2-State 2-Symbol Turing Machines with Periodic Support Produce Regular Sets

Turlough Neary(✉)

Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zürich, Switzerland
tneary@ini.phys.ethz.ch

Abstract. We say that a Turing machine has periodic support if there is an infinitely repeated word to the left of the input and another infinitely repeated word to the right. In the search for the smallest universal Turing machines, machines that use periodic support have been significantly smaller than those for the standard model (i.e. machines with the usual blank tape on either side of the input). While generalising the model allows us to construct smaller universal machines it makes proving decidability results for the various state-symbol products that restrict program size more difficult. Here we show that given an arbitrary 2-state 2-symbol Turing machine and a configuration with periodic support the set of reachable configurations is regular. Unlike previous decidability results for 2-state 2-symbol machines, here we include in our consideration machines that do not reserve a transition rule for halting, which further adds to the difficulty of giving decidability results.

1 Introduction

The search for Turing machines with small states-symbol products has received attention for a number of different variations of the model [2,8,12,13]. The variant that has received the most attention is what we call here the standard model [1,5,6,11] (single-tape Turing machines with the usual blank symbol and a specially reserved halt instruction). As one might expect if we generalise the standard model or the type of encoding it may use we can give machines with smaller state-symbol products. One generalisation used in the literature is to allow periodic support (an infinitely repeated word to the left of the input and another infinitely repeated word to the right). Universal machines that use such a generalisation are call weakly universal. Watanabe gave a number of small semi-weakly universal machines [13] (a repeated word only appears on one side of the input). Later, Cook [2] gave small weakly universal machines that simulate

This work is supported by Swiss National Science Foundation grant numbers 200021-153295 and 200021-166231. The author thanks the anonymous reviewers for their careful reading of the paper and their helpful comments.

G. Pighizzini and C. Câmpeanu (Eds.): DCFS 2017, LNCS 10316, pp. 274–286, 2017.
DOI: 10.1007/978-3-319-60252-3_22

the cellular automaton Rule 110, and these were improved upon in [7] to give weakly universal machines for the state-symbol pairs of $(6, 2)$, $(3, 3)$ and $(2, 4)$.

For the standard model a number of authors have given lower bounds on the size of the smallest possible universal Turing machines by proving the halting problem decidable for machines with the following state-symbol pairs: $(2, 2)$ [4,9], $(3, 2)$ [10], $(2, 3)$ (claimed by Pavlotskaya [9]), $(1, n)$ [3] and $(n, 1)$ (trivial), where $n \geqslant 1$. Unfortunately, these results do not provide lower bounds relevant to the weak and semi-weak machines mentioned above, and while generalising the encoding can simplify the task of finding machines with smaller state-symbol products the task of giving decidability results becomes more difficult. In this work we give decidability results for 2-state 2-symbol machines with periodic support. Our main result is as follows:

Theorem 1. *Given an arbitrary 2-state, 2-symbol, single-tape Turing machine with periodic support, the set of reachable configurations from an arbitrary configuration is regular.*

This result implies the decidability of many questions for 2-state 2-symbol machines with periodic support such as (1) Will a computation halt? (2) Does word w appear on the tape during a computation? (3) Given a number x does a computation enter a repeating sequence of configurations of length x? The halting property in question 1 is the standard way of signaling the end of a computation in Turing machines. Each of the properties in (2) and (3) above have been used as a means to signal the end of a computation in universal machines [2,12]. The weakly universal machines in [2,7] use the appearance of a special word to simulate a halting Turing machine and so our results show that there exists no 2-state 2-symbol weakly universal machine that ends its computation in this way. Our proof does not assume that there is a transition rule reserved for halting which further adds to the difficulty when compared to previous decidability proofs for 2-state 2-symbol machines.

The layout of the paper is as follows: In Sect. 2 we explain notation and give some definitions including definitions for two types of computation, periodic and semi-periodic, and we then show that if a computation is periodic or semi-periodic the set of reachable configurations is regular. In Sect. 3 we reduce the number of cases to be considered leaving the cases only in Fig. 1 to be solved, and in Sect. 4 we solve the Fig. 1 cases to prove Theorem 1.

2 Turing Machines with Periodic Support

Definition 1. *A Turing machine with periodic support is a tuple* $M = (Q, \Sigma, f, l, r)$. *Here* $Q = \{q_1, \ldots q_{|Q|}\}$ *and* Σ *are the finite sets of states and tape symbols respectively, and* $l, r \in \Sigma^*$ *are the left and right blank words respectively, where* $|r| > 0$ *and* $|l| > 0$. *The transition function is of the form* $f : Q \times \Sigma \to \Sigma \times \{L, R\} \times Q$.

We write f as a list of transition rules. Each transition rule is a quintuple $(q_x, \sigma_1, \sigma_2, d, q_y)$, with initial state q_x, read symbol $\sigma_1 \in \Sigma$, write symbol $\sigma_2 \in \Sigma$, move direction $d \in \{L, R\}$ and next state q_y.

Definition 2. *A configuration c of a Turing machine with periodic support has the form $c = u\,\mathbf{q_x}\alpha\,v$ where q_x is the current state, the tape head is reading the symbol α, and the words $u, v \in \Sigma^*$.*

For a machine with periodic support, when the tape head moves to the right of v the word r is appended to the right of v and when the tape head moves to the left of u the word l is prepended to the left of u. When $l = r = 0$ we have the classic Turing machine with blank symbol 0. We write $c_1 \vdash c_2$ to denote that configuration c_2 is obtained via a single Turing machine computation step on c_1 and we write $c_1 \vdash^* c_t$ if there exists a sequence of 0 or more computations steps of the form $c_1 \vdash c_2 \vdash \ldots c_{t-1} \vdash c_t$. We say that c_t is *reachable* from c_1 and call $c_1 \vdash c_2 \vdash \ldots c_{t-1} \vdash c_t$ a computation sequence.

