# Small weakly universal Turing machines*

Turlough Neary[1] and Damien Woods[2]

[1] Boole Centre for Research in Informatics,
University College Cork, Ireland.
tneary@cs.may.ie
[2] Department of Computer Science and Artificial Intelligence
University of Seville, Spain.
dwoods@us.es

**Abstract.** We give small universal Turing machines with state-symbol pairs of $(6, 2)$, $(3, 3)$ and $(2, 4)$. These machines are weakly universal, which means that they have an infinitely repeated word to the left of their input and another to the right. They simulate Rule 110 and are currently the smallest known weakly universal Turing machines. Despite their small size these machines are efficient polynomial time simulators of Turing machines.

## 1 Introduction

Shannon [22] was the first to consider the problem of finding the smallest universal Turing machine, where size is the number of states and symbols. Here we say that a Turing machine is standard if it has a single one-dimensional tape, one tape head, and is deterministic [7]. Over the years, small universal programs were given for a number of variants on the standard model. By generalising the standard model we often find smaller universal programs. One such generalisation is to allow the blank portion of the Turing machine's tape to have an infinitely repeated word to the left, and another to the right. We refer to such universal machines as weakly universal Turing machines, and they are the subject of this work.

Beginning in the early sixties Minsky and Watanabe engaged in a vigorous competition to see who could come up with the smallest universal Turing machine [13, 14, 23, 24]. In 1961, Watanabe [23] gave a 6-state, 5-symbol machine that was the first weakly universal machine. In 1962, Minsky [14] found a small 7-state, 4-symbol standard universal Turing machine. Not to be out-done, Watanabe improved on his earlier machine to give 5-state, 4-symbol and 7-state, 3-symbol weakly universal machines [24].

* Turlough Neary is funded by the Irish Research Council for Science, Engineering and Technology and by Science Foundation Ireland Research Frontiers Programme grant number 07/RFP/CSMF641. Damien Woods was supported by a Project of Excellence from the Junta de Andalucía grant TIC-581, and by Science Foundation Ireland grant 04/IN3/1524.

The 7-state universal Turing machine of Minsky has received much attention. Minsky's machine simulates Turing machines via 2-tag systems, which were proved universal by Cocke and Minsky [3]. The technique of simulating 2-tag systems, pioneered by Minsky, was extended by Rogozhin [21] to give the (then) smallest known universal Turing machines for a number of state-symbol pairs. Many of these 2-tag simulators were subsequently reduced in size by Kudlek and Rogozhin [9], and Baiocchi [2]. Neary and Woods [17] gave small universal machines that simulate Turing machines via a new variant of tag systems called bi-tag systems. All of the smallest known universal Turing machines, that obey the standard definition (deterministic, one tape, one head), simulate either 2-tag or bi-tag systems. They are plotted as circles and triangles in Figure 1. To get the polynomial time overhead for 2-tag simulators in Figure 1 the 2-tag simulation of Turing Machines given in [15, 26] is used instead of the exponentially slow technique given in [3].

The small weak machines of Watanabe have received little attention. In particular the 5-state and 7-state machines seem little known and are largely ignored in the literature. It is worth noting that while all other weak machines simulate Turing machine via other simple models, Watanabe's weak machines simulate Turing machines directly. His machines are the most time efficient of the small weak machines. More precisely, let $t$ be the running time of any deterministic single tape Turing machine $M$, then Watanabe's machines are the smallest weak machines that simulate $M$ with a time overhead of $O(t^2)$.

We often refer to Watanabe's machines as being semi-weak. Semi-weak machines are a restriction of weak machines: they have an infinitely repeated word to one side of their input, and on the other side they have a (standard) infinitely repeated blank symbol. Recently, Woods and Neary [28] have given semi-weakly universal machines that simulate cyclic tag systems with state-symbol pairs of $(3, 7)$, $(4, 5)$ and $(2, 13)$. All of the smallest known semi-weakly universal machines are plotted as diamonds in Figure 1.

Cook [4] and Wolfram [25], recently gave weakly universal Turing machines, smaller than Watanabe's semi-weak machines, that simulate the universal cellular automaton Rule 110. These machines have state-symbol pairs of $(7, 2)$, $(4, 3)$, $(3, 4)$ and $(2, 5)$ and are plotted as hollow squares in Figure 1. (Note that David Eppstein constructed the $(7, 2)$ machine to be found in [4].)

Here we present weakly universal Turing machines with state-symbol pairs of $(6, 2)$, $(3, 3)$ and $(2, 4)$ making them the smallest known weakly universal machines. Our machines efficiently simulate (single tape, deterministic) Turing machines in time $O(t^4 \log^2 t)$, via Rule 110. These machines are plotted as solid squares in Figure 1 and induce a weakly universal curve.

