The Complexity of Small Universal Turing Machines: A Survey*

Turlough Neary[1] and Damien Woods[2]

[1] School of Computer Science & Informatics, University College Dublin, Ireland
turlough.neary@ucd.ie
[2] Division of Engineering & Applied Science, California Institute of Technology, Pasadena, CA 91125, USA
woods@caltech.edu

Abstract. We survey some work concerned with small universal Turing machines, cellular automata, tag systems, and other simple models of computation. For example it has been an open question for some time as to whether the smallest known universal Turing machines of Minsky, Rogozhin, Baiocchi and Kudlek are efficient (polynomial time) simulators of Turing machines. These are some of the most intuitively simple computational devices and previously the best known simulations were exponentially slow. We discuss recent work that shows that these machines are indeed efficient simulators. In addition, another related result shows that Rule 110, a well-known elementary cellular automaton, is efficiently universal. We also discuss some old and new universal program size results, including the smallest known universal Turing machines. We finish the survey with results on generalised and restricted Turing machine models including machines with a periodic background on the tape (instead of a blank symbol), multiple tapes, multiple dimensions, and machines that never write to their tape. We then discuss some ideas for future work.

1 Introduction

In this survey we explore results related to the time and program size complexity of universal Turing machines, and other models of computation. We also discuss results for variants on the Turing machine model to give an idea of the many strands of work in the area. Of course the choice of topics is incomplete and reflects the authors' interests, and there are other related surveys that may interest the reader [32,38,41,57].

In 1956 Shannon [95] considered the question of finding the smallest possible universal Turing machine [99], where size is the number of states and symbols.

* This paper is extended and updated from [110]. T. Neary is supported by Science Foundation Ireland, Grant Number 09/RFP/CMS2212. D. Woods is supported by National Science Foundation Grant 0832824, the Molecular Programming Project. We thank Astrid Haberleitner for her tireless work in translating a number of highly technical papers from German to English, and Beverley Henley for her support.

M. Bieliková et al. (Eds.): SOFSEM 2012, LNCS 7147, pp. 385–405, 2012.

In the early Sixties, Minsky and Watanabe had a running competition to see who could find the smallest universal Turing machine [51,54,103,104]. Early attempts [23,104] gave small universal Turing machines that efficiently (in polynomial time) simulated Turing machines. In 1962, Minsky [54] found a small 7-state, 4-symbol universal machine. Minsky's machine worked by simulating 2-tag systems, which were shown to be universal by Cocke and Minsky [8,55]. Rogozhin [88] extended Minsky's technique of 2-tag simulation and found small machines with a number of state-symbol pairs. Subsequently, some of Rogozhin's machines were reduced in size or improved by Robinson [86,91], Kudlek and Rogozhin [27], and Baiocchi [4]. All of the smallest known 2-tag simulators are plotted as circles in Figure 1. Also, Table 1 lists a number of these machines.

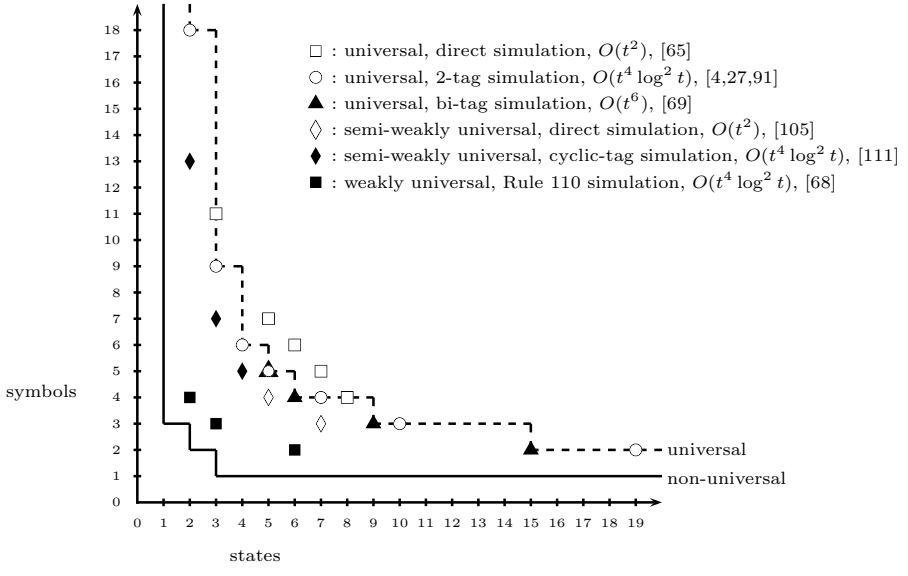

Fig. 1. State-symbol plot of small universal Turing machines. The type of simulation is given for each group of machines. Simulation time overheads are given in terms of simulating a single-tape deterministic Turing machine that runs in time t.

Unfortunately, Cocke and Minsky's 2-tag simulation of Turing machines was exponentially slow. The exponential slowdown was essentially caused by the use of a unary encoding of Turing machine tape contents. Therefore, for many years it was entirely plausible that there was an exponential trade-off between program size complexity on the one hand, and time/space complexity on the other: the smallest universal Turing machines seemed to be exponentially slow.

Figure 1 shows a non-universal curve. This curve is a lower bound that gives the state-symbol pairs for which it is known that the halting problem is decidable. The 1-symbol case is trivial and Shannon [95] claimed that 1-state Turing machines are non-universal. However, both Fischer [12] and Nozaki [70] noted that Shannon's definition of universal Turing machine is too strict and so his proof is not sufficiently general. Later, the 1-state case was shown by Hermann [19].

