for each SBTM $\mathfrak{V}$, an equivalent TM $\langle\!|\mathfrak{V}|\!\rangle$ which simulates $\mathfrak{V}$.

$q, p \in \mathscr{Q}$, $x \in \mathscr{S}$, and $\mathscr{o} \in \mathscr{S} \cup \{\curvearrowleft, \curvearrowright\}$, whose meaning is that when a TM is in state q while reading the symbol x, the head performs the active operation represented by $\mathscr{o}$, and then the TM enters state p (see [2]). A (deterministic) TM is then formally defined as a finite set of quadruples of the above type, no two of which begin with the same state-symbol pair (q, x), and such that at least one quadruple begins with q_0. For every TM $\mathfrak{M} := \{(q_i, x_i, \mathscr{o}_i, p_i) : 0 \leqslant i \leqslant n\}$, where $n \geqslant 0$, we put:

$$\mathscr{Q}_{\mathfrak{M}} := \{q_i, p_i : 0 \leqslant i \leqslant n\} \quad \text{and} \quad \mathscr{S}_{\mathfrak{M}} := \{\square\} \cup (\{x_i, \mathscr{o}_i : 0 \leqslant i \leqslant n\} \setminus \{\curvearrowleft, \curvearrowright\}) .$$

Thus, $\mathscr{Q}_{\mathfrak{M}}$ and $\mathscr{S}_{\mathfrak{M}}$ are the *set of states* and the *tape alphabet* of $\mathfrak{M}$, respectively. We say that a TM $\mathfrak{M}$ *activates a state-symbol pair* $(q, x) \in \mathscr{Q}_{\mathfrak{M}} \times \mathscr{S}_{\mathfrak{M}}$ when $\mathfrak{M}$ reads the symbol x from the tape while in state q.[8]

Let $\mathfrak{M}$ be a given TM and $\mathscr{S}_{\mathfrak{M}}$ its tape alphabet. Also, let m be the smallest *positive* index such that $\mathscr{S}_{\mathfrak{M}} \subseteq \{\mathsf{s}_0, \mathsf{s}_1, \ldots, \mathsf{s}_m\}$. For simplicity, we shall write s_m as s (hence, $\mathsf{s} \neq \square$). In addition, for $x \in \mathscr{S}$, let the 2-symbol string xx be denoted by $(\!|x|\!)$.[9] The SBTM $\langle\mathfrak{M}\rangle$ intended to simulate the TM $\mathfrak{M}$ will encode each state q_i of $\mathfrak{M}$ with the trace $(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)$, consisting of a sequence of $(i+4)$ printing steps which uniquely characterizes q_i. The simulation proceeds in such a way that when $\mathfrak{M}$ activates the state-symbol pair (q_i, x), (i) the string $(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)$ turns out to be a suffix of the current complete trace of $\langle\mathfrak{M}\rangle$ and, additionally, (ii) $\mathfrak{M}$ and $\langle\mathfrak{M}\rangle$ have the same tape configuration.

The first state-symbol pair activated by $\mathfrak{M}$ is $(\mathsf{q}_0, \square)$. Thus we put into $\langle\mathfrak{M}\rangle$ the following block $\mathfrak{S}_0$ of c-rules:

$$\odot\square \rightarrowtriangle \mathsf{s},\ \odot\square(\!|\mathsf{s}|\!) \rightarrowtriangle \square,\ \odot\square(\!|\mathsf{s}|\!)(\!|\square|\!) \rightarrowtriangle \square,\ \odot\square(\!|\mathsf{s}|\!)(\!|\square|\!)^2 \rightarrowtriangle \mathsf{s},\ \odot\square(\!|\mathsf{s}|\!)(\!|\square|\!)^2(\!|\mathsf{s}|\!) \rightarrowtriangle \square .$$

The c-rules in $\mathfrak{S}_0$ have the effect to generate the complete trace $\odot\square(\!|\mathsf{s}|\!)(\!|\square|\!)^2(\!|\mathsf{s}|\!)(\!|\square|\!)$ (while leaving the tape as in its initial configuration), whose suffix $(\!|\mathsf{s}|\!)(\!|\square|\!)^2(\!|\mathsf{s}|\!)(\!|\square|\!)$ correctly encodes the activation of the pair $(\mathsf{q}_0, \square)$. Notice that the block $\mathfrak{S}_0$ is independent of the specific TM $\mathfrak{M}$.