We label each Turing machine tape cell with an index. The cell with index i, which we will call cell i, has cell $i-1$ immediately to its left and cell $i+1$ immediately to its right. The position function $p(t)$ gives the index of the cell being read by the tape head at time t. The rightmost position up to time t is a cell index $P_r(t) = \max_{t' \leqslant t}(p(t'), p_r)$ and the leftmost position up to time t is a cell index $P_l(t) = \min_{t' \leqslant t}(p(t'), p_l)$, where p_l and p_r are respectively the leftmost and rightmost cells occupied by the input w.

Definition 3 (Periodic computation). *Let M be a Turing machine with periodic support. The computation of M is periodic if there is a sequence s of transition rules and a time t, such that sequence s is executed between times $t + i|s|$ and $t + (i+1)|s|$ for all $i \in \mathbb{N}$, where $|s|$ is the number of rules in s.*

Lemma 1. *If a computation is periodic, then $\{c_i | c_1 \vdash^* c_i\}$ is a regular set.*

Proof. After t time steps M has gone through a sequence of configurations $c_1 \vdash c_2 \vdash \ldots c_t$. There are two possible cases for the repeated sequence s, the sequence has either an equal or unequal number of left and right instructions. If there is an equal number of left and right instructions then after a further $|s|$ time steps we get $c_t, c_{t+1}, \ldots, c_{t+s}$ where c_t and c_{t+s} are identical. So in this case the set of configurations reachable from c_1 is the finite regular set $\{c_1, c_2, \ldots, c_{t+s-1}\}$.

If the number of left and right instructions in s are not equal, then we need only consider the case where there are more right than left instructions (as the case of more left than right instructions is symmetric). At time t we have a configuration $c_t = uu_1\,\mathbf{q_{x,1}}\alpha_1\,v_1$ such that the tape head visits the leftmost symbol in u_1 but does not revisit any symbol in u. Between times t and $t+s$ sequence s is executed giving the configuration sequence $c_t \vdash c_{t+1} \vdash \ldots c_{t+s}$ where $c_{t+i} = uu_i\,\mathbf{q_{x,i}}\alpha_i\,v_i$ for $0 \leqslant i < s$ and $c_{t+|s|} = uu'u_1\,\mathbf{q_{x,1}}\alpha_1\,v_1$. In configuration $c_{t+|s|}$ the words u_1 and v_1 must appear immediately to the left and right of the tape head position as the sequence of symbols read when executing s in the $|s|$ time steps following $c_{t+|s|}$ is identical to the sequence of symbols read in the $|s|$ time steps following c_t. Since the tape head will not visit

any cell to the left of the leftmost symbol in u_1 when executing s the word u' is not revisited by the tape head and is not altered for the remainder of the computation. For each subsequent execution of s another u' is placed on the tape and so on iteration $j+1$ of s we get a configuration sequence of the form $c_{t+|s|j} \vdash c_{t+|s|j+1} \vdash c_{t+|s|j+2} \vdash \ldots c_{t+|s|(j+1)}$ where $c_{t+|s|j+i} = u(u')^j u_i \, \mathbf{q_{x,i}} \alpha_i \, v_i$ and $0 \leqslant i < s$. So the set of configurations reachable from c_1 is the regular set $\{c_1, c_2, \ldots, c_t\} \cup u_i(u')^* u_i \, \mathbf{q_{x,i}} \alpha_i \, v_i$ where $0 \leqslant i < s$. □

Lemma 2. *Let M be a 2-state, 2-symbol Turing machine with periodic support. If, when M is started on a configuration c_1, there exists an $m \in \mathbb{N}$ such that on more than $|r|2^{m+1}$ occasions the tape head of M is over a cell $P_r(t)$ and does not visit cell $P_r(t) - m$ after time t, then the set $\{c_i | c_1 \vdash^* c_i\}$ is regular.*

Proof. Let $uu' \, \mathbf{q_x} \alpha v_r$ be a configuration at time t with $|u'| = m$ and the tape head over cell $P_r(t)$. If no cell to the left of $P_r(t) - m$ is visited after time t, then the future computation depends only on $u' \, \mathbf{q_x} \alpha v_r$. By definition of $P_r(t)$ no cell to the right of the tape head's location at symbol α has yet been read and so v_r must be a suffix of r. As a result, there are only $|r|2^{m+1}$ possible values for $u' \, \mathbf{q_x} \alpha v_r$. So, after $|r|2^{m+1}$ times where the tape head is over cell $P_r(t)$ and no longer visits cell $P_r(t) - m$, there are two times t_j and t_k that have the same value for $u' \, \mathbf{q_x} \alpha v_r$. Since the future computation depends only on $u' \, \mathbf{q_x} \alpha v_r$, the sequence of transition rules executed between times t_j and t_k are repeated ad infinitum. So from Lemma 1 the set $\{c_i | c_1 \vdash^* c_i\}$ is regular. □

Using a similar argument to the one used in Lemma 2 we get Comment 1.