Table 1. Small standard universal Turing machines, ordered by date and then by state-symbol product. If there are multiple machines with the same state-symbol pair, the machine with the smallest number of instructions is denoted *.

states	symbols	state-symbol product	author
m	2	$2m$	Shannon [95]
2	n	$2n$	Shannon [95]
12	6	72	Takahashi [98] (mentioned in [104])
10	6	60	Ikeno [23] (also appears in [51])
8	6	48	Watanabe [103] (mentioned in [54])
7	6	42	Minsky [51]
8	5	40	Watanabe [104]
9	4	36	Tritter (mentioned in [54])
25	2	50	Minsky [55]
6	6	36	Minsky [54]
7	4	28	Minsky [54]
24	2	48	Rogozhin [87,88,91]
2	21	42	Rogozhin [87,88]
11	3	33	Rogozhin [87,88]
3	10	30	Rogozhin [87,88]
7	4	28	Rogozhin [87,88,91]
5	5	25	Rogozhin [87,88,91]
4	6	24	Rogozhin [87,88,91]
2	18	36	Rogozhin [91]
10	3	30	Rogozhin [89,91]
3	10	30	Rogozhin [90,91]*
22	2	44	Rogozhin [92]
19	2	38	Baiocchi [4]
7	4	28	Baiocchi [4]*
3	9	27	Kudlek & Rogozhin [27]
18	2	36	Neary & Woods [66]
9	3	27	Neary & Woods [69]
5	5	25	Neary & Woods [69]*
6	4	24	Neary & Woods [69]
15	2	30	Neary & Woods [69]

Pavlotskaya [75] and, via another method, Kudlek [26] have shown that there are no universal 2-state, 2-symbol machines, where one transition rule is reserved for halting. Pavlotskaya [77] has also shown that there are no universal 3-state, 2-symbol machines, and also claimed [75], without publishing a proof, that there are no universal machines for the 2-state, 3-symbol case. Again, both of these cases assume that a transition rule is reserved for halting.

2 Time and Size Efficiency of Universal Machines

As mentioned above, some of the very earliest small Turing machines were polynomial time simulators. Subsequently, attention turned to the smaller, but exponentially slower, 2-tag simulators given by Minsky, Rogozhin and others.

Recently [65] we have given small machines that are efficient polynomial time simulators. More precisely, if M is a deterministic single-tape Turing machine that runs in time t and space s, then there are machines, with state-symbol pairs given by the squares in Figure 1, that directly simulate M in polynomial time $O(t^2)$ and linear space $O(s)$. These machines define a $O(t^2)$ curve. They are currently the smallest known universal Turing machines that simulate Turing machines in $O(t^2)$ time. Their $O(s)$ space usage is also extremely efficient, more efficient than the other machines in Figure 1, all of which use space that is up to square root of their simulation time.

Despite the existence of these efficient $O(t^2)$ simulators, it still remained the case that the smallest universal machines were exponentially slow. However, we have recently shown that the smallest machines are in fact efficient simulators of Turing machines, by showing that 2-tag systems are efficient [108]. Tag systems are one of a number of rewriting systems invented in the 1920s by Post, although published somewhat later [79]. Post wanted to prove the decidability of various properties of tag systems, but found that even very simple examples had extremely complicated behaviour. Forty years later, Minsky showed that tag systems [53] are in fact computationally universal, and then Cocke and Minsky [8,55] showed universality for a particularly simple form called 2-tag systems. Minsky [54,55] saw that one could find very small universal Turing machines by simulating 2-tag systems, and since then 2-tag systems have been at the core of many results in the field.

A 2-tag system acts on a dataword, which is a string of symbols taken from a finite alphabet Σ. There is a fixed set of rules $R : \Sigma \to \Sigma^*$. In a single timestep, the leftmost symbol σ_j of the dataword is read, if there is a rule $\sigma_j \to \alpha_j$ then the string α_j is appended to the right of the dataword and the leftmost *two* dataword symbols are deleted. This process is iterated until a suitable halting condition is reached (i.e. there is no rule for the read symbol, the dataword has length less than 2, or the 2-tag system enters a repeating loop). Part of the reason why it was presumed that 2-tag systems were exponentially slow is that it is not obvious how to locate a specific symbol based solely on its position relative to other symbols in the dataword (one might want to do this to simulate the local action of a Turing machine tape head). The main result of [108] uses an algorithm that solves this problem, and does so efficiently.

More precisely, given a deterministic single-tape Turing machine M that runs in time t, there is a 2-tag system that simulates M and runs in polynomial time $O(t^4 \log^2 t)$. The small machines of Minsky, Rogozhin, and others have a quadratic time overhead when simulating 2-tag systems, hence by the result in [108] they simulate Turing machines in time $O(t^8 \log^4 t)$. It turns out that the time overhead can be improved [63] to $O(t^4 \log^2 t)$, giving the $O(t^4 \log^2 t)$ time overhead for the machines shown in Figure 1 as hollow circles. Thus, there is currently little evidence for the claim of an exponential trade-off between program size complexity, and time/space complexity.

From the point of view of program size, Neary and Woods [63,69] have recently given four Turing machines that are presently the smallest known (standard)

machines with 2, 3, 4 and 5 symbols. The 5-symbol machine improves on the 5-symbol machine of Rogozhin [91] by one transition rule. The remainder of these machines improve on the 2- and 4-symbol machines of Baiocchi [4], and the 3-symbol machine of Rogozhin [91]. These small machines simulate Turing machines in polynomial time $O(t^6)$ and are illustrated as triangles in Figure 1. They were proven universal via simulation of our universal variant of tag systems called *bi-tag systems* [69]. Bi-tag systems are essentially 1-tag systems (and so they read and delete one symbol per timestep) augmented with additional context sensitive rules that read, and delete, two symbols per timestep. Bi-tag systems are a restriction of Post's normal systems [79]. On the one hand bi-tag systems are universal, while on the other hand they are sufficiently 'simple' to be simulated by such small machines.

Exponentially improving the time efficiency of 2-tag systems has implications for a number of models of computation, besides small universal Turing machines. Following our result, the simulation efficiency of many biologically inspired models of computation, including neural networks, H systems and P systems, has been improved from exponential to polynomial. For example, Siegelmann and Margenstern [96] give a neural network that uses only nine high-order neurons to simulate 2-tag systems. Taking each synchronous update of the nine neurons as a single parallel timestep, their neural network simulates 2-tag systems in linear time. They note that "tag systems suffer a significant slow-down ... and thus our result proves only Turing universality and should not be interpreted complexity-wise as a Turing equivalent." Now we know that their neural network is in fact efficiently universal. Rogozhin and Verlan [93] give a tissue P system with eight rules that simulates 2-tag systems in linear time, and thus we have improved its simulation time overhead from exponential to polynomial. This system uses splicing rules (from H systems) with membranes (from P systems) and is non-deterministic. Harju and Margenstern [18] gave an extended H-system with 280 rules that generates recursively enumerable sets using Rogozhin's 7-state, 4-symbol universal Turing machine. Using our result from 2-tag systems, the time efficiency of their construction is improved from exponential to polynomial, with a possible small constant increase in the number of rules. The efficiency of Hooper's [22] small 2-tape universal Turing machine is also improved from exponential to polynomial, as is Rothemund's [94] restriction enzyme implementation of Minsky's 7-state, 4-symbol UTM. The technique of simulation via 2-tag systems is at the core of many of the universality proofs in Margenstern's survey [41]. Our work exponentially improves the time overheads in these simulations, such as Lindgren and Nordahl's cellular automata [31], Margenstern's non-erasing Turing machines [34,36], and Robinson's tiling [85].