Next, for each instruction $I := (\mathsf{q}_i, x, \mathscr{o}, \mathsf{q}_j)$ in $\mathfrak{M}$ we define a corresponding simulating block $\langle I\rangle$ of c-rules for $\langle\mathfrak{M}\rangle$. We distinguish the following cases:

Case $\mathscr{o} = y$, with $y \in \mathscr{S}$: In this case $\langle I\rangle$ consists of the following c-rules:

$$(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!) \rightarrowtriangle \mathsf{s}\,, \qquad (\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)(\!|\mathsf{s}|\!)(\!|\square|\!)^k \rightarrowtriangle \square\,, \text{ for } 0 \leqslant k \leqslant j+1$$
$$(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)(\!|\mathsf{s}|\!)(\!|\square|\!)^{j+2} \rightarrowtriangle \mathsf{s}\,, \qquad (\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)(\!|\mathsf{s}|\!)(\!|\square|\!)^{j+2}(\!|\mathsf{s}|\!) \rightarrowtriangle y\,.$$

[*Comment*: Assuming recursively that the current complete trace $\mathcal{T}$ of the simulating computation of $\langle\mathfrak{M}\rangle$ has the suffix $(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)$ (corresponding to the

[8] Note that, since a TM initially scans the blank preceding the leftmost symbol of its input string (see [2]), the pair $(\mathsf{q}_0, \square)$ is always activated by every TM at the very beginning of each of its computations.

[9] Thus, $(\!|x|\!)$ represents the computation step consisting of printing the symbol x and then reading the same symbol x (just printed) from the tape.

activation of the pair (q_i, x)), on a tape configuration $\mathcal{C}$ in which the head scans a cell C with the symbol x, the above block of c-rules has the effect of: (i) appending the new suffix $(\!|\mathsf{s}|\!)(\!|\square|\!)^{j+2}(\!|\mathsf{s}|\!)(\!|y|\!)$ to $\mathcal{T}$; and (ii) returning a tape configuration $\mathcal{C}'$ in which the head scans the cell C, now containing the symbol y, and that otherwise is identical to $\mathcal{C}$. Notice that the suffix $(\!|\mathsf{s}|\!)(\!|\square|\!)^{j+2}(\!|\mathsf{s}|\!)(\!|y|\!)$ encodes the pair state-symbol (q_j, y), which is the next pair to be activated by $\mathfrak{M}$.]

Case $\mathscr{o} \in \{\curvearrowleft, \curvearrowright\}$: In this case $\langle I \rangle$ consists of the c-rule $(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!) \twoheadrightarrow \mathscr{o}$, plus the blocks $\langle I \rangle_z$, for each $z \in \mathscr{S}_{\mathfrak{M}}$, consisting of the following c-rules:

$$(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)\mathscr{o}z \twoheadrightarrow \mathsf{s}\,, \qquad (\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)\mathscr{o}z(\!|\mathsf{s}|\!)(\!|\square|\!)^{k} \twoheadrightarrow \square\,, \text{ for } 0 \leqslant k \leqslant j+1$$

$$(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)\mathscr{o}z(\!|\mathsf{s}|\!)(\!|\square|\!)^{j+2} \twoheadrightarrow \mathsf{s}\,, \qquad (\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)\mathscr{o}z(\!|\mathsf{s}|\!)(\!|\square|\!)^{j+2}(\!|\mathsf{s}|\!) \twoheadrightarrow z\,.$$