Weakness has not been the only generalisation on the standard model in the search for small universal Turing machines. Priese [20] gave a 2-state, 4-symbol machine with a 2-dimensional tape, and a 2-state, 2-symbol machine with a 2-dimensional tape and 2 tape heads. Margenstern and Pavlotskaya [11] gave a 2-state, 3-symbol Turing machine that is universal when coupled with a finite automaton. The Turing machine part of this couple uses only 5 instructions, and

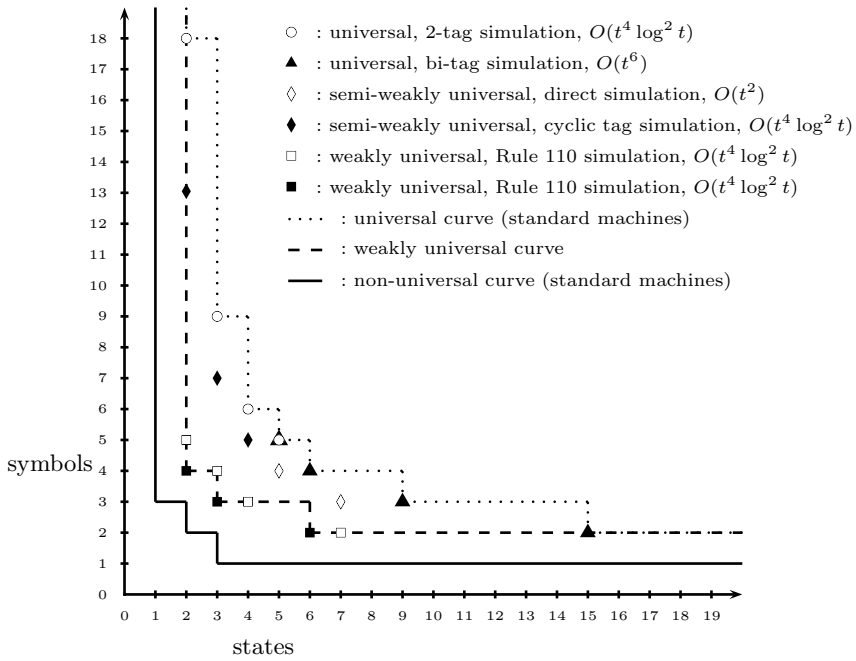


**Fig. 1.** State-symbol plot of small universal Turing machines. Each of our new weak machines is represented by a solid square. These machines induce a weakly universal curve. Simulation time overheads are specified. The non-universal curve shows standard machines that are known to have a decidable halting problem.

they also show that the halting problem is decidable for couples in which the Turing machine has only 4 instructions. Hence, it is not possible to have a universal couple with a 4-instruction Turing machine that simulates any Turing machine $M$ and halts if and only if $M$ halts. Thus, they have given the smallest possible Turing machine that is universal when coupled with a finite automaton. It is worth noting that the weakly universal machines that we present in this paper have the smallest number of instructions of any known universal machines with polynomial time overhead. This comparison even includes all other generalised Turing machine models such as those mentioned above: all known machines that use fewer instructions but generalise other aspects (multiple tapes, coupling with automata etc.) of the model are exponentially slow.

More on small universal Turing machines, and related notions, can be found in [10, 15, 27].

### 1.1 Preliminaries

The Turing machines considered in this paper are deterministic and have a single bi-infinite tape. We let $U_{m,n}$ denote our weakly universal Turing machine with $m$ states and $n$ symbols. We write $c_1 \vdash c_2$ if a configuration $c_2$ is obtained from $c_1$ via a single computation step. We let $c_1 \vdash^s c_2$ denote a sequence of $s$ computation steps, and let $c_1 \vdash^* c_2$ denote zero or more computation steps.

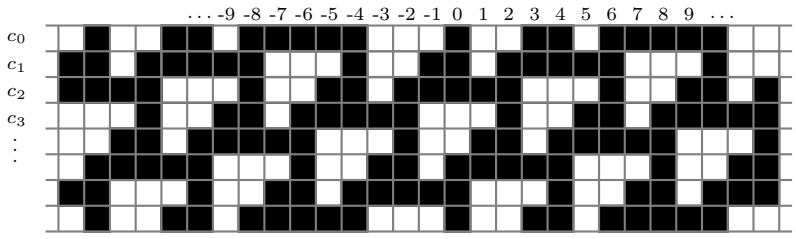


**Fig. 2.** Seven consecutive timesteps of Rule 110. These seven timesteps show the evolution of the background ether that is used in the proof [4] of universality of Rule 110. Each black or each white square represents, a Rule 110 cell containing, state 1 or 0 respectively. Each cell is identified by the index given above it. To the left of each row of cells there is a configuration label that identifies that row.