Comment 1. *Let M be a 2-state, 2-symbol Turing machine with periodic support. If, when M is started on a configuration c_1, there exists an $m \in \mathbb{N}$ such that on more than $|l|2^{m+1}$ occasions the tape head of M is over a cell $P_l(t)$ and does not visit cell $P_l(t) + m$ after time t, then the set $\{c_i | c_1 \vdash^* c_i\}$ is regular.*

Comment 2. *Let M be a Turing machine with periodic support. If M started on configuration c does not make at least 2 consecutive right moves an inifinite number of times and at least 2 consecutive left moves an infinite number of times, then it either enters a loop or gives a computation of the type describe by Lemma 1 or Comment 1, and so the set of reachable configurations is regular.*

Definition 4 (Semi-periodic computation). *Let M be a Turing machine with periodic support. The computation of M is semi-periodic if there are $2x+2$ sequences of transition rules, $s_r, s_l, e_1, e_2 \ldots e_x, h_1, h_2 \ldots h_x$, such that there is a time z where the sequence*

$$S = (s_r)^{i+1} e_1 (s_l)^{i+1} h_1 (s_r)^{i+1} e_2 (s_l)^{i+1} h_2 \ldots (s_r)^{i+1} e_x (s_l)^{i+1} h_x$$

is executed between times $t(i)$ and $t(i+1)$ for all $i \in \mathbb{N}$, where $t(i) = z + \frac{(i^2+i)x}{2}(|s_r| + |s_l|) + i(|e_1 h_1 e_2 h_2 \ldots e_x h_x|)$, $f(s_r) = m$, $f(s_l) = -m$, $0 \leqslant f(e_i) < m$, $-m < f(h_i) \leqslant 0$, and $-m < f(S) \leqslant 0$ where $f(s) =$ (number of right instructions in sequence s) − (number of left instructions in sequence s). Also, for every instruction sequence y that is a prefix of s_l we have $f(y) \geqslant -m$, and for every instruction sequence y that is a prefix of s_r we have $f(y) \leqslant m$.

Lemma 3. *If a computation is semi-periodic, then* $\{c_i | c_1 \vdash^* c_i\}$ *is a regular set.*

Proof. At time $t(i)$ the sequence S is about to be executed by M for the $(i+1)^{\text{th}}$ time. The configuration at time $t(i)$ is given below on the left side of (1). From S in Definition 4 the execution begins with $(s_r)^{i+1}$, and so between times $t(i)+|s_r|k$ and $t(i)+|s_r|(k+1)$ we execute the sequence s_r, for $0 \leqslant k \leqslant i$. In Eq. (1) during the execution of each s_r the tape head does not move to the left of the word $u_{r,1}$ or to the right of the word $v_{r,1}$, and so since s_r has more right move than left move instructions we can use an argument similar to the one used in Lemma 1 to show that after k iterations of s_r ($|s^r|k$ time steps) the configuration on the right side of Eq. (1) is produced. Continuing on, the left side of Eq. (2) gives the configuration after i iterations of s_r, and one further iteration of s_r gives the configuration on the right. In the configuration on the right of Eq. (2) the word $v_{e_1,1} = v_{r',1}v_{e'_1}$, where $v_{r',1}$ is the length $|v_{r,1}|-m$ word that appears immediately to the right of the tape head after the last s_r in $(s_r)^{i+1}$ has executed (recall that each s_r shifts the tape head m cells to the right in the word $v_{r,1}$). The leftmost $(s_r)^{i+1}$ instruction sequence in S has now completed and so e_1, the next sequence in S, executes.

In Eq. (3) we show the $|e_1|$ times steps that complete the execution of e_1. Before we proceed we explain the presence of the word u'_1 (the length $f(e_1)$ prefix of u') that appears in the configuration on the right of Eq. (3). Immediately following the execution of e_1 we execute the sequence $(s_l)^{i+1}$, and as usual we give words $u_{l,1}$ and $v_{l,1}$ such that during the execution of each s_l the tape head does not move to the left of $u_{l,1}$ or to the right of $v_{l,1}$. From Definition 4 we know that executing e_1 shifts the tape head $f(e_1)$ cells to the right. So from Eq. (3) after the execution of e_1 (before the first s_l is about to execute) the tape head is at least $f(e_1)$ cells to the right of the rightmost u' subword in $(u')^{i+1}$. For every prefix y of s_l we have $f(y) \geqslant -m$ and so when executing s_l the tape head does not visit cells more than m positions to the left of its initial location when it began s_l. So since $|u'| = m$ and the initial tape head location is at least $f(e_1)$ cells to the right of the rightmost u', the leftmost $f(e_1)$ symbols in the rightmost u' (i.e. the prefix word u'_1) are not entered when executing the first s_l and so we have u'_1 to the left of $u_{l,1}$ in Eq. (3). Note that in Eq. (3) permitting the tape head to enter cells to the left of $u_{e_1,1}$ when executing e_1 will not effect the value of u'_1, as it is only possible to iterate s_l if u'_1 is a prefix of u' (see Eq. (4)) and so the prefix value of u'_1 is implied by the fact that $(s_l)^{i+1}$ is executed immediately after e_1.

$$u_{h'_1} u_{r,1}\, \mathbf{q_{r,1}} \alpha_{r,1}\, v_{r,1} (v')^i v_{e'_1} \quad \vdash^{|s_r|k} \quad u_{h'_1} (u')^k u_{r,1}\, \mathbf{q_{r,1}} \alpha_{r,1}\, v_{r,1} (v')^{i-k} v_{e'_1} \tag{1}$$

$$u_{h'_1} (u')^i u_{r,1}\, \mathbf{q_{r,1}} \alpha_{r,1}\, v_{r,1} v_{e'_1} \quad \vdash^{|s_r|} \quad u_{h'_1} (u')^{i+1} u_{e_1,1}\, \mathbf{q_{e_1,1}} \alpha_{e_1,1} v_{e_1,1} \tag{2}$$