3 Non-standard Universal Turing Machines: Time Efficiency and Program Size

So far we have been discussing results for universal Turing machines that have one tape, one tape head, and are deterministic (we often refer to this setup as

the *standard* model). Of course one can consider results for other variants of the model. There are many generalised models, for example allowing multiple tapes, multiple dimensions, or even coupling the Turing machine with a finite automaton. Restricted models include non-printing, non-erasing and reversible Turing machines, and machines with restricted instructions. In this section we explore program size and time complexity results for a number of generalised and restricted models. Table 2 contains program size results for a number of such non-standard machines.

Table 2. Small non-standard universal Turing machines. Semi-weak machines are denoted by †, weak machines by ‡, machines coupled with a finite automaton by ⋆, and a machine with 2 tape heads by △.

states	symbols	dimensions	tape	author
15	2	1	3	Moore [60]†
6	5	1	1	Watanabe [104]†
1	2	1	4	Hooper [21,22]†
2	3	1	2	Hooper [21,22]
7	3	1	1	Watanabe (mentioned in [105,70])†
5	4	1	1	Watanabe [105]†
8	4	2	1	Wagner [100]
2	7	2	1	Ottmann [73]‡
10	2	2	1	Ottmann [74,25]‡
6	3	2	1	Ottmann [74,25]‡
4	4	2	1	Ottmann [74,25]‡
2	6	2	1	Kleine-Büning & Ottmann [25]‡
2	5	2	1	Kleine-Büning & Ottmann [25]‡
2	3	2	1	Kleine-Büning & Ottmann [25]‡
1	7	3	1	Kleine-Büning & Ottmann [25]‡
4	5	2	1	Kleine-Büning & Ottmann [25]
3	6	2	1	Kleine-Büning & Ottmann [25]
10	2	2	1	Kleine-Büning [24]
2	5	2	1	Kleine-Büning [24]
2	4	2	1	Priese [82]
4	2	2	1	Gajardo et al. [15]
2	2	2	1	Priese [82]△
2	5	1	1	Margenstern & Pavlotskaya [45]⋆
4	7	1	1	Pavlotskaya [78]⋆
2	3	1	1	Margenstern & Pavlotskaya [46]⋆
7	2	1	1	Eppstein (published by Cook [9])‡
4	3	1	1	Cook [9] & Wolfram [107]‡
3	4	1	1	Cook [9] & Wolfram [107]‡
2	5	1	1	Cook [9] & Wolfram [107]‡
6	2	1	1	Neary & Woods [68]‡
3	3	1	1	Neary & Woods [68]‡
2	4	1	1	Neary & Woods [68]‡
3	7	1	1	Woods & Neary [111]†
4	5	1	1	Woods & Neary [111]†
2	13	1	1	Woods & Neary [111]†

3.1 Weak Universality and Rule 110

An interesting generalisation occurs when we stick to the standard conventions, but we allow the blank portion of the tape to contain a word, that is constant (independent of the input), and is repeated infinitely often in one direction, say to the left of the input. We say that such Turing machines are *semi-weakly universal*. Some of the earliest small universal Turing machines were semi-weak [104,105]. Sometimes another word is also repeated infinitely often to the right. Universal machines that use this setup are called *weakly universal* [43].

It is not difficult to see how this generalisation can help to reduce program size. For example, it is typical of small universal Turing machine simulations that the program being simulated is stored on the tape. When reading an instruction we often mark certain symbols. At a later time we then restore marked symbols to their original values. If the simulated program is repeated infinitely often, say to the left of the input, things may be much easier as we can simply skip the 'restore' phase of our algorithm and access a new copy of the program when simulating the next instruction, thus reducing the universal program's size.

This was the strategy used by Watanabe [104,105] to find the semi-weak, direct Turing machine simulators shown as hollow diamonds in Figure 1. Recently [111] we have given three new semi-weakly universal machines and these are shown as solid diamonds in Figure 1. These machines simulate cyclic tag systems [9]. It is interesting to note that two of our machines are symmetric with those of Watanabe (around the line where states = symbols), despite the fact that we use a different simulation technique. Our 4-state, 5-symbol machine has only 17 transition rules, making it the smallest known semi-weakly universal machine (Watanabe's 5-state, 4-symbol machine has 18 transition rules, and his 7-state, 3-symbol machine has 21 rules [105])[1]. The time overhead for these machines is polynomial. More precisely, if M is a single-tape deterministic Turing machine that runs in time t, then M is simulated by either of our semi-weak machines in time $O(t^4 \log^2 t)$. Watanabe's semi-weak machines also ran in polynomial time, with a very efficient time overhead of $O(t^2)$.

Cook, Eppstein, and Wolfram [9,107] gave weakly universal Turing machines that were significantly smaller than the existing semi-weak machines. These were improved upon by Neary and Woods [68] to give the smallest known weakly universal machines. In (states, symbols) notation their sizes are $(2, 4)$, $(3, 3)$ and $(6, 2)$, and they are illustrated in Figure 1. These machines work by simulating Rule 110, a very simple kind of cellular automaton. Rule 110 is an elementary cellular automaton, which means that it is a one-dimensional, nearest neighbour, binary cellular automaton [106]. More precisely, 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 $t + 1$ the value of cell $p_{i,t+1} = F(p_{i-1,t}, p_{i,t}, p_{i+1,t})$ is given by the synchronous local update function F

[1] Watanabe mentions that he found a $(7, 3)$ universal machine with 21 transition rules in reference [105]. We have not found the details of this machine, however the most reasonable inference from the literature is that it is semi-weakly universal.