## 2 Rule 110

Rule 110 is a very simple (2 state, nearest neighbour, one dimensional) cellular automaton. It is composed of a sequence of cells $\ldots p_{-1}p_0p_1\ldots$ where each cell has a binary state $p_i \in \{0,1\}$. At timestep $s+1$, the value $p_{i,s+1} = F(p_{i-1,s}, p_{i,s}, p_{i+1,s})$ of the cell at position $i$ is given by the synchronous local update function $F$

$$\begin{aligned} F(0,0,0) &= 0 & F(1,0,0) &= 0 \\ F(0,0,1) &= 1 & F(1,0,1) &= 1 \\ F(0,1,0) &= 1 & F(1,1,0) &= 1 \\ F(0,1,1) &= 1 & F(1,1,1) &= 0 \end{aligned} \tag{1}$$

Rule 110 was proven universal by Cook [4] (Cook's proof is sketched in [25]). Neary and Woods [16] proved that Rule 110 simulates Turing machines efficiently in polynomial time $O(t^3 \log t)$, an exponential improvement. This time overhead was further improved to $O(t^2 \log t)$ [15]. Rule 110 simulates cyclic tag systems in linear time. The weak machines in this paper, and in [4, 25], simulate Rule 110 with a quadratic polynomial increase in time and hence simulate Turing machines in time $O(t^4 \log^2 t)$. It is worth noting that the prediction problem [5] for these machines is P-complete, and this is also the case when we consider only bounded initial conditions [16].

## 3 Three small weakly universal Turing machines

The following observation is one of the reasons for the improvement in size over previous weak machines, and gives some insight into the simulation algorithm we use. Notice from Equation (1) that the value of the update function $F$, with the exception of $F(0,1,1)$ and $F(1,1,1)$, may be determined using only the rightmost two states. Each of our universal Turing machines exploit this fact as follows. The machines scan from right to left, and in six of the eight cases they need only remember the cell immediately to the right of the current cell $i$ in

order to compute the update for $i$. Thus for these six cases we need only store a single cell value, rather than two values. The remaining two cases are simulated as follows. If two consecutive encoded states with value 1 are read, it is assumed that there is another encoded 1 to the left and the update $F(1,1,1) = 0$ is simulated. If our assumption proves false (we instead read an encoded 0 to the left), then our machine returns to the wrongly updated cell and simulates the update $F(0,1,1) = 1$.

Before giving our three small Rule 110 simulators, we give some further background explanation. Rule 110 simulates Turing machines via cyclic tag systems. A Rule 110 instance that simulates a cyclic tag system computation is of the following form (for more details see [4]). The input to the cyclic tag system is encoded in a contiguous finite number of Rule 110 cells. On the left of the input a fixed constant word (representing the 'ossifiers') is repeated infinitely many times. On the right, another fixed constant word (representing the cyclic tag system program/appendants, and the 'leaders') is repeated infinitely many times. Both of these repeated words are independent of the input.

As in [4], our weakly universal machines operate by traversing a finite amount of the tape from left to right and then from right to left. This simulates a single timestep of Rule 110 over a finite part of the encoded infinite Rule 110 instance. With each simulated timestep the length of a traversal increases. To ensure that each traversal is of finite length, the left blank word $l$ and the right blank word $r$ of each of our weak machines must have a special form. These words contain special subwords or symbols that terminate each traversal, causing the tape head to turn. When the head is turning it overwrites any symbols that caused a turn. Thus the number of cells that are being updated increases monotonically over time. This technique simulates Rule 110 properly if the initial condition is set up so that within each repeated blank word, the subword between each successive turn point is shifted one timestep forward in time.

In the sequel we describe the computation of our three machines by showing a simulation of the update on the ether in Figure 2. In the next paragraph below, we outline why this example is in fact general enough to prove universality. First, we must define blank words that are suitable for this example. The left blank word $l$, on the Turing machine tape, encodes the Rule 110 sequence 0001. In the initial configuration as we move left each subsequent sequence 0001 is one timestep further ahead. To see this note from Figure 2 that 0001 occupies, cells $-7$ to $-4$ in configuration $c_1$, cells $-11$ to $-8$ in $c_2$, cells $-15$ to $-12$ in $c_3$, etc. Similarly, the right blank word $r$ encodes the Rule 110 sequence 110011. Looking at the initial configuration, as we move right from cell 0, in the first blank word the first four cells 1100 are shifted two timesteps ahead, and the next two cells 11 are shifted a further one timestep. To see this note from Figure 2 that 1100 occupies cells 1 to 4 in $c_2$ and 11 occupies cells 5 and 6 in $c_3$. In each subsequent sequence the first four cells 1100 are shifted only one timestep ahead and the last two cells 11 are shifted one further timestep. In each row the ether in Figure 2 repeats every 14 cells and if the number of timesteps $s$ between two rows is $s \equiv 0$

mod 7 then the two rows are identical. The periodic nature of the ether, in both time and space, allows us to construct such blank words.

It should be noted that the machines we present here, and those in [4, 25], require suitable blank words to simulate a Rule 110 instance directly. If no suitable blank words can be found (i.e. if it is not possible to construct subwords that terminate traversals in the encoding) then it may be the case that the particular instance can not be simulated directly. In the sequel our machines simulate the background ether that is used in the universality proof of Rule 110 [4]. The gliders used by Cook [4] that move through this ether are periodic in time and space. Thus, we can construct blank words that include these gliders and place the subwords that terminate traversals in the ether. By this reasoning, our example is sufficiently general to prove that our machines simulate Turing machines via Rule 110 and we do not give a full (and possibly tedious) proof of correctness. For $U_{3,3}$ we explicitly simulate three updates from Figure 2, which is general enough so that an update [Equation (1)] on each of the eight possible three state combinations is simulated. We give shorter examples for the machines $U_{2,4}$ and $U_{6,2}$ as they use the same simulation algorithm as $U_{3,3}$.