$$u_{h'_1} (u')^{i+1} u_{e_1,1}\, \mathbf{q_{e_1,1}} \alpha_{e_1,1} v_{e_1,1} \quad \vdash^{|e_1|} \quad u_{h'_1} (u')^i u'_1 u_{l,1}\, \mathbf{q_{l,1}} \alpha_{l,1} v_{l,1} v_{e'_2} \tag{3}$$

$$u_{h'_1} (u')^i u'_1 u_{l,1}\, \mathbf{q_{l,1}} \alpha_{l,1} v_{l,1} v_{e'_2} \quad \vdash^{|s_l|k} \quad u_{h'_1} (u')^{i-k} u'_1 u_{l,1}\, \mathbf{q_{l,1}} \alpha_{l,1} v_{l,1} (v')^k v_{e'_2} \tag{4}$$

$$u_{h_1,1}\, \mathbf{q_{h_1,1}} \alpha_{h_1,1} v_{h_1,1} (v')^{i+1} v_{e'_2} \quad \vdash^{|h_1|} \quad u_{h'_2} u_{r,1}\, \mathbf{q_{r,1}} \alpha_{r,1} v_{r,1} v'_1 (v')^i v_{e'_2} \tag{5}$$

$$u_{h'_n} u_{r,1}\, \mathbf{q_{r,1}} \alpha_{r,1} v_{r,1} v'_n (v')^i v_{e'_n} \quad \vdash^{|s_r|k} \quad u_{h'_n} (u')^k u_{r,1}\, \mathbf{q_{r,1}} \alpha_{r,1} v_{r,1} v'_n (v')^{i-k} v_{e'_n} \tag{6}$$

The execution of sequence $(s_l)^{i+1}$ as shown in Eq. (4) proceeds in a manner similar to that of $(s_r)^{i+1}$. The main difference is that s_l has more left move than right move instructions and so each successive s_l sequence begins m cells further to the left. Following the execution of $(s_l)^{i+1}$, the sequence h_1 is executed as

shown in Eq. (5). In the configuration on the left of Eq. (5) the word $u_{h_1,1} = u_{h'_1}u'_1u_{l'_1}$, where $u_{l'_1}$ is the length $|u_{l,1}| - m$ word that appears immediately to the left of the tape head after the last s_l in $(s_l)^{i+1}$ has executed (recall each s_l shifts the tape head m cells to the left in the word $u_{l,1}$). The word v'_1 that appears in the configuration on the right of Eq. (5) is the length $f(h_1)$ suffix of v' and its presence can be explained in a manner similar to that of u'_1 in the previous paragraph. Continuing with the execution of sequence S, on the right side of Eq. (5) following the execution of h_1 the configuration is ready to allow the next $(s_r)^{i+1}$ sequence to execute. Note that unlike the configuration on the left of Eq. (1), the configuration on the right of Eq. (5) has the word v'_1 to the left of $(v')^i$. This implies that $(v')^i$ is a prefix of $v'_1(v')^i$ as to execute $(s_r)^{i+1}$ we must have the word $(v')^i$ to the right of $v_{r_1,1}$. This prefix property holds for each v'_j word produced by executing a h_j sequence. Similarly all words of the form $(u')^iu'_j$ share a suffix that allows $(s_r)^{i+1}$ to execute, where u'_j is a word produced by executing an e_j sequence (see Eq. (3)). Returning to Eq. (5), at the right end of the configuration $v_{e'_2}$ is of the correct form to allow e_2 to execute at the end of the $(s_r)^{i+1}$ scan right, and following this we once again scan left with $(s_l)^{i+1}$ where $u_{h'_2}$ allows the execution of h_2. So the process of scanning right with $(s_r)^{i+1}$ and left with $(s_l)^{i+1}$ is repeated with the n^{th} scan right given in Eq. (6). This continues until the sequence S is completed for the value i, whereupon S begins for $i+1$. The configurations for the n^{th} iteration of S can be obtained from the configurations for the first iteration of S simply by increasing the number of u' and/or the number of v' words in each configuration. So the set of configurations reachable by M is the union of the finite set of configurations before the first iteration of S and the configurations indicated in Eqs. (1) to (6). □

3 Reducing the Number of Cases Through Symmetries

There are 4096 possible 2-state 2-symbol machines and in this section we show how to reduce the number of cases to the 378 machines given in Fig. 1. Eq. (7) below gives the form of an arbitrary 2-state 2-symbol Turing machine where $\sigma_i \in \{0,1\}$, $d_j \in \{L,R\}$ and $q_k \in \{q_a, q_b\}$. In the sequel we denote each possible machine as a triple (Σ, D, Q) where $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, $D = \{d_1, d_2, d_3, d_4\}$, and $Q = \{q_1, q_2, q_3, q_4\}$. Each Σ, D and Q has 16 possible values and here we show how to reduce the number of cases under consideration to 6, 7, and 9, respectively (see Tables 1 to 3). We do this by identifying symmetries using the notion of regular equivalent machines which we define below, and we also solve some of the simplest cases from Σ, D and Q.

$$q_a, 0, \sigma_1, d_1, q_1 \qquad q_a, 1, \sigma_2, d_2, q_2 \qquad q_b, 0, \sigma_3, d_3, q_3 \qquad q_b, 1, \sigma_4, d_4, q_4 \tag{7}$$

Before we proceed we give some preliminary definitions and notation. Given a set with two elements $\{y, z\}$ we define $\overline{y} = z$ and $\overline{z} = y$. Given $w = w_0w_1 \dots w_m \in \{0,1\}^*$, we write $\overline{w} = \overline{w}_0\overline{w}_1 \dots \overline{w}_m$ to denote the word obtained by flipping each

bit $w_i \in \{0,1\}$ in w. We define three functions g_{reverse}, $g_{\text{b-flip}}$ and $g_{\text{s-flip}}$ each of which map a configuration $c = u\,\mathbf{q_x}\alpha\,v$ to another configuration as follows

$$g_{\text{reverse}}(c) = \underline{v}\,\mathbf{q_x}\alpha\,\underline{u} \qquad g_{\text{b-flip}}(c) = \overline{u}\,\mathbf{q_x}\overline{\alpha}\,\overline{v} \qquad g_{\text{s-flip}}(c) = u\,\overline{\mathbf{q_x}}\alpha\,v$$

where if $w = w_0 w_1 \dots w_m$ then $\underline{w} = w_m w_{m-1} \dots w_0$.