[*Comment*: For $z \in \mathscr{S}_{\mathfrak{M}}$, let us put $\langle I \rangle_z^+ := \langle I \rangle_z \cup \{(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!) \twoheadrightarrow \mathscr{o}\}$. For simplicity, let us suppose that $\mathscr{o} = \curvearrowleft$ (we can reason similarly in the case in which $\mathscr{o} = \curvearrowright$). Assuming recursively that the current complete trace $\mathcal{T}$ of the simulating computation of $\langle \mathfrak{M} \rangle$ has the suffix $(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)$ (corresponding to the activation of the pair (q_i, x)), on a tape configuration $\mathcal{C}$ in which the head scans a cell C and the cell L on the left of C contains the tape symbol $z \in \mathscr{S}_{\mathfrak{M}}$, the subblock $\langle I \rangle_z^+$ of $\langle I \rangle$ has the effect of: (i) appending to $\mathcal{T}$ the new suffix $\mathscr{o}z(\!|\mathsf{s}|\!)(\!|\square|\!)^{j+2}(\!|\mathsf{s}|\!)(\!|z|\!)$; and (ii) returning a new tape configuration $\mathcal{C}'$ in which the head scans the cell L and that otherwise is identical to $\mathcal{C}$. Notice that the suffix $(\!|\mathsf{s}|\!)(\!|\square|\!)^{j+2}(\!|\mathsf{s}|\!)(\!|z|\!)$ of the prolongated trace of $\langle \mathfrak{M} \rangle$ encodes the pair state-symbol (q_j, z), which is the next pair to be activated by $\mathfrak{M}$.]

Finally, we put:

$$\langle \mathfrak{M} \rangle := \mathfrak{S}_0 \cup \bigcup_{I \in \mathfrak{M}} \langle I \rangle\,,$$

completing the formal definition of $\langle \mathfrak{M} \rangle$.

It can easily be verified that the set of c-rules $\langle \mathfrak{M} \rangle$ just given is suffix-tree (and, therefore, correctly defines a SBTM according to Definition 2). Indeed, observe that, for each c-rule $\sigma \twoheadrightarrow \mathscr{o}$ in the set $\bigcup_{I \in \mathfrak{M}} \langle I \rangle$, the string σ starts with a prefix π of the form $(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)$, for some $i \geqslant 0$ and $x \in \mathscr{S}_{\mathfrak{M}}$, such that, for each c-rule $\sigma' \twoheadrightarrow \mathscr{o}'$ in $\bigcup_{I \in \mathfrak{M}} \langle I \rangle$: (i) π is not a factor of $\mathsf{Tail}(\sigma')$; and (ii) if $\pi \sqsubseteq \sigma'$ and $|\sigma| = |\sigma'|$, then $\sigma = \sigma'$ and $\mathscr{o} = \mathscr{o}'$. Also, observe that each c-rule $\sigma \twoheadrightarrow \mathscr{o}$ in $\mathfrak{S}_0$ is such that $\mathsf{Head}(\sigma) = \odot$, and that $\odot$ does not occur in any c-rule in $\bigcup_{I \in \mathfrak{M}} \langle I \rangle$. Notice also that, apart from the trivial case in which $\mathscr{S}_{\mathfrak{M}} = \{\square\}$ (in which case we have $\mathsf{s} = \mathsf{s}_1 \notin \mathscr{S}_{\mathfrak{M}}$), the c-rules of $\langle \mathfrak{M} \rangle$ use only tape symbols in the alphabet $\mathscr{S}_{\mathfrak{M}}$ of $\mathfrak{M}$. Finally, we observe that the above comments to the definitions of the blocks $\langle I \rangle$ of c-rules, for each instruction $I \in \mathfrak{M}$, could be easily translated into a formal proof of the fact that $\langle \mathfrak{M} \rangle$ correctly simulates $\mathfrak{M}$ in the strong sense described at the beginning of the section.

Complexity of various encodings. As discussed above, the main idea behind the construction of $\langle \mathfrak{M} \rangle$ is to simulate the activation by $\mathfrak{M}$ of each state-symbol pair (q_i, x) by means of the trace $(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)(\!|x|\!)$ of $\langle \mathfrak{M} \rangle$, where $(\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)$

encodes the state q_i. This approach can be generalized as follows. First of all, we associate to each state q_i of $\mathfrak{M}$ a suitable *codeword* $\mathsf{c}(\mathsf{q}_i)$ in the alphabet $\{(\!|\mathsf{s}|\!), (\!|\square|\!)\}$ (in which each of the two strings $(\!|\mathsf{s}|\!)$ and $(\!|\square|\!)$ is temporarily regarded as a single symbol) and then simulate the activation of any pair (q_i, x) by means of the trace $\mathsf{c}(\mathsf{q}_i)\,.\,(\!|x|\!)$. The case $\mathsf{c}(\mathsf{q}_i) := (\!|\mathsf{s}|\!)(\!|\square|\!)^{i+2}(\!|\mathsf{s}|\!)$ corresponds to the approach adopted in the proof sketched above. For each choice of the particular encoding c of the states of $\mathfrak{M}$, the definitions of the sets $\mathfrak{S}_0$ and $\langle I\rangle$, for $I \in \mathfrak{M}$, provided before, generalize straightforwardly, thus obtaining the corresponding sets of c-rules $\mathfrak{S}_0^{\mathsf{c}}$ and $\langle I\rangle^{\mathsf{c}}$, for $I \in \mathfrak{M}$ (details are omitted for brevity). Clearly, we are interested only in encodings c which are *admissible*, in the sense that the resulting set of c-rules