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

Rule 110 was shown to be universal via an impressive and detailed simulation of cyclic tag systems, the result is stated and described in [107] and the full proof is given in [9]. In the proof, the Rule 110 instance has a special (constant) word repeated infinitely to the left of the input, and another to the right. Rule 110 has a very simple update rule which facilitates the writing of very small weak Turing machines to simulate it.

As noted, Rule 110 was shown to be universal by simulating cyclic tag systems, which in turn simulate 2-tag systems. The chain of simulations included the exponentially slow 2-tag algorithm of Cocke and Minsky, thus Rule 110, and the weakly universal machines that simulate it, were exponentially slow. In a recent paper [64] we have improved their simulation time overhead to polynomial by showing that cyclic tag systems are efficient simulators of Turing machines. In doing so, we solved what Cook [10] has called the "geometry problem of cyclic-state tape processors." The difficult in overcoming this problem is that there is no obvious way for the system to efficiently determine which symbols or objects are adjacent to each other. Previous works used unary encodings as it was not obvious how to determine the relative positions of adjacent digits in a sequence. Our main result was in providing an efficient solution to this problem.

Our result has interesting implications for Rule 110. For example, given an initial configuration of Rule 110, and a value t in unary, predicting t timesteps of a Rule 110 computation is P-complete. Therefore, unless P = NC, which is widely believed to be false, we cannot hope to quickly (in polylogarithmic time) predict the evolution of this simple cellular automaton even if we have a polynomial amount of parallel hardware. Rule 110 is the simplest (one-dimensional, nearest neighbour) cellular automaton that has been shown to have a P-complete prediction problem. In particular, Ollinger's [71] intrinsic universality result already shows that prediction for one dimensional nearest neighbour cellular automata is P-complete for 6 states (later improved to 4 states by Richard and Ollinger [84,72]), and our result improves this to 2 states. The question of whether Rule 110 prediction is P-complete has been asked, directly or indirectly, in a number of previous works (for example [2,58,59]).

It is currently unknown whether all of the lower bounds in Figure 1 hold for weak machines. For example, the non-universality results of Pavlotskaya were proven for the case where one transition rule is reserved for halting, however the smallest weak machines do not halt.

3.2 Other Non-standard Universal Turing Machines

Weakness has not been the only generalisation on the standard model in the search for ever smaller universal machines. We give some notable examples here, many others are to be found in Table 2.

Before Shannon's famous paper, Moore [60] observed that 2-symbol machines were universal as any Turing machine could be converted into a 2-symbol machine by the (now) usual encoding. In the same paper Moore used this observation to give a universal 3-tape machine with 15 states and 2 symbols. Moore's machine uses only 57 instructions, each instruction being a sextuple that either moves one of its tape heads or prints a single symbol to one of its tapes. One of the tapes in Moore's 3-tape machine is circular and contains the simulated program, therefore his machine also operates correctly if the circular tape is replaced with a one-way infinite tape with a periodic background (i.e. semi-weak). Moore's result has been largely ignored in the literature despite being the first published small universal Turing machine. Interestingly, Moore's paper cites unpublished work by Shannon on the universality of non-erasing machines.

Hooper [21,22] gave universal machines with 2 states, 3 symbols and 2 tapes, and with 1 state, 2 symbols and 4 tapes. One of the tapes in Hooper's 4-tape machine is circular and contains the simulated program, and so could be replaced by a one-way infinite tape with a periodic background (i.e. semi-weak). Priese [82] gave a 2-state, 4-symbol machine with a 2-dimensional tape, and a 2-state, 2-symbol machine with 2 tape heads and a 2-dimensional tape. Margenstern and Pavlotskaya [46,45] gave a 2-state, 3-symbol Turing machine that uses only 5 instructions and is universal when coupled with a finite automaton. They also showed that the halting problem is decidable for such machines with 4 instructions [46].

3.3 Restricted Universal Turing Machines

If we suitably restrict the standard Turing machine model the problem of finding universal machines with small state-symbol products becomes more difficult. Over the years, a number of authors have looked at non-erasing Turing machines, that is machines that are permitted to overwrite blank symbols only. Moore [60] mentions that Shannon had proved that such non-erasing Turing machines simulate arbitrary Turing machines, however Shannon's work was never published. Shortly after, Shannon published a proof that 2-symbol Turing machines are universal, and Wang [101] proved that 2-symbol non-erasing Turing machines are universal. Later, Minsky proved the same result as Wang, but using the technique of simulation via non-writing Turing machines, yet another (universal) restriction [53].

Margenstern has examined the universality of 2-symbol Turing machines for a number of different restrictions. One such restriction is the number of colours of a machine, defined as the number of distinct triples (α, D, δ), where α is the read symbol, D is the move direction, and δ is the write symbol of a transition rule. Pavlotskaya [75,76] has shown that there are standard universal Turing machines with 3 colours and no standard universal Turing machines with 2 colours. Margenstern [34] has shown that there are non-erasing universal Turing machines with 5 colours and no non-erasing universal Turing machines with 4 colours.

Laterality number is another property examined by Margenstern. The laterality number of a Turing machine is defined as the minimum of the number of left move instructions and the number of right move instructions. Margenstern and Pavlotskaya [75,44] have shown that there are universal Turing machines with laterality number 2 and no universal machines with laterality number 1. Margenstern [36,39] has shown that there are universal non-erasing Turing machines with laterality number 3 and no universal non-erasing machines with laterality number 2. For more on these results see [33,34,35,36,37,42].

Fischer [12] gives a number of universality results for Turing machines that use restricted forms of transition rules. In one result he proves that 3-state Post machines are universal (Post machines [80] are like Turing machines, except that in a single timestep they can move or write, but not both). Interestingly, Aanderaa and Fischer [1] show that the halting problem for 2-state Post machines is decidable.

Bennett [5] has shown that 3-tape reversible Turing machines are universal. Morita and others have since shown universality results for reversible Turing machines with 1 tape and 2 symbols [61], and 17 states and 5 symbols [62].

3.4 Universal Turing Machines with Multidimensional Tapes: Time Efficiency and Program Size

During the 1970s a number of authors [82,25,100] were interested in finding small universal Turing machines with multidimensional tapes. The machines of these authors have not, to our knowledge, been analysed from the perspective of time/space complexity. We discuss this topic here.