As with the machines in [4, 25], the machines we present here do not halt. Cook [4] shows how a special glider may be produced during the simulation of a Turing machine by Rule 110. This glider may be used to simulate halting as the encoding can be such that it is generated by Rule 110 if and only if the simulated machine halts. The glider would be encoded on the tape of our machines as a unique, constant word.

### 3.1 $U_{3,3}$

We begin by describing an *initial* configuration of $U_{3,3}$. To the left of, and including, the tape head position, the Rule 110 state 0 is encoded by 0, and the Rule 110 state 1 is encoded by either 1 or $b$. The word $1b0$ is used to terminate a left traversal. (Note an exception: the 1 in the subword $1b0$ encodes the Rule 110 state 0.) To the right of the tape head position, the Rule 110 state 0 is encoded by 1, and the Rule 110 state 1 is encoded by 0 or $b$. The tape symbol 0 is used to terminate a right traversal. The left and right blank words, described in paragraph 4 of Section 3, are encoded as $001b$ and $0b110b$ respectively.

| | $u_1$ | $u_2$ | $u_3$ |
|---|---|---|---|
| 0 | $1Lu_1$ | $0Ru_1$ | $bLu_1$ |
| 1 | $bLu_2$ | $1Lu_2$ | $0Ru_3$ |
| $b$ | $bLu_3$ | | $1Ru_3$ |

**Table 1.** Table of behaviour for $U_{3,3}$.

We give an example of $U_{3,3}$ simulating the three successive Rule 110 timesteps $c_0 \vdash c_1 \vdash c_2 \vdash c_3$ given in Figure 2. In the below configurations the current

state of $U_{3,3}$ is highlighted in bold, to the left of its tape contents. The tape head position of $U_{3,3}$ is given by an underline and the start state is $u_1$. The configuration immediately below encodes $c_0$ from Figure 2 with the tape head over cell index 0.

$$
\begin{array}{rccccc}
 & \boldsymbol{u_1}, \ldots 001b & 001b & 001b & 0001 & 0b110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_2}, \ldots 001b & 001b & 001b & 00\underline{0}b & 0b110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_1}, \ldots 001b & 001b & 001b & 000\underline{b} & 0b110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_3}, \ldots 001b & 001b & 001b & 00\underline{0}b & 0b110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_1}, \ldots 001b & 001b & 001b & 0\underline{0}bb & 0b110b & 0b110b\ldots \\
\vdash^2 & \boldsymbol{u_1}, \ldots 001b & 001b & 001\underline{b} & 11bb & 0b110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_3}, \ldots 001b & 001b & 00\underline{1}b & 11bb & 0b110b & 0b110b\ldots
\end{array}
$$

When the tape head reads the subword $1b0$ the left traversal is complete and the right traversal begins.

$$
\begin{array}{rccccc}
\vdash^6 & \boldsymbol{u_3}, \ldots 001b & 001b & 0001 & 0011 & \underline{0}b110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_1}, \ldots 001b & 001b & \mathbf{0001} & \mathbf{0011} & bb110b & 0b110b\ldots
\end{array}
$$

Immediately after the tape head reads a 0, during a right traversal, the simulation of timestep $c_0 \vdash c_1$ is complete. To see this, compare the part of the Turing machine tape in bold with cells $-7$ to 0 of configuration $c_1$ in Figure 2. We continue our simulation to give timestep $c_1 \vdash c_2$.

$$
\begin{array}{rccccc}
\vdash & \boldsymbol{u_2}, \ldots 001b & 001b & 0001 & 00\underline{1}b & bb110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_2}, \ldots 001b & 001b & 0001 & 0\underline{0}1b & bb110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_1}, \ldots 001b & 001b & 0001 & 00\underline{1}b & bb110b & 0b110b\ldots \\
\vdash & \boldsymbol{u_2}, \ldots 001b & 001b & 0001 & 0\underline{0}bb & bb110b & 0b110b\ldots \\
\vdash^3 & \boldsymbol{u_1}, \ldots 001b & 001b & 0001 & \underline{0}bbb & bb110b & 0b110b\ldots \\
\vdash^2 & \boldsymbol{u_2}, \ldots 001b & 001b & 00\underline{0}b & 1bbb & bb110b & 0b110b\ldots \\
\vdash^3 & \boldsymbol{u_1}, \ldots 001b & 001b & 0\underline{0}bb & 1bbb & bb110b & 0b110b\ldots \\
\vdash^3 & \boldsymbol{u_3}, \ldots 001b & 00\underline{1}b & 11bb & 1bbb & bb110b & 0b110b\ldots \\
\vdash^{15} & \boldsymbol{u_1}, \ldots 001b & \mathbf{0001} & \mathbf{0011} & \mathbf{0111} & \mathbf{1100}\underline{0}bb & 0b110b\ldots
\end{array}
$$