$$\langle\mathfrak{M}\rangle^{\mathsf{c}} := \mathfrak{S}_0^{\mathsf{c}} \cup \bigcup_{I \in \mathfrak{M}} \langle I\rangle^{\mathsf{c}}$$

is suffix-free. For instance, a family of admissible encodings is provided by the functions c_h, for $h \geqslant 2$, such that $\mathsf{c}_h(\mathsf{q}_i) := (\!|\mathsf{s}|\!)(\!|\square|\!)^{i+h}(\!|\mathsf{s}|\!)$. Again, for $h = 2$ we obtain the encoding c_2 adopted in our previous proof. If we assume that $\mathscr{Q}_{\mathfrak{M}} = \{\mathsf{q}_0, \mathsf{q}_1, \ldots, \mathsf{q}_{k-1}\}$, for some $k \geqslant 1$ (in which case we say that $\mathfrak{M}$ is *tight*), each of the encodings c_h generates a SBTM $\langle\mathfrak{M}\rangle^{\mathsf{c}_h}$ whose longest c-rule has linear length in the number k of the states of $\mathfrak{M}$. More in general, for each TM $\mathfrak{T}$ and SBTM $\mathfrak{V}$, let us define the *sizes* $\|\mathfrak{T}\|$ and $\|\mathfrak{V}\|$ by putting:

$$\|\mathfrak{T}\| := |\mathscr{Q}_{\mathfrak{T}}| \quad \text{and} \quad \|\mathfrak{V}\| := \max\{|\sigma| : \sigma \twoheadrightarrow \mathscr{o} \in \mathfrak{V}, \text{ for some } \mathscr{o} \in \mathscr{S} \cup \{\curvearrowleft, \curvearrowright\}\}$$

(where $|\mathscr{Q}_{\mathfrak{T}}|$ denotes the cardinality of $\mathscr{Q}_{\mathfrak{T}}$). Then, using the asymptotic notation Θ, we have that $\|\langle\mathfrak{M}\rangle^{\mathsf{c}_h}\| = \Theta(\|\mathfrak{M}\|)$, for each $h \geqslant 2$. This indeed follows from the facts that (**A**) for each encoding c, $\|\langle\mathfrak{M}\rangle^{\mathsf{c}}\| = \Theta(\max(\{|\mathsf{c}(q)| : q \in \mathscr{Q}_{\mathfrak{M}}\}))$, and (**B**) $|\mathsf{c}_h(\mathsf{q}_i)| = 2\cdot(2+i+h)$, for $h \geqslant 2$. Notice, however, that more "compact" admissible encodings c can be devised such that $\|\langle\mathfrak{M}\rangle^{\mathsf{c}}\| = \Theta(\log\|\mathfrak{M}\|)$. Indeed, let $\zeta_0, \zeta_1, \zeta_2, \ldots$ be the list in quasi-lexicographic order[10] of the nonempty strings ζ in the alphabet $\{(\!|\mathsf{s}|\!), (\!|\square|\!)\}$ such that ζ contains no two consecutive occurrences of $(\!|\mathsf{s}|\!)$, and consider the encoding f such that $\mathsf{f}(\mathsf{q}_i) := (\!|\mathsf{s}|\!)^2(\!|\square|\!)\zeta_i(\!|\square|\!)(\!|\mathsf{s}|\!)^2$, for each $i \geqslant 0$. By reasoning much as we did for $\langle\mathfrak{M}\rangle$, it can be shown that $\langle\mathfrak{M}\rangle^{\mathsf{f}}$ is suffix-free, i.e., f is admissible.[11] Concerning the size of $\langle\mathfrak{M}\rangle^{\mathsf{f}}$, we show next that it is logarithmic in the size of $\mathfrak{M}$, provided that $\mathfrak{M}$ is tight. To begin with, it can easily be verified that, for each $n \geqslant 0$, the number of strings of length n in the alphabet $\{(\!|\mathsf{s}|\!), (\!|\square|\!)\}$ containing no two consecutive $(\!|\mathsf{s}|\!)$'s is exactly F_{n+2}, where F_n is the nth *Fibonacci number*.[12] Exploiting the identity $\sum_{j=1}^{n} F_j = F_{n+2} - 1$