Definition 5. *Given Turing machines M and M' with periodic support, we say that M and M' are regular equivalent if for every computation sequence $c_1 \vdash c_2 \vdash \dots\ c_t$ of M there is a computation sequence $g(c_1) \vdash g(c_2) \vdash \dots\ g(c_t)$ of M' and vice versa, where $g \in \{g_{reverse}, g_{b\text{-}flip}, g_{s\text{-}flip}\}$.*

Showing one direction of the equivalence is sufficient to prove that a pair of machines are regular equivalent as each function from g is its own inverse. Comment 3 follows from Definition 5 as regularity is closed under g, and used with Lemma 4 reduces the number of cases for Theorem 1.

Comment 3. *Let M be a Turing machine with periodic support such that the set of configurations reachable from any arbitrary configuraiton is regular. Then if Turing machines M and M' are regular equivalent, for every configuration of M' with periodic support, the set of configurations reachable for M' is regular.*

Lemma 4. *Let M be an arbitrary 2-state, 2-symbol Turing machine with periodic support. Then applying any one of the mappings in Eqs. (8) to (10) to all transition rules in M gives a regular equivalent Turing machine M'.*

$$f_{d\text{-}flip}(q_x, \sigma, \sigma', D, q_y) \to (q_x, \sigma, \sigma', \overline{D}, q_y) \tag{8}$$
$$f_{b\text{-}flip}(q_x, \sigma, \sigma', D, q_y) \to (q_x, \overline{\sigma}, \overline{\sigma'}, D, q_y) \tag{9}$$
$$f_{s\text{-}flip}(q_x, \sigma, \sigma', D, q_y) \to (\overline{q_x}, \sigma, \sigma', D, \overline{q_y}) \tag{10}$$

Proof. To prove this lemma we show that for each mapping on M given by Eqs. (8) to (10) there is a $g \in \{g_{\text{reverse}}, g_{\text{b-flip}}, g_{\text{s-flip}}\}$, such that for each computation sequence $c_1 \vdash c_2 \vdash \dots\ c_t$ of M there is a computation sequence $g(c_1) \vdash g(c_2) \vdash \dots\ g(c_t)$ of M' satisfying Definition 5 (recall that it is sufficient to prove only one direction of the equivalence). For each of the translations in Eqs. (8), (9) and (10) we set g to g_{reverse}, $g_{\text{b-flip}}$, or $g_{\text{s-flip}}$, respectively. For example, if we apply the translation in Eq. (8) to M, then for each computation sequence $c_1 \vdash c_2 \vdash \dots\ c_t$ of M there is a computation sequence $g_{\text{reverse}}(c_1) \vdash g_{\text{reverse}}(c_2) \vdash \dots\ g_{\text{reverse}}(c_t)$ of M'. Each of the three cases (i.e. applying Eqs. (8), (9) or (10)) can easily be verified using an inductive argument showing that if M gives $c_i \vdash c_{i+1}$ then M' gives $g(c_i) \vdash g(c_{i+1})$. □

It is a straightforward matter to apply the mappings in Lemma 4 to the cases in Tables 1 to 3 to show that every case not given in the tables is regular equivalent to a case that is in the tables. Applying the mapping in (9) to machines for Σ_1, Σ_2, Σ_3 and Σ_4 gives machines for the Σ cases $(1,1,1,1)$, $(1,1,0,1)$,

$(1,1,1,0)$ and $(1,1,0,0)$ respectively[1]. Applying the mapping in (10) to Σ_2, Σ_3 and Σ_5 gives the Σ cases $(0,1,0,0)$, $(1,0,0,0)$, and $(1,0,0,1)$ respectively, and applying the mapping (9) and then (10) to Σ_2 and Σ_3 gives cases $(0,1,1,1)$ and $(1,0,1,1)$ respectively. We have now shown that 9 of the 10 possible Σ values not given in Table 1 are regular equivalent to cases in the table, and now the only case not covered is $(0,1,0,1)$. For $\Sigma = (0,1,0,1)$ the read symbol is the same as the write symbol for each transition rule and so it never changes the tape contents. It follows that if machines for the case $(0,1,0,1)$ enter the same cell more than twice they enter a loop and so computations for this case either loop or scan in one direction only never making 2 consecutive moves in the opposite direction and are thus covered by Comment 2.

Table 1. The 16 possible cases for the 4 read symbols in (7) reduced to 6 cases.

	Σ_1	Σ_2	Σ_3	Σ_4	Σ_5	Σ_6
$\sigma_1 =$	0	0	0	0	0	1
$\sigma_2 =$	0	0	0	0	1	0
$\sigma_3 =$	0	0	1	1	1	1
$\sigma_4 =$	0	1	0	1	0	0

Table 2. The 16 possible cases for the 4 move values in (7) reduced to 7 cases.

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$d_1 =$	L	L	L	R	L	L	L
$d_2 =$	L	L	R	L	L	R	R
$d_3 =$	L	R	L	L	R	L	R
$d_4 =$	R	L	L	L	R	R	L

Table 3. The 16 possible cases for the 4 next state values in (7) reduced to 9 cases.