The simulation of timestep $c_1 \vdash c_2$ is complete. To see this, compare the part of the Turing machine tape in bold with cells $-11$ to 4 of configuration $c_2$ in Figure 2. We continue our simulation to give timestep $c_2 \vdash c_3$.

$$
\begin{array}{rccccc}
\vdash^3 & \boldsymbol{u_2}, \ldots 001b & 0001 & 0011 & 0111 & \underline{1}b11bb & 0b110b\ldots \\
\vdash^4 & \boldsymbol{u_2}, \ldots 001b & 0001 & 0011 & \underline{0}111 & 1b11bb & 0b110b\ldots
\end{array}
$$

$$
\begin{array}{rlllllll}
\vdash^{5} & \boldsymbol{u_1}, \ldots 0\,0\,1\,b & 0\,0\,0\,1 & 0\,0\,1\,\underline{1} & b\,b\,1\,1 & 1\,b\,1\,1\,b\,b & 0\,b\,1\,1\,0\,b\ldots \\
\vdash^{2} & \boldsymbol{u_2}, \ldots 0\,0\,1\,b & 0\,0\,0\,1 & 0\,\underline{0}\,1\,b & b\,b\,1\,1 & 1\,b\,1\,1\,b\,b & 0\,b\,1\,1\,0\,b\ldots \\
\vdash^{5} & \boldsymbol{u_1}, \ldots 0\,0\,1\,b & 0\,0\,0\,1 & \underline{0}\,b\,b\,b & b\,b\,1\,1 & 1\,b\,1\,1\,b\,b & 0\,b\,1\,1\,0\,b\ldots \\
\vdash^{2} & \boldsymbol{u_2}, \ldots 0\,0\,1\,b & 0\,0\,\underline{0}\,b & 1\,b\,b\,b & b\,b\,1\,1 & 1\,b\,1\,1\,b\,b & 0\,b\,1\,1\,0\,b\ldots \\
\vdash^{6} & \boldsymbol{u_3}, \ldots 0\,0\,\underline{1}\,b & 1\,1\,b\,b & 1\,b\,b\,b & b\,b\,1\,1 & 1\,b\,1\,1\,b\,b & 0\,b\,1\,1\,0\,b\ldots \\
\vdash^{21} & \boldsymbol{u_1}, \ldots \mathbf{0\,0\,0\,1} & \mathbf{0\,0\,1\,1} & \mathbf{0\,1\,1\,1} & \mathbf{1\,1\,0\,0} & \mathbf{0\,1\,0\,0\,1\,\underline{1}} & b\,b\,1\,1\,0\,b\ldots
\end{array}
$$

The simulation of timestep $c_2 \vdash c_3$ is complete. To see this, compare the part of the Turing machine tape in bold with cells $-15$ to 6 of configuration $c_3$ in Figure 2.

### 3.2 $U_{2,4}$

We begin by describing an *initial* configuration of $U_{2,4}$. To the left of, and including, the tape head position, the Rule 110 state 0 is encoded by either 0 or $\emptyset$ and the Rule 110 state 1 is encoded by either 1 or $\not{1}$. The word $\emptyset\,1$ is used to terminate a left traversal. To the right of the tape head position, the Rule 110 state 0 is encoded by $\emptyset$ and the Rule 110 state 1 is encoded by $\not{1}$ or 0. The tape symbol 0 is used to terminate a right traversal. The left and right blank words, from paragraph 4 of Section 3, are encoded as $0\,0\,\emptyset\,1$ and $0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1}$ respectively.

| | $u_1$ | $u_2$ |
|---|---|---|
| 0 | $\emptyset\,Lu_1$ | $\not{1}\,Ru_1$ |
| 1 | $\not{1}\,Lu_2$ | $\emptyset\,Lu_2$ |
| $\emptyset$ | $\not{1}\,Lu_1$ | $0Ru_2$ |
| $\not{1}$ | $\not{1}\,Lu_1$ | $1Ru_2$ |

**Table 2.** Table of behaviour for $U_{2,4}$.

We give an example of $U_{2,4}$ simulating the two successive Rule 110 timesteps $c_0 \vdash c_1 \vdash c_2$ given in Figure 2. The configuration immediately below encodes $c_0$ from Figure 2 with the tape head over cell index 0.

$$
\begin{array}{rlllll}
 & \boldsymbol{u_1}, \ldots 0\,0\,\emptyset\,1 & 0\,0\,\emptyset\,1 & 0\,0\,\emptyset\,1 & 0\,0\,0\,\underline{1} & 0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1} & 0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1}\ldots \\
\vdash^{6} & \boldsymbol{u_1}, \ldots 0\,0\,\emptyset\,1 & 0\,0\,\emptyset\,1 & 0\,0\,\emptyset\,\underline{1} & \emptyset\,\emptyset\,\not{1}\,\not{1} & 0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1} & 0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1}\ldots \\
\vdash & \boldsymbol{u_2}, \ldots 0\,0\,\emptyset\,1 & 0\,0\,\emptyset\,1 & 0\,0\,\underline{\emptyset}\,\not{1} & \emptyset\,\emptyset\,\not{1}\,\not{1} & 0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1} & 0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1}\ldots
\end{array}
$$