Lutz Priese [82] gives a 2D machine with 2 states and 4 symbols that is universal on finite initial conditions (i.e. all except a finite number of symbols are initially blank), and another 2D machine with 2 states, 2 symbols and 2 tape heads that is derived from this 4-symbol machine. Priese's machines simulate counter machines (also called register machines), via a sequence of reductions. Given a counter machine that runs in time τ, Priese's machines simulate its computation in time $O(\tau^2)$ and space $O(\tau)$. Due to the unary encoding used by counter machines [13], both of Priese's machines simulate Turing machines with an exponential time overhead. Priese's machines do not end their computation in the conventional manner of halting on a state-symbol pair that has no transition rule: instead there is a choice, via the initial input encoding, of ending a computation either by entering a sequence of 6 repeating configurations or by halting when an attempt is made to move off the edge of the 2D tape.

Langton's ant [29] is usually described as an ant that lives on a 2D grid of binary-valued cells. The ant chooses which adjacent cell to move to based on (a) the current cell's binary value and (b) the ant's current orientation. The ant flips the current cell's bit as it moves away. So Langton's ant is a 2D Turing machine with 2 symbols and 4 states (North, South, East and West). Gajardo et al. [15] showed that predicting the behaviour of the ant is P-hard, by simulating

Boolean circuits in polynomial time. By then showing how the ant can simulate an infinite sized circuit (with a simple repeating structure), which in turn can simulate the space-time diagram of a cellular automata (CA), they prove that Langton's ant is weakly universal in 2D.

It is worth pausing to describe a form of weak universality in 2D, where the tape has a background that is ultimately periodic in both dimensions of single quadrant. A one-way infinite sequence is ultimately periodic [14] if it is of the form $s_1 s_2^\omega$ where $s_2^\omega = s^2 s^2 s^2 \ldots$, and s_1 and s_2 are finite sequences. We say that a $\mathbb{N} \times \mathbb{N}$ pattern is ultimately periodic in the x direction if for each $y \in \mathbb{N}$ the infinite sequence of symbols at the coordinates $(0, y), (1, y), (2, y), \ldots$ is ultimately periodic. This is defined analogously for the y direction.

Kleine-Büning and Ottmann [25] give universal Turing machines which have a single multidimensional tape, a number of which are weakly universal. Remarkably, their 2D, 2-state, 3-symbol machine does not even print to the tape! The two counter values of a simulated 2-counter machine are encoded by the (x, y) position of the tape head on the 2D tape. Testing for zero amounts to detecting one of the axes. It is well-known that 2-counter machines are universal [56]. However, using known algorithms, 2-counter machines suffer from a doubly-exponential slowdown when simulating Turing machines [97], and so the 2-state, 3-symbol machine of Kleine-Büning and Ottmann also suffers from a doubly-exponential slowdown when simulating Turing machines. We give a brief overview of this machine's computation.

The 2D tape uses only the upper-right quadrant of the plane and so each tape cell may be indexed by a coordinate of the form $(x, y) \in \mathbb{N} \times \mathbb{N}$. The quadrant is filled using 4, infinitely repeated, finite square blocks (of tape symbols) which we will call A, B, C, and D. The infinite pattern on the 2D tape given by the arrangement of these blocks is ultimately periodic in both the x and y directions. Each block is of size $O(r^2)$ where r is the number of instructions in the 2-counter machine being simulated. The block at the origin of the quadrant is of type A. Types B and C are repeated along along the x-axis and y-axis, respectively, and the remainder of the quadrant is tiled by blocks of type D. Each block encodes the entire program of the 2-counter machine being simulated. The current counter machine instruction being simulated is given by the position of the tape head within a block. If the counters have values x_1 and y_1 respectively, then the tape head will be in the x_1^{th} block from the y-axis and the y_1^{th} block from the x-axis. The blocks contain specially defined paths that the tape head follows to (a) arrive at the next counter machine instruction and (b) move to one of the adjacent blocks if a change in the value of a counter is being simulated. A, B and C blocks lie along the axes and so are used to simulate any instruction where one or more counters have value zero, and in particular they contain special paths that simulate a positive test for zero.

Kleine-Büning and Ottmann adapt their technique to give a non-printing 1-state, 7-symbol universal machine with a 3D tape. Only 2 planes in the third dimension are used, giving tape cells that are indexed by coordinates (x, y, z),

where $x, y \in \mathbb{N}$ and $z \in \{0, 1\}$. The pattern defined by the symbols on each of the infinite 2D planes given by $(x, y, 0)$ and $(x, y, 1)$ is ultimately periodic in both the x and y directions. The technique used to simulate 2-counter machines by the 1-state, 7-symbol machine is, in essence, the same as the technique use by the 2-state, 3-symbol machine. The 2D machine uses 2 states to remember which path it is following when two different paths cross (the tape head follows paths that encode instructions of the counter machine being simulated). With the introduction of a third dimension it is no longer necessary for paths to cross and so it is possible to give a universal Turing machine with only 1 state. Finally, we note that an immediate corollary of this machine's design is the existence of a non-halting universal machine with only 6 symbols, as the only purpose of one of the 7 symbols is to provide an undefined transition rule for halting.

It is a fairly straightforward matter to show that for each Turing machine with a single, ultimately periodic, 2D tape and no print instructions there is a 2-counter machine that simulates it in linear time. It immediately follows that improvement on the doubly-exponential time overhead when simulating Turing machines with such non-printing 2D machines is not possible unless such an improvement is also possible for 2-counter machines. Thus, it could be interesting to see if the simulation time overhead for such machines can be reduced to singly-exponential when a slightly more complicated background is permitted on the tape.

Wagner [100] shows that the halting problem for Turing machines with a single kD tape ($k \in \mathbb{N}$), 2 symbols and 2 states is decidable[2]. Specifically, he shows that if such machines halt then they do so in space $O(n)$, where n is input length. It is not difficult to give relevant decidability results (such as predicating looping or halting) for machines with a single 1D tape and non-printing instructions, even when an ultimately periodic background is permitted. Regarding kD machines, it can be shown for some classes of these machines that only weaker forms of universality are possible. For the case of kD non-printing machines, it is not difficult to give relevant decidability results when the initial tape contains only a finite number of non-blank symbols. Herman has shown that the halting problem is decidable for 1-state kD printing machines when all but a finite number of tape cells are blank at the start of each computation [20].