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
$q_1 =$	q_a	q_b	q_a	q_a	q_b	q_b	q_b	q_b	q_b
$q_2 =$	q_b	q_a	q_b	q_b	q_a	q_a	q_b	q_b	q_b
$q_3 =$	q_a	q_a	q_a	q_b	q_a	q_b	q_a	q_a	q_b
$q_4 =$	q_a	q_a	q_b	q_a	q_b	q_a	q_a	q_b	q_a

Applying the mapping in (8) to cases D_1 to D_7 in Table 2 gives 7 regular equivalent cases. The remaining cases, (R,R,R,R) and (L,L,L,L), need not be included in Table 2 as they are covered by Comment 2.

The 16 possible cases for Q are reduced to the 9 given in Table 3 by omitting the 7 cases where $q_1 = q_2 = q_a$ or $q_3 = q_4 = q_b$. These 7 cases are not included in Table 3 as the behaviour for these cases is easily explained. To see this note that when we have $q_1 = q_2 = q_a$ state q_a is a trap state where if we enter q_a we never again enter state q_b and so from that point on the computation is essentially that of a 1-state 2-symbol Turing machine. Taking the case $q_1 = q_2 = q_a$, if the pair of transition rules for q_a both have the same write symbol (i.e. $\sigma_1 = \sigma_2$)

[1] Note that applying mapping (9) to machines of the form given by Eq. (7) also flips the read symbols and so applying it to $\Sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ gives $(\overline{\sigma_2}, \overline{\sigma_1}, \overline{\sigma_4}, \overline{\sigma_3})$ instead of $(\overline{\sigma_1}, \overline{\sigma_2}, \overline{\sigma_3}, \overline{\sigma_4})$.

or shift direction (i.e. $d_1 = d_2$), or if the write symbol is the same as the read symbol for both rules (i.e. $\sigma_1 = 0, \sigma_2 = 1$) then when the machine enters q_a the computation is periodic and from Lemma 1 this means that the set of reachable configurations is regular. For the remaining cases of $q_1 = q_2 = q_a$ we need only consider machines with the rules $q_a, 0, 1, R, q_a$ and $q_a, 1, 0, L, q_a$ as such machines are regular equivalent to all remaining cases for $q_1 = q_2 = q_a$. When in state q_a this machine scans right changing 0's to 1's until it reads a 1, and then it scans left changing 1's to 0's until it reads a 0, and this left scan right scan behaviour is repeated ad infinitum giving a semi-periodic computation. So from Lemma 3 the set of reachable configurations for machines with $q_1 = q_2 = q_a$ is regular. The same argument is applicable to the case $q_3 = q_4 = q_b$.

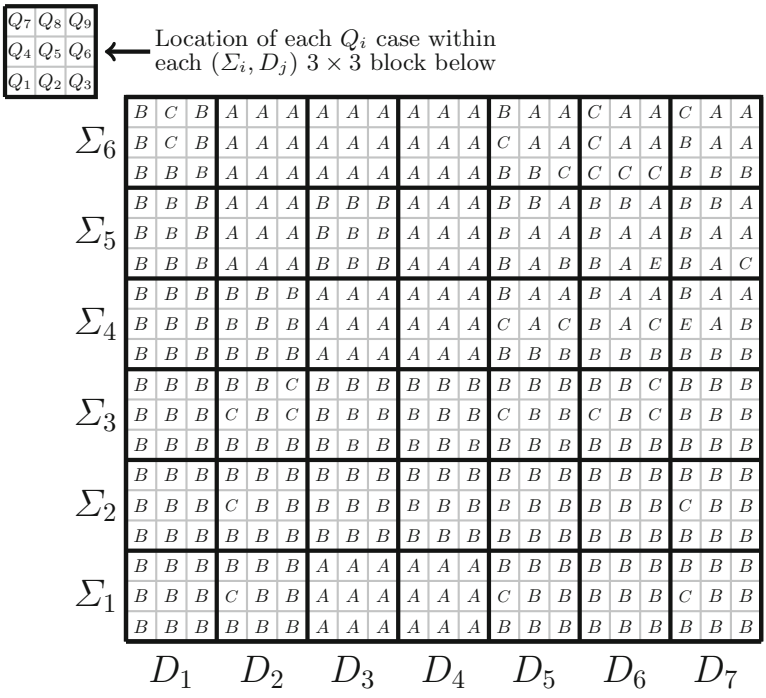

Fig. 1. An overview of the 378 cases given by Tables 1 to 3, which were obtained from the reduction in the number of cases given in Sect. 3. Each small square in the above figure represents a Turing machine. There is a 3×3 block of small squares for each (Σ_i, D_j) pair which gives the 9 possible values for Q_i. Letter A indicates a reduction to a symmetric case, B is for machines that have periodic computations, C is for machines that give semi-periodic computations, and E is for machines that are binary counters.

4 Solving Cases in Fig. 1

4.1 Case A: Reduce the Number of Cases Using Symmetries

From Comment 3 we know that if a pair of machines M and M' are regular equivalent we need only consider one of the two machines in our list of open cases. Table 4 shows that if the square in Fig. 1 for a machine (Σ_i, D_j, Q_k) contains an A then it is regular equivalent to another machine in the figure whose square does not contain an A. This means that we need not consider Case A machines because by solving the remaining cases each machine in Case A will be regular equivalent to a case that has been solved.