When the tape head reads the subword $\emptyset\,1$ the left traversal is complete and the right traversal begins.

$$
\begin{array}{rlllll}
\vdash^{6} & \boldsymbol{u_2}, \ldots 0\,0\,\emptyset\,1 & 0\,0\,\emptyset\,1 & 0\,0\,0\,1 & 0\,0\,1\,1 & \underline{0}\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1} & 0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1}\ldots \\
\vdash & \boldsymbol{u_1}, \ldots 0\,0\,\emptyset\,1 & 0\,0\,\emptyset\,1 & \mathbf{0\,0\,0\,1} & \mathbf{0\,0\,1\,1} & \not{1}\,\underline{\not{1}}\,\emptyset\,\emptyset\,0\,\not{1} & 0\,\not{1}\,\emptyset\,\emptyset\,0\,\not{1}\ldots
\end{array}
$$

Immediately after the tape head reads a 0, during a right traversal, the simulation of timestep $c_0 \vdash c_1$ is complete. To see this, compare the part of the Turing machine tape in bold with cells $-7$ to 0 of configuration $c_1$ in Figure 2. We continue our simulation to give timestep $c_1 \vdash c_2$.

$$
\begin{array}{rl}
\vdash^2 & \boldsymbol{u_1}, \ldots 00\not\emptyset 1 \quad 00\not\emptyset 1 \quad 0001 \quad 001\underline{1} \quad \not1\not1\not\emptyset\not0\not1 \quad 0\not1\not\emptyset\not0\not1 \ldots \\
\vdash^2 & \boldsymbol{u_2}, \ldots 00\not\emptyset 1 \quad 00\not\emptyset 1 \quad 0001 \quad 0\underline{0}\not\emptyset\not1 \quad \not1\not1\not\emptyset\not0\not1 \quad 0\not1\not\emptyset\not0\not1 \ldots \\
\vdash & \boldsymbol{u_1}, \ldots 00\not\emptyset 1 \quad 00\not\emptyset 1 \quad 0001 \quad 0\not1\underline{\not0}\not1 \quad \not1\not1\not\emptyset\not0\not1 \quad 0\not1\not\emptyset\not0\not1 \ldots \\
\vdash^4 & \boldsymbol{u_2}, \ldots 00\not\emptyset 1 \quad 00\not\emptyset 1 \quad 00\underline{0}\not1 \quad \not\emptyset\not1\not1\not1 \quad \not1\not1\not\emptyset\not0\not1 \quad 0\not1\not\emptyset\not0\not1 \ldots \\
\vdash^5 & \boldsymbol{u_1}, \ldots 00\not\emptyset 1 \quad 00\not\emptyset\underline{1} \quad \not\emptyset\not\emptyset\not1\not1 \quad \not\emptyset\not1\not1\not1 \quad \not1\not1\not\emptyset\not0\not1 \quad 0\not1\not\emptyset\not0\not1 \ldots \\
\vdash & \boldsymbol{u_2}, \ldots 00\not\emptyset 1 \quad 00\underline{\not0}\not1 \quad \not\emptyset\not\emptyset\not1\not1 \quad \not\emptyset\not1\not1\not1 \quad \not1\not1\not\emptyset\not0\not1 \quad 0\not1\not\emptyset\not0\not1 \ldots \\
\vdash^{15} & \boldsymbol{u_1}, \ldots 00\not\emptyset 1 \quad \mathbf{0001} \quad \mathbf{0011} \quad \mathbf{0111} \quad \mathbf{1100}\not1\underline{\not1} \quad 0\not1\not\emptyset\not0\not1 \ldots
\end{array}
$$

The simulation of timestep $c_1 \vdash c_2$ is complete. To see this, compare the part of the Turing machine tape in bold with cells $-11$ to 4 of configuration $c_2$ in Figure 2.

### 3.3 $U_{6,2}$

We begin by describing an *initial* configuration of $U_{6,2}$. To the left of, and including, the tape head position, the Rule 110 state 0 is encoded by the word 00 and the Rule 110 state 1 is encoded by the word 11. The word 010100 is used to terminate a left traversal and encodes the sequence of Rule 110 states 010. To the right of the tape head position the Rule 110 state 0 is encoded by the word 00 and the Rule 110 state 1 is encoded by either of the words 01 or 10. The word 10 is used to terminate a right traversal. The left and right blank words, from paragraph 4 of Section 3, are encoded as 00000101 and 100100001001 respectively.

| | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ |
|---|---|---|---|---|---|---|
| 0 | $0Lu_1$ | $0Lu_6$ | $0Ru_2$ | $1Ru_5$ | $1Lu_4$ | $1Lu_1$ |
| 1 | $1Lu_2$ | $0Lu_3$ | $1Lu_3$ | $0Ru_6$ | $1Ru_4$ | $0Ru_4$ |

**Table 3.** Table of behaviour for $U_{6,2}$.

To illustrate the operation of $U_{6,2}$ we simulate the Rule 110 timestep $c_0 \vdash c_1$ given in Figure 2. The configuration immediately below encodes $c_0$ from Figure 2 with the tape head over cell index 0.