[10] Hence, the strings ζ_i's are firstly ordered by their length and then lexicographically, where w.l.o.g. we conventionally assume that $(\!|\square|\!) < (\!|\mathsf{s}|\!)$.

[11] Note that, if we remove from $\mathsf{f}(\mathsf{q}_i)$ the "symbol" $(\!|\square|\!)$ surrounding ζ_i, the resulting encoding $\mathsf{g}(\mathsf{q}_i) := (\!|\mathsf{s}|\!)^2\zeta_i(\!|\mathsf{s}|\!)^2$ turns out to be not admissible. For, consider the case in which the TM $\mathfrak{M}$ contains the instruction $I := (\mathsf{q}_0, \mathsf{s}, x, \mathsf{q}_3)$, where $x \in \mathscr{S}$. Then, since $\zeta_0 = (\!|\square|\!)$ and $\zeta_3 = (\!|\square|\!)(\!|\mathsf{s}|\!)$, we have $\mathsf{g}(\mathsf{q}_0) = (\!|\mathsf{s}|\!)^2(\!|\square|\!)(\!|\mathsf{s}|\!)^2$ and $\mathsf{g}(\mathsf{q}_3) = (\!|\mathsf{s}|\!)^2(\!|\square|\!)(\!|\mathsf{s}|\!)^3$, so that the set $\langle I\rangle^{\mathsf{g}}$ contains the c-rules $(\!|\mathsf{s}|\!)^2(\!|\square|\!)(\!|\mathsf{s}|\!)^3 \twoheadrightarrow \mathsf{s}$ and $(\!|\mathsf{s}|\!)^2(\!|\square|\!)(\!|\mathsf{s}|\!)^5(\!|\square|\!)(\!|\mathsf{s}|\!)^3 \twoheadrightarrow x$, plainly implying that $\langle\mathfrak{M}\rangle^{\mathsf{g}}$ is not suffix-free.

[12] Thus, $F_0 = 0$, $F_1 = 1$, and $F_{k+2} = F_k + F_{k+1}$, for $k \geqslant 0$.

(for $n \geqslant 0$), we have

$$F_{|\zeta_i|+3} - 3 = \sum_{j=1}^{|\zeta_i|-1} F_{j+2} \leqslant i < \sum_{j=1}^{|\zeta_i|} F_{j+2} = F_{|\zeta_i|+4} - 3$$

for each $i \geqslant 2$. Given that $F_n = \Theta(\phi^n)$ (in fact $\frac{F_n}{\phi^n} \to \frac{1}{\sqrt{5}}$), where $\phi := \frac{1+\sqrt{5}}{2} \approx 1.618$ is the *golden ratio*, the latter inequalities imply $|\zeta_i| = \Theta(\log_\phi i)$. Hence we have $|\mathsf{f}(\mathsf{q}_i)| = \Theta(\log_\phi i)$, which yields, by (**A**) above and by the tightness of $\mathfrak{M}$,

$$\|\langle\mathfrak{M}\rangle^{\mathsf{f}}\| = \Theta(\log_\phi \|\mathfrak{M}\|)\,,$$

proving that the size of $\langle\mathfrak{M}\rangle^{\mathsf{f}}$ is logarithmic in the size of $\mathfrak{M}$, when $\mathfrak{M}$ is tight.

3.1 Standard TMs with bounded temporal memory

The intuitive considerations of Section 1 concerning the independence of the computational power of TMs from the property of these devices to remember subsequences of arbitrarily old actions within their control states can be made more precise by using a notion of *trace of* TMs identical to that of SBTM's trace introduced earlier. More precisely, a trace of a TM $\mathfrak{M}$ is a sequence of consecutive computation steps of $\mathfrak{M}$, where a computation step consists in the execution of an active operation of the scanning head on the tape (i.e., printing a symbol or a left/right motion) followed by the subsequent reading of the symbol contained in the newly scanned cell, as in the case of SBTMs.