Table 4. Lemma 4 mappings that map each A case in Fig. 1 to a non-A case in Fig. 1. The mappings given on the left of each row are applied to the cases in that row to show that each case is regular equivalent to another case in Fig. 1. Here $Q' \in \{Q_1, Q_2, \ldots Q_9\}$, $Q_m \in \{Q_2, Q_5, Q_6, Q_9\}$, $Q_l \in \{Q_6, Q_8, Q_9\}$, and $Q_p \in \{Q_5, Q_8, Q_9\}$. Keep in mind Footnote 1 when applying $f_{\text{b-flip}}$.

$f_{\text{s-flip}}$	(Σ_1, D_3, Q')	(Σ_1, D_4, Q')	(Σ_6, D_3, Q')	(Σ_6, D_6, Q_l)
$f_{\text{b-flip}}$	(Σ_5, D_2, Q')	(Σ_5, D_4, Q')	(Σ_5, D_5, Q_m)	(Σ_6, D_2, Q')
	(Σ_6, D_5, Q_5)			
$f_{\text{s-flip}}, f_{\text{b-flip}}$	(Σ_4, D_3, Q')	(Σ_4, D_4, Q')	(Σ_4, D_7, Q_p)	(Σ_6, D_4, Q')
$f_{\text{s-flip}}, f_{\text{d-flip}}$	(Σ_6, D_5, Q_l)	(Σ_6, D_7, Q_l)		
$f_{\text{b-flip}}, f_{\text{d-flip}}$	(Σ_5, D_6, Q_m)	(Σ_5, D_7, Q_m)	(Σ_6, D_6, Q_5)	(Σ_6, D_7, Q_5)
$f_{\text{s-flip}}, f_{\text{b-flip}}, f_{\text{d-flip}}$	(Σ_4, D_5, Q_p)	(Σ_4, D_6, Q_p)		

4.2 Cases B: Machines that Give Periodic Computations

In Fig. 1, if the square for a machine (Σ_i, D_j, Q_k) contains a B then the machine's computation is periodic. From Lemma 2 and Comment 1, periodicity can be proved by showing that a machine eventually scans right never again visiting cell $i - m$ after it visits cell i, or alternatively that the head scans left never again visiting cell $i + m$ after cell i. To achieve this we consider what happens after the tape head attempts to scan in the opposite direction after either $\geqslant 2$ consecutive right moves or $\geqslant 2$ consecutive left moves. Comment 2 allows us to consider only machines that make consecutive left moves and consecutive right moves. As an example of the above technique we show that after machine (Σ_5, D_6, Q_8) (given in (11)) makes 2 or more consecutive left moves it scans left for the rest of the computation never again visiting cell $i + 3$ after it visits cell i.

$$q_a, 0, 0, L, q_b \qquad q_a, 1, 1, R, q_b \qquad q_b, 0, 1, L, q_a \qquad q_b, 1, 0, R, q_b \tag{11}$$

If the tape head is about to make a right move that follows $\geqslant 2$ consecutive left moves then either the configuration has the form given on the left of (12) or

it has the form given on the left of (13) (where $\sigma'_i, \sigma'_{-i} \in \{0,1\}$). On the right of (12) we see that after 4 time steps the configuration has the same form as the configuration on the left side of (13) (so we need consider only the case in (13)). On the right of (13) we see that after 3 time steps the tape head is now reading σ'_{-1}, one cell to the left of its original position. If $\sigma'_{-1} = 0$ the tape head will move left again to make its second consecutive left move and so if we wish to make a right move in the future it will follow $\geqslant 2$ consecutive left moves and we will repeat the entire sequence of steps we have just described. Alternatively, if $\sigma'_{-1} = 1$ then we have the same case as the left side of (13) and the 3 steps in (13) are repeated. For both cases (12) and (13) the process repeats with the tape head moving left never visiting cell $i+3$ after it has entered cell i, and so from Comment 1 the computation is periodic. The technique given here can be applied to all B cases in Fig. 1 to show that they have periodic computations.

$$\ldots \sigma'_{-2}\sigma'_{-1}\, \mathbf{q_a} 1\, 10 \sigma'_1\sigma'_2\sigma'_3 \ldots \quad \vdash^4 \quad \ldots \sigma'_{-2}\sigma'_{-1}\, \mathbf{q_b} 1\, 01 \sigma'_1\, \sigma'_2\sigma'_3 \ldots \tag{12}$$

$$\ldots \sigma'_{-2}\sigma'_{-1}\, \mathbf{q_b} 1\, 01 \sigma'_1\sigma'_2\sigma'_3 \ldots \quad \vdash^3 \quad \ldots \sigma'_{-4}\, \mathbf{q_b} \sigma'_{-1}\, 011 \sigma'_1\sigma'_2\sigma'_3 \ldots \tag{13}$$

4.3 Cases C: Machines that Give Semi-periodic Computation

From the proof of Lemma 3 a semi-periodic machine operates by scanning right using a repeating sequence of rules that print a repeating pattern until some sequence of symbols is met on the tape that causes the machine to end the scan right, and following this the machine scans left printing out another repeating pattern until it meets another sequence of symbols on the tape that causes the scan left to end. The entire process is then repeated. The behaviour of the machine when ending a rightward scan depends on the sequence of symbols it reads when it ends the scan right. The scan length increases with each subsequent pass and the sequence of symbols that ends each scan is provided by the blank word that is repeated to the right of the input. Since there is only a finite number of positions at which a scan right can end in relation to the right blank word the behaviour of the machine at the end of scans to the right becomes periodic over time. This can be seen in Definition 4 and Lemma 3 where the first scan right ends with e_1, the second scan ends with e_2, the third with e_3, and so on until scan x which ends with e_x. Following scan x the process is repeated with e_1 ending the next scan right. The same is true for scans left with the sequence $h_1, h_2, \ldots h_x$ being repeated once for every x scans to the left. When a 2-state 2-symbol machine is executing a semi-periodic computation the patterns it prints during leftward and rightward scans have length $\leqslant 2$ and the behaviour at the end of left and right scans are repeated periodically as described above. For this reason it is straightforward to determine when a 2-state 2-symbol machine is executing a semi-periodic computation.