$$
\begin{array}{rl}
& \boldsymbol{u_1}, \ldots 00000101 \quad 00000101 \quad 0000001\underline{1} \quad 100100001001 \ldots \\
\vdash & \boldsymbol{u_2}, \ldots 00000101 \quad 00000101 \quad 000000\underline{1}1 \quad 100100001001 \ldots \\
\vdash & \boldsymbol{u_3}, \ldots 00000101 \quad 00000101 \quad 00000\underline{0}01 \quad 100100001001 \ldots \\
\vdash & \boldsymbol{u_2}, \ldots 00000101 \quad 00000101 \quad 000000\underline{0}1 \quad 100100001001 \ldots
\end{array}
$$

$\vdash$ $u_6$, ...00000101 00000101 00000<u>0</u>01 100100001001...
$\vdash$ $u_1$, ...00000101 00000101 0000<u>0</u>101 100100001001...
$\vdash^5$ $u_1$, ...00000101 0000010<u>1</u> 00000101 100100001001...
$\vdash$ $u_2$, ...00000101 000001<u>0</u>1 00000101 100100001001...
$\vdash$ $u_6$, ...00000101 00000<u>1</u>01 00000101 100100001001...
$\vdash$ $u_4$, ...00000101 000000<u>0</u>1 00000101 100100001001...

When the tape head reads the subword 10100 the left traversal is complete and the right traversal begins.

$\vdash$ $u_5$, ...00000101 0000001<u>1</u> 00000101 100100001001...
$\vdash$ $u_4$, ...00000101 00000011 <u>0</u>0000101 100100001001...
$\vdash$ $u_5$, ...00000101 00000011 1<u>0</u>000101 100100001001...
$\vdash$ $u_4$, ...00000101 00000011 <u>1</u>1000101 100100001001...
$\vdash$ $u_6$, ...00000101 00000011 0<u>1</u>000101 100100001001...
$\vdash$ $u_4$, ...00000101 00000011 00<u>0</u>00101 100100001001...
$\vdash^4$ $u_4$, ...00000101 00000011 0000<u>0</u>101 100100001001...
$\vdash$ $u_5$, ...00000101 00000011 00001<u>1</u>01 100100001001...
$\vdash$ $u_4$, ...00000101 00000011 000011<u>0</u>1 100100001001...
$\vdash^2$ $u_4$, ...00000101 00000011 00001111 <u>1</u>00100001001...
$\vdash$ $u_6$, ...00000101 00000011 00001111 0<u>0</u>0100001001...
$\vdash$ $u_1$, ...00000101 **00000011 00001111** <u>0</u>10100001001...

Immediately after the tape head reads a 10, during a right traversal, the simulation of timestep $c_0 \vdash c_1$ is complete. To see this, compare the part of the Turing machine tape in bold (recall 0 and 1 are encoded as 00 and 11 respectively) with cells $-7$ to 0 of configuration $c_1$ in Figure 2.

## 4 Discussion on lower bounds

The pursuit to find the smallest possible universal Turing machine must also involve the search for lower bounds, finding the largest Turing machines that are in some sense non-universal. One approach is to settle the decidability of the halting problem, but this approach is not suitable for the machines we have presented.

It is known that the halting problem is decidable for (standard) Turing machines with the following state-symbol pairs $(2, 2)$ [8, 18], $(3, 2)$ [19], $(2, 3)$ (claimed by Pavlotskaya [18]), $(1, n)$ [6] and $(n, 1)$ (trivial), where $n \geqslant 1$. Then, these decidability results imply that a universal Turing machine, that simulates any Turing machine $M$ and halts if and only if $M$ halts, is not possible for these state-symbol pairs. Hence these results give lower bounds on the size of universal machines of this type. While it is trivial to prove that the halting problem

is decidable for (possibly halting) weak machines with state-symbol pairs of the form $(n, 1)$, it is not known whether the other decidability results above generalise to (possibly halting) weak Turing machines.

The weakly universal machines presented in this paper, and those in [4, 25], do not halt. Hence the non-universality results discussed in the previous paragraph would have to be generalised to *non-halting weak* machines to give lower bounds that are relevant for our machines. This may prove difficult for two reasons. The first issue is that, intuitively speaking, weakness gives quite an advantage. For instance, the program of a universal machine may be encoded in one of the infinitely repeated blank words of the weak machine. The second issue is related to the problem of defining a computation. Informally, a computation could be defined as a sequence of configurations that ends in a special terminal configuration. For non-halting machines, there are many ways to define a terminal configuration. Given a definition of terminal configuration we may prove that the terminal configuration problem (will a machine ever enter a terminal configuration) is decidable for a machine or set of machines. However this result may not hold as a proof of non-universality if we subsequently alter our definition of terminal configuration. In fact, it may be easily shown that the Turing machine $U_{2,4}$ from Table 2, which we prove weakly universal, is provably non-universal when it is restricted to the standard blank background.