Next, suppose that a TM $\mathfrak{M}$ performs an active operation O on its tape, during a given computation step $\boldsymbol{S}$. Then, letting $\mathcal{T}$ be the *complete trace* of $\mathfrak{M}$ consisting of the whole sequence of computation steps preceding $\boldsymbol{S}$, it can be readily verified that O is *uniquely determined* by $\mathcal{T}$; i.e., there is a function, $\Upsilon_{\mathfrak{M}}$, which maps each complete trace $\mathcal{T}$ to the particular active operation $O := \Upsilon_{\mathfrak{M}}(\mathcal{T})$ to be performed next. Now, let us define the class $\boldsymbol{\Omega}$ of *TMs with bounded temporal memory*. Specifically, the class $\boldsymbol{\Omega}$ consists of all TMs $\mathfrak{M}$ for which there is a constant $\ell_{\mathfrak{M}}$ (depending on $\mathfrak{M}$) such that the (active) operations to be performed next on the tape can be determined only by suffixes of the complete traces of $\mathfrak{M}$ consisting of at most $\ell_{\mathfrak{M}}$ consecutive computation steps; i.e., more formally, the function $\Upsilon_{\mathfrak{M}}$ is such that $\Upsilon_{\mathfrak{M}}(\mathcal{T}') = \Upsilon_{\mathfrak{M}}(\mathcal{T}'')$, for any two complete traces $\mathcal{T}'$ and $\mathcal{T}''$ which share a common suffix of $\ell_{\mathfrak{M}}$ computation steps. Intuitively, TMs in the class $\boldsymbol{\Omega}$ do not care of (or *forget*) actions strictly older than $\ell_{\mathfrak{M}}$ computation steps. Then, the question arises whether the computational power of TMs decreases when we restrict to the class $\boldsymbol{\Omega}$ above. As will be outlined below, given any SBTM $\mathfrak{V}$, an equivalent TM $\langle\!|\mathfrak{V}|\!\rangle$ can be constructed such that $\langle\!|\mathfrak{V}|\!\rangle \in \boldsymbol{\Omega}$; therefore, since SBTMs are Turing complete, it follows that every TM $\mathfrak{M}$ is equivalent to some TM $\mathfrak{T} \in \boldsymbol{\Omega}$ (e.g., $\mathfrak{T} := \langle\!|\langle\mathfrak{M}\rangle|\!\rangle$). Hence the answer to the above question is that the class $\boldsymbol{\Omega}$ is, in fact, Turing complete.

The basic idea of the construction of the TM $\langle\!|\mathfrak{V}|\!\rangle$, for a given SBTM $\mathfrak{V}$, is to use the control states of $\langle\!|\mathfrak{V}|\!\rangle$ to store in turn the strings θ of length $2\ell - 1$, with