As an example let us consider the machine (Σ_6, D_6, Q_3) given in (14). It is easy to determine the behaviour of this machine just by looking at its instructions. The machine scans left in state q_a reading 0's and printing 1's to the tape until it reads a 1, then it scans right in state q_b reading 1's and printing 0's until it reads a 0, and then it begins another scan left in q_a. We now show how this

behaviour matches the behaviour of the sequence S given in Definition 4. In the scans mentioned above the growth to the left depends on the position of 1's in the left blank word as each 1 terminates a scan left, and similarly the growth to the right depends on the position of 0's in the right blank word. Since the left and right blank words are repeated periodically the growth is periodic for some constant number of scans right, this constant is the x value in sequence S. We set sequences s_r and s_l so that $f(s_r) = m$ and $f(s_l) = -m$ (see Definition 4) where m is the growth over the x scans left and right during an iteration of S, and this means that during iteration i of S, scans right have the form $(s_r)^{i+1}$ and scans left have the form $(s_l)^{i+1}$. To account for the growth between successive left and right scans during a single iteration of S, we define each e_j and h_j so that the extra distance traveled between each successive execution of $(s_r)^{i+1}$ or $(s_l)^{i+1}$ during S is considered part of e_j or h_j. This completes our explanation of how to give the $2x+2$ sequence that define S in Definition 4. All the machines for Case C can be shown to be semi-periodic using similar analysis to that given above.

$$q_a, 0, 1, L, q_a \qquad q_a, 1, 0, R, q_b \qquad q_b, 0, 1, L, q_a \qquad q_b, 1, 0, R, q_b \tag{14}$$

4.4 Cases *E*: Binary Counters

The two binary machines in Fig. 1 compute in a similar manner and so we will just look at machine (Σ_5, D_6, Q_3) given in (15). Below we show (Σ_5, D_6, Q_3) incrementing from 4 to 8. The left most 1 is not part of the number and the most significant bit is on the right. To increment a number the machine scans right in q_b changing 1's to 0's until it reads a 0 which it changes to a 1 and it then scans left in q_a until it reads a 1, which signals the beginning of the next increment. It is easy to see that when started on any configuration the set of configurations generated by this machine is regular as it generates all possible strings and the scans left and right that increment the number have a simple form. If the machine has periodic support then the set of reachable configurations remains regular. The repeated words on the left have no effect on the computation as the tape head can not move left over a 1. On the right when each blank words becomes part of the computation it effects the form of only a constant number of bits at the right end of the number and so the set of reachable configurations is regular.

$$\mathbf{q_a}1\,0010 \;\vdash^2\; \mathbf{q_a}1\,1010 \;\vdash^4\; \mathbf{q_a}1\,0110 \;\vdash^2\; \mathbf{q_a}1\,1110 \;\vdash^8\; \mathbf{q_a}1\,0001$$

$$q_a, 0, 0, L, q_a \qquad q_a, 1, 1, R, q_b \qquad q_b, 0, 1, L, q_a \qquad q_b, 1, 0, R, q_b \tag{15}$$

References

1. Baiocchi, C.: Three small universal Turing machines. In: Margenstern, M., Rogozhin, Y. (eds.) MCU 2001. LNCS, vol. 2055, pp. 1–10. Springer, Heidelberg (2001). doi:10.1007/3-540-45132-3_1
2. Cook, M.: Universality in elementary cellular automata. Complex Syst. **15**(1), 1–40 (2004)

3. Hermann, G.: The uniform halting problem for generalized one state Turing machines. In: Proceedings, Ninth Annual Symposium on Switching and Automata Theory (FOCS), pp. 368–372. IEEE Computer Society Press, October 1968
4. Kudlek, M.: Small deterministic Turing machines. TCS **168**(2), 241–255 (1996)
5. Minsky, M.: Size and structure of universal Turing machines using tag systems. In: Recursive Function Theory, Symposium in Pure Mathematics, vol. 5, pp. 229–238 (1962)
6. Neary, T., Woods, D.: Four small universal Turing machines. Fundam. Inform. **91**(1), 123–144 (2009)
7. Neary, T., Woods, D.: Small weakly universal Turing machines. In: Kutyłowski, M., Charatonik, W., Gębala, M. (eds.) FCT 2009. LNCS, vol. 5699, pp. 262–273. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03409-1_24
8. Neary, T., Woods, D.: The complexity of small universal Turing machines: a survey. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) SOFSEM 2012. LNCS, vol. 7147, pp. 385–405. Springer, Heidelberg (2012). doi:10.1007/978-3-642-27660-6_32
9. Pavlotskaya, L.: Solvability of the halting problem for certain classes of Turing machines. Math. Notes (Springer) **13**(6), 537–541 (1973)
10. Pavlotskaya, L.: Dostatochnye uslovija razreshimosti problemy ostanovki dlja mashin T'juring. Problemi kibernetiki, pp. 91–118 (1978). (in Russian)
11. Rogozhin, Y.: Small universal Turing machines. TCS **168**(2), 215–240 (1996)
12. Wagner, K.: Universelle Turingmaschinen mit n-dimensionale band. Elektronische Informationsverarbeitung und Kybernetik **9**(7–8), 423–431 (1973)
13. Watanabe, S.: 5-symbol 8-state and 5-symbol 6-state universal Turing machines. J. ACM **8**(4), 476–483 (1961)