Though lacking in formal rigour, a comparison between the three 2D machines we discussed in this section poses some interesting questions about the possible trade-offs for different 2D models. For example, out of the three machines the 2-state, 3-symbol weak machine has the smallest state-symbol product, is the only non-writing machine, and the only machine that can halt. The 2-state, 4-symbol machine of Priese is the only machine of the three that does not use a periodic (weak) encoding, and the 4-state, 2-symbol machine of Gajardo et al. (Langton's ant) is the only machine of the three that simulates Turing machines in polynomial time. The best we can hope for with non-printing 2D machines is

[2] Machines using Wagner's definition end their computation with a simple loop: repeatedly executing a special transition rule that does not change the configuration. This is equivalent to executing a halting transition rule.

a singly exponential time overhead, but achieving even this bound would seem to be very tricky. It is interesting to note that the only non-weak 2D machine of the three, that of Priese, has an exponential time overhead when simulating Turing machines. This is not the case for the smallest non-weak 1D machines. It begs the question, is there a non-weak 2D machine with the same number of states and symbols as Priese's machine that is universal with a polynomial time overhead?

3.5 Termination of a Computation

As we hope has been made clear so far, it is vitally important to clearly specify the computational model one is using when trying to find small universal programs or give lower bounds on universal program size. In the absence of a clear model description and matching lower bounds, one can never claim to have found the "smallest" universal program. Throughout this work we have described results on upper bounds and lower bounds on universal program size and we have described how both change when the model definition changes. In this section we focus on one such issue: computation termination.

A number of authors have given universal Turing machines where successful computations do not end in a halt state. Many of the machines given in Table 2 are non-halting. What about the problem of proving relevant non-universality results for these models? Such non-universality results are not achievable by proving the halting problem decidable. Before we attempt such an endeavour we must agree on a clear definition of universal Turing machine. For example, instead of specifying the end of a computation by a single halting (or terminal) configuration, a computation could end with a specific sequence of configurations. We refer to this as a terminal configuration sequence. The output of the simulated Turing machine is retrieved by applying a recursive decoding function to the entire computation (also a configuration sequence). There are many ways to define terminal configuration sequence, some examples are:

- a configuration sequence that goes through a specified sequence of states,
- a configuration sequence that contains two identical configurations,
- a configuration that contains a specific subword.

Given a definition of a terminal configuration sequence we may prove that the terminal sequence problem (will a machine execute a terminal configuration sequence) is decidable. This gives non-universal lower bounds that are relevant to universal machines that end their computation with such a sequence. However, this result may not hold as a proof of non-universality if we subsequently alter our definition of terminal configuration sequence. One more general approach is to prove that the terminal sequence problem for all possible terminal sequences, of a machine or set of machines, is decidable. In any case, it is important to specify these details when giving upper and lower bounds on program size.

4 Busy Beavers

Besides small universal Turing machines, one finds small, yet complicated, programs in the busy beaver literature. The term busy beaver was introduced by Rado [83] who put forward a game where the goal for a given $k \in \mathbb{N}$ is to find, out of all the k-state, 2-symbol Turing machines, the machine that prints the most 1s and then halts when started on a blank tape. The busy beaver function $\Sigma : \mathbb{N} \to \mathbb{N}$ is then defined by letting $\Sigma(k)$ be the maximum number of 1's printed by any halting k-state, 2-symbol Turing machine. Busy beavers essentially adhere to the standard Turing machine model described in previous sections (one tape, one head, usual blank symbol, deterministic). It is known that $\Sigma(1) = 1$ (trivial), $\Sigma(2) = 4$ [83], $\Sigma(3) = 6$ [30], and $\Sigma(4) = 13$ [6]. However for 5 states or more the best we currently have are lower bounds. For example, Michel [50] cites $\Sigma(5) \geq 4098$ to Marxen and Buntrock [47], and $\Sigma(6) \geq 3.5 \times 10^{18267}$ to Pavel Kropitz. $S(k)$, the step-counting analogue of $\Sigma(k)$, is also considered. In fact, both Σ and S grow faster than any computable function [83]. Green [16] has given a lower bound on the growth of the function Σ.

The busy beaver problem has been generalised to machines with $\ell \geq 2$ symbols [7], where $\Sigma(k, \ell)$ is the largest number of non-zeros written by any k-state, ℓ-symbol Turing machine. It has been shown [7,28] that $\Sigma(2, 3) = 9$. Terry Ligocki and Shawn Ligocki have shown that $\Sigma(2, 4) \geq 2,050$ and $\Sigma(3, 3) \geq 374,676,383$, and have given lower bounds on a number of other state-symbol pairs. See Michel's survey [50] for more results.

Although finding busy beavers is somewhat orthogonal to the goal of finding small universal Turing machines, there are potential connections between the two fields. On the one hand, when designing small universal programs one often has to reuse instructions in many different contexts, something which busy beavers might also do, so perhaps small instruction sets from one field might be useful for the other. On the other hand, proving lower bounds on universal program size, and upper bounds on values for the busy beaver function, both involve hefty case analyses so once again techniques developed in one field could potentially be useful for the other. In particular, the search for busy beavers has produced small programs with very complicated behaviour, which lend weight to the idea that proving non-universality of such program classes might be difficult.

5 Further Work

There are many avenues for further work, here we highlight a few examples.

Applying computational complexity theory to the area of small universal Turing machines allows us to ask a number of questions that are more subtle than the usual questions about program size. As we move towards the origin in Figure 1, the universal machines have larger (but polynomial) time overheads. Can the time overheads in Figure 1 be further improved (lowered)? Can we prove lower bounds on the simulation time of machines with a given state-symbol pair? Proving non-trivial simulation time lower bounds seems like a difficult problem.

Such results could be used to prove that there is a polynomial trade-off between simulation time and universal program size.

As we move away from the origin, the non-universal machines seem to have more power. For example Kudlek's classification of 2-state, 2-symbol machines shows that the sets accepted by these machines are regular, with the exception of one context free language ($a^n b^n$). Can we hope to fully characterise the sets accepted by non-universal machines (e.g. in terms of complexity or automata theoretic classes) with given state-symbol pairs or other program restrictions?

When discussing the complexity of small machines the issue of encodings becomes very important. For example, when proving that the prediction problem for a small machine is P-complete [17], the relevant encodings should be in logspace, and this is the case for all of the polynomial time machines in Figure 1.