$\ell := \frac{\|\mathfrak{V}\|}{2}$, such that $\theta \,.\, x$ represents the last ℓ computation steps of the current (complete) trace of $\langle\!|\mathfrak{V}|\!\rangle$, where x is the symbol currently scanned by the head of $\langle\!|\mathfrak{V}|\!\rangle$. When $\langle\!|\mathfrak{V}|\!\rangle$ assumes a state q, while reading the symbol x from the tape, the particular operation O that $\langle\!|\mathfrak{V}|\!\rangle$ has to perform next is determined by the c-rule $\sigma \,.\, x \rightarrowtriangle \mathscr{o}$ of $\mathfrak{V}$ such that σ is a suffix of the string θ stored within q, i.e, O is the operation represented by $\mathscr{o}$. Formally, the construction of $\langle\!|\mathfrak{V}|\!\rangle$ goes as follows. Let $\theta_0, \theta_1, \theta_2, \ldots$ be any listing of the strings in the alphabet $\mathscr{B} := \mathscr{S}_{\mathfrak{V}} \cup \{\square, \odot, \curvearrowleft, \curvearrowright\}$, where $\theta_0 = \odot$, and let $\mathbb{Q}\colon \mathscr{B}^* \longrightarrow \{\mathsf{q}_0, \mathsf{q}_1, \mathsf{q}_2, \ldots\}$ be the function such that, for each $i \geqslant 0$: if $|\theta_i| < 2\ell$, then $\mathbb{Q}(\theta_i) = \mathsf{q}_i$; otherwise, if $|\theta_i| \geqslant 2\ell$, then $\mathbb{Q}(\theta_i) = \mathbb{Q}(\overline{\theta_i})$, where $\overline{\theta_i}$ is the suffix of θ_i of length $2\ell - 1$.[13] Then, we let $\langle\!|\mathfrak{V}|\!\rangle$ be the Turing machine whose instructions are all the quadruples $(\mathbb{Q}(\theta), x, \mathscr{o}, \mathbb{Q}(\theta x \mathscr{o}))$ such that $\sigma \,.\, x \rightarrowtriangle \mathscr{o} \in \mathfrak{V}$ and $\sigma \sqsupseteq \theta$, where $\theta \in \mathscr{B}^*$, $x \in \mathscr{S}_{\mathfrak{V}} \cup \{\square\}$, and $\mathscr{o} \in \mathscr{S}_{\mathfrak{V}} \cup \{\square, \curvearrowleft, \curvearrowright\}$. It can be verified that $\langle\!|\mathfrak{V}|\!\rangle$ is indeed a TM equivalent to $\mathfrak{V}$ and that $\langle\!|\mathfrak{V}|\!\rangle \in \boldsymbol{\Omega}$ (details are omitted for brevity).

4 Conclusions

We have presented a variant of the Turing machine model of computation with no control states, named SBTM. In each computation step, the operation performed by the head is determined by the symbol currently scanned and by a suffix of bounded length of the sequence of the computation steps previously executed. We have shown that SBTMs are Turing complete, namely they are computationally as powerful as standard Turing machines. In addition, based on the Turing completeness of SBTMs, we have also shown that the computational power of TMs is independent of their ability to remember subsequences of arbitrarily old actions within their control states.

We plan to investigate further computational properties of SBTMs and, in particular, properties related to the following notion of string complexity which naturally arises in this context: given a string η, the SBTM-*complexity of* η is defined as the size of a minimum sized SBTM $\mathfrak{V}$ such that: (i) the alphabet of $\mathfrak{V}$ consists precisely of the symbols occurring in η; and (ii) $\mathfrak{V}$ generates η.

Acknowledgments

This work has been partially supported by the FIR project COMPACT: "Computazione affidabile su testi firmati" (code D84C46) and by project PRISMA PON04a2_A, funded by the Italian Ministry of University and Research within the PON 2007-2013 "Smart cities and communities" framework.

References

1. Joshua J. Arulanandham: Unconventional "Stateless" Turing-Like Machines. In: Selim G. Akl, Cristian S. Calude, Michael J. Dinneen, Grzegorz Rozenberg, and

[13] Note that $\overline{\mathbb{Q}(\odot)} = \mathsf{q}_0$.

H. Todd Wareham, editors, Unconventional Computation, volume 4618 of *Lecture Notes in Computer Science*, pages 55–61. Springer Berlin Heidelberg, 2007.
2. Martin Davis, Ron Sigal, Elaine J. Weyuker: *Computability, complexity, and languages*. Second Edition, Academic Press, 1994.
3. M. Kutrib, H. Messerschmidt, O. Friedrich: On stateless deterministic restarting automata. *Acta Informatica*, volume 47, pages 391–412. Springer-Verlag, 2010.
4. O. H. Ibarra, J. Karhumäki, A. Okhotin: On Stateless Multihead Automata: Hierarchies and the Emptiness Problem. In: E. S. Laber, C. Bornstein, L. T. Nogueira, L. Faria, editors, *LATIN 2008: Theoretical Informatics*, volume 4957 of *Lecture Notes in Computer Science*, pages 94–105, Springer Berlin Heidelberg, 2008.
5. S. Muthukrishnan: Data Streams: Algorithms and Applications. Foundations and Trends in Theoretical Computer Science, volume 1, issue 2, pages 117–236, 2005.
6. Claude E. Shannon: A universal Turing machine with two internal states. Automata Studies, *Annals of Mathematics Studies*, volume 34, pages 157–165, 1956.
7. L. Valiant: Decision procedures for families of deterministic pushdown automata. PhD thesis, University of Warwick, 1973.