It is trivial that no weakly universal Turing machines exist for the state-symbol pair $(n, 1)$ even when we consider machines with no halting condition. We also believe that relevant decidability results for the state-symbol pair $(2, 2)$ may be given. If this is true, then the problem of whether or not there are 2-state and 3-state weakly universal machines remains open for only $(2, 3)$ and $(3, 2)$ respectively.

Margenstern [10], Baiocchi [1], and Michel [12] have found small machines that simulate iterations of the $3x + 1$ problem and other Collatz-like functions. The smallest known weakly universal machines are almost at the minimum possible size, thus implementing the Collatz problem on weak machines could be an interesting way of exploring the little space remaining between these machines and the state-symbol pairs where weak universality is not possible.

## References


1. C. Baiocchi. 3N+1, UTM e tag-system. Technical Report Pubblicazione 98/38, Dipartimento di Matematico, Università di Roma, 1998. (In Italian).
2. C. Baiocchi. Three small universal Turing machines. In M. Margenstern and Y. Rogozhin, editors, *Machines, Computations, and Universality (MCU)*, volume 2055 of *LNCS*, pages 1–10, Chişinău, Moldova, May 2001. Springer.
3. J. Cocke and M. Minsky. Universality of tag systems with $P = 2$. *Journal of the ACM*, 11(1):15–20, Jan. 1964.
4. M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.
5. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to parallel computation: P-completeness theory.* Oxford university Press, Oxford, 1995.

6. G. Hermann. The uniform halting problem for generalized one state Turing machines. In *Proceedings, Ninth Annual Symposium on Switching and Automata Theory (FOCS)*, pages 368–372, Schenectady, New York, Oct. 1968. IEEE.
7. J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1979.
8. M. Kudlek. Small deterministic Turing machines. *Theoretical Computer Science*, 168(2):241–255, Nov. 1996.
9. M. Kudlek and Y. Rogozhin. A universal Turing machine with 3 states and 9 symbols. In W. Kuich, G. Rozenberg, and A. Salomaa, editors, *Developments in Language Theory (DLT) 2001*, volume 2295 of *LNCS*, pages 311–318, 2002.
10. M. Margenstern. Frontier between decidability and undecidability: A survey. *Theoretical Computer Science*, 231(2):217–251, Jan. 2000.
11. M. Margenstern and L. Pavlotskaya. On the optimal number of instructions for universality of Turing machines connected with a finite automaton. *International Journal of Algebra and Computation*, 13(2):133–202, Apr. 2003.
12. P. Michel. Small Turing machines and the generalized busy beaver competition. *Theoretical Computer Science*, 326:45–56, 2004.
13. M. Minsky. A 6-symbol 7-state universal Turing machines. Technical Report 54-G-027, MIT, Aug. 1960.
14. M. Minsky. Size and structure of universal Turing machines using tag systems. In *Recursive Function Theory, Symp. in Pure Math.*, volume 5, pages 229–238, 1962.
15. T. Neary. *Small universal Turing machines*. PhD thesis, Department of Computer Science, National University of Ireland, Maynooth, 2008.
16. T. Neary and D. Woods. P-completeness of cellular automaton Rule 110. In M. Bugliesi et al., editor, *International Colloquium on Automata Languages and Programing 2006, (ICALP) Part I*, volume 4051 of *LNCS*, pages 132–143, Venice, July 2006. Springer.
17. T. Neary and D. Woods. Four small universal Turing machines. *Fundamenta Informaticae*, 91(1):123–144, 2009.
18. L. Pavlotskaya. Solvability of the halting problem for certain classes of Turing machines. *Mathematical Notes (Springer)*, 13(6):537–541, June 1973.
19. L. Pavlotskaya. Dostatochnye uslovija razreshimosti problemy ostanovki dlja mashin T'juring. *Problemi kibernetiki*, pages 91–118, 1978. (In Russian).
20. L. Priese. Towards a precise characterization of the complexity of universal and non-universal Turing machines. *Siam journal of Computing*, 8(4):508–523, 1979.
21. Y. Rogozhin. Small universal Turing machines. *Theoretical Computer Science*, 168(2):215–240, Nov. 1996.
22. C. E. Shannon. A universal Turing machine with two internal states. *Automata Studies, Annals of Mathematics Studies*, 34:157–165, 1956.
23. S. Watanabe. 5-symbol 8-state and 5-symbol 6-state universal Turing machines. *Journal of ACM*, 8(4):476–483, Oct. 1961.
24. S. Watanabe. 4-symbol 5-state universal Turing machine. *Information Processing Society of Japan Magazine*, 13(9):588–592, 1972.
25. S. Wolfram. *A new kind of science*. Wolfram Media, 2002.
26. D. Woods and T. Neary. On the time complexity of 2-tag systems and small universal Turing machines. In *47$^{th}$ Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 439–448, Berkeley, California, Oct. 2006. IEEE.
27. D. Woods and T. Neary. The complexity of small universal Turing machines: A survey. *Theoretical Computer Science*, 410(4–5):443–450, Feb. 2009.
28. D. Woods and T. Neary. Small semi-weakly universal Turing machines. *Fundamenta Informaticae*, 91(1):179–195, 2009.