Of course there are many models of computation that we have not mentioned where researchers have focused on finding small universal programs. Post's [79] tag systems are an interesting example. Minsky [52,53] showed that tag systems are universal with deletion number 6. Cocke and Minsky lowered the deletion number to 2, by showing that 2-tag systems were universal. They used productions (appendants) of length at most 4. Wang [102] further lowered the production length to 3. Recently, De Mol [11] has given a lower bound by showing that the reachability (and thus halting) problems are decidable for 2-tag systems with 2 symbols; a problem which Post claimed [81] to have solved but never published. It would be interesting to find the smallest universal tag systems in terms of number of symbols, deletion length, and production length.

The space between the non-universal curve and the smallest non-weakly universal machines in Figure 1 contains some complicated beasts. These lend weight to the feeling that finding new lower bounds on universal program size is tricky. Most noteworthy are the weakly and semi-weakly universal machines discussed earlier. Table 2 highlights that the existence of general models that provably have less states and symbols than the standard universal machines can have (for example the machines with (states, symbols, dimensions, tapes) of (2,3,2,1) [25], (1,7,3,1) [25], and (1,2,1,4) [22]). Also of importance are the busy beavers [50] and small machines of Margenstern [40,41], Baiocchi [3], and Michel [48,49] that live in this region and simulate iterations of the $3x + 1$ problem and other Collatz-like functions. So it seems that there are plenty of animals yet to be tamed.

References

1. Aanderaa, S., Fischer, P.C.: The solvability of the halting problem for 2-state post machines. Journal of the Association for Computing Machinery 14(4), 677–682 (1967)
2. Aaronson, S.: Book review: A new kind of science. Quantum Information and Computation 2(5), 410–423 (2002)
3. Baiocchi, C.: 3N+1, UTM e tag-system. Technical Report Pubblicazione 98/38, Dipartimento di Matematico, Università di Roma (1998) (in Italian)

4. Baiocchi, C.: Three Small Universal Turing Machines. In: Margenstern, M., Rogozhin, Y. (eds.) MCU 2001. LNCS, vol. 2055, pp. 1–10. Springer, Heidelberg (2001)
5. Bennett, C.H.: Logical reversibility of computation. IBM Journal of Research and Development 17(6), 525–532 (1973)
6. Brady, A.H.: The determination of the value of Rado's noncomputable function $\Sigma(k)$ for four-state Turing machines. Mathematics of Computation 40(163), 647–665 (1983)
7. Brady, A.H.: The busy beaver game and the meaning of life. In: Herken, R. (ed.) The Universal Turing Machine: A Half-Century Survey, pp. 259–277. Oxford University Press (1988)
8. Cocke, J., Minsky, M.: Universality of tag systems with $P = 2$. Journal of the Association for Computing Machinery 11(1), 15–20 (1964)
9. Cook, M.: Universality in elementary cellular automata. Complex Systems 15(1), 1–40 (2004)
10. Cook, M.: A Concrete View of Rule 110 Computation. Electronic Proceedings in Theoretical Computer Science 1, 31–55 (2009)
11. De Mol, L.: Study of Limits of Solvability in Tag Systems. In: Durand-Lose, J., Margenstern, M. (eds.) MCU 2007. LNCS, vol. 4664, pp. 170–181. Springer, Heidelberg (2007)
12. Fischer, P.C.: On formalisms for Turing machines. Journal of the Association for Computing Machinery 12(4), 570–580 (1965)
13. Fischer, P.C., Meyer, A.R., Rosenberg, A.L.: Counter machines and counter languages. Mathematical Systems Theory 2(3), 265–283 (1968)
14. Friedman, J.: A decision procedure for computations of finite automata. Journal of the ACM 9(3), 315–323 (1962)
15. Gajardo, A., Moreira, A., Goles, E.: Complexity of Langton's ant. Discrete Applied Mathematics 117, 41–50 (2002)
16. Green, M.W.: A lower bound on Rado's sigma function for binary Turing machines. In: Proceedings of the 5th IEEE Annual Symposium on Switching Circuit Theory and Logical Design, pp. 91–94 (1964)
17. Greenlaw, R., Hoover, H.J., Ruzzo, W.L.: Limits to parallel computation: P-completeness theory. Oxford university Press, Oxford (1995)
18. Harju, T., Margenstern, M.: Splicing Systems for Universal Turing Machines. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 149–158. Springer, Heidelberg (2005)
19. Hermann, G.T.: The uniform halting problem for generalized one state Turing machines. In: Proceedings of the Ninth Annual Symposium on Switching and Automata Theory (FOCS), pp. 368–372. IEEE Computer Society Press, Schenectady (1968)
20. Hermann, G.T.: The halting problem of one state Turing machines with n-dimensional tape. Mathematical Logic Quarterly 14(7-12), 185–191 (1968)
21. Hooper, P.: Some small, multitape universal Turing machines. Technical report, Computation Laboratory, Harvard University, Cambridge, Massachusetts (1963)
22. Hooper, P.: Some small, multitape universal Turing machines. Information Sciences 1(2), 205–215 (1969)
23. Ikeno, N.: A 6-symbol 10-state universal Turing machine. In: Proceedings, Institute of Electrical Communications, Tokyo (1958)
24. Kleine-Büning, H.: Über probleme bei homogener Parkettierung von $\mathbb{Z} \times \mathbb{Z}$ durch Mealy-automaten bei normierter verwendung. PhD thesis, Institut für Mathematische Logik, Münster (1977)

25. Kleine-Büning, H., Ottmann, T.: Kleine universelle mehrdimensionale Turingmaschinen. Elektronische Informationsverarbeitung und Kybernetik 13(4-5), 179–201 (1977) (in German)
26. Kudlek, M.: Small deterministic Turing machines. Theoretical Computer Science 168(2), 241–255 (1996)
27. Kudlek, M., Rogozhin, Y.: A Universal Turing Machine with 3 States and 9 Symbols. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 311–318. Springer, Heidelberg (2002)
28. Lafitte, G., Papazian, C.: The fabric of small Turing machines. In: Computation and Logic in the Real World, Third Conference on Computability in Europe, CiE 2007, Local Proceedings, pp. 219–227 (2007)
29. Langton, C.: Studying artificial life with cellular automata. Physica D 2(1-3), 120–149 (1986)
30. Lin, S., Rado, T.: Computer studies of Turing machine problems. Journal of the ACM 12(2), 196–212 (1965)
31. Lindgren, K., Nordahl, M.G.: Universal computation in simple one-dimensional cellular automata. Complex Systems 4(3), 299–318 (1990)
32. Margenstern, M.: Surprising areas in the quest for small universal devices. Electronic Notes in Theoretical computer Science 225, 201–220 (2009)
33. Margenstern, M.: Sur la frontière entre machines de Turing á arrêt décidable et machines de Turing universelles. Technical Report 92-83, LITP Institut Blaise Pascal (1992)
34. Margenstern, M.: Non-erasing Turing Machines: A Frontier Between a Decidable Halting Problem and Universality. In: Ésik, Z. (ed.) FCT 1993. LNCS, vol. 710, pp. 375–385. Springer, Heidelberg (1993)
35. Margenstern, M.: Une machine de Turing universelle sur {0,1}, non-effaçante et à trois instructions gauches. Technical Report 94-08, LITP Institut Blaise Pascal (1994)
36. Margenstern, M.: Non-Erasing Turing Machines: A New Frontier Between a Decidable Halting Problem and Universality. In: Baeza-Yates, R.A., Poblete, P.V., Goles, E. (eds.) LATIN 1995. LNCS, vol. 911, pp. 386–397. Springer, Heidelberg (1995)
37. Margenstern, M.: Une machine de Turing universelle non-effaçante à exactement trois instructions gauches. C. R. Acad. Sci. Paris, Série I 320, 101–106 (1995)
38. Margenstern, M.: Decidability and Undecidability of the Halting Problem on Turing Machines, a Survey. In: Adian, S., Nerode, A. (eds.) LFCS 1997. LNCS, vol. 1234, pp. 226–236. Springer, Heidelberg (1997)
39. Margenstern, M.: The laterality problem for non-erasing Turing machines on $\{0, 1\}$ is completely solved. Theoretical Informatics and Applications 31(2), 159–204 (1997)
40. Margenstern, M.: Frontier between decidability and undecidability: a survey. In: Margenstern, M. (ed.) Machines, Computations and Universality (MCU), France, IUT, Metz., vol. 1, pp. 141–177 (1998)
41. Margenstern, M.: Frontier between decidability and undecidability: a survey. Theoretical Computer Science 231(2), 217–251 (2000)
42. Margenstern, M.: On quasi-unilateral universal Turing machines. Theoretical Computer Science 257(1-2), 153–166 (2001)
43. Margenstern, M.: An algorithm for building intrinsically universal cellular automata in hyperbolic spaces. In: Proceedings of the 2006 International Conference on Foundations of Computer Science (FCS), Las Vegas, NV, pp. 3–9. CSREA Press (2006)

44. Margenstern, M., Pavlotskaya, L.: Deux machines de Turing universelles á au plus deux instructions gauches. C. R. Acad. Sci. Paris, Série I 320, 1395–1400 (1995)
45. Margenstern, M., Pavlotskaya, L.: Vers ue nouvelle approche de l'universalité concernant les machines de Turing. Technical Report 95-58, LITP Institut Blaise Pascal (1995)
46. Margenstern, M., Pavlotskaya, L.: 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 (2003)
47. Marxen, H., Buntrock, J.: Attacking the Busy Beaver 5. Bulletin of the EATCS 40, 247–251 (1990)
48. Michel, P.: Busy beaver competition and Collatz-like problems. Archive Mathematical Logic 32(5), 351–367 (1993)
49. Michel, P.: Small Turing machines and generalized busy beaver competition. Theoretical Computer Science 326, 45–56 (2004)
50. Michel, P.: The busy beaver competition: a historical survey, arXiv:0906.3749v2 (2010), http://www.logique.jussieu.fr/~michel/ha.html
51. Minsky, M.: A 6-symbol 7-state universal Turing machines. Technical Report 54-G-027, MIT (1960)
52. Minsky, M.: Recursive unsolvability of Post's tag problem. Technical Report 54-G-023, Massachusetts Institute of Technology (1960)
53. Minsky, M.: Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. Annals of Mathematics 74(3), 437–455 (1961)
54. Minsky, M.: Size and structure of universal Turing machines using tag systems. In: Recursive Function Theory: Proceedings, Symposium in Pure Mathematics, Provelence, vol. 5, pp. 229–238 (1962)
55. Minsky, M.: Universality of (p=2) tag systems and a 4 symbol 7 state universal Turing machine. In: AIM-33, A.I. memo 33, Computer Science and Artificial Intelligence Laboratory. MIT, Cambridge (1962)
56. Minsky, M.: Computation, finite and infinite machines. Prentice-Hall, Englewood Cliffs (1967)
57. Moore, C., Mertens, S.: The Nature of Computation. Oxford University Press (2011)
58. Moore, C.: Quasi-linear cellular automata. Physica D 103, 100–132 (1997)
59. Moore, C.: Predicting non-linear cellular automata quickly by decomposing them into linear ones. Physica D 111, 27–41 (1998)
60. Moore, E.F.: A simplified universal Turing machine. In: ACM National Meeting, Toronto, Canada, pp. 50–54. ACM Press (1952)
61. Morita, K., Shirasaki, A., Gono, Y.: A 1-tape 2-symbol reversible Turing machine. The Transactions of the IEICE Japan E72(3), 223–228 (1989)
62. Morita, K., Yamaguchi, Y.: A Universal Reversible Turing Machine. In: Durand-Lose, J., Margenstern, M. (eds.) MCU 2007. LNCS, vol. 4664, pp. 90–98. Springer, Heidelberg (2007)
63. Neary, T.: Small universal Turing machines. PhD thesis, National University of Ireland, Maynooth (2008)
64. Neary, T., Woods, D.: P-Completeness of Cellular Automaton Rule 110. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part I. LNCS, vol. 4051, pp. 132–143. Springer, Heidelberg (2006)
65. Neary, T., Woods, D.: Small fast universal Turing machines. Theoretical Computer Science 362(1-3), 171–195 (2006)

66. Neary, T., Woods, D.: Four Small Universal Turing Machines. In: Durand-Lose, J., Margenstern, M. (eds.) MCU 2007. LNCS, vol. 4664, pp. 242–254. Springer, Heidelberg (2007)
67. Neary, T., Woods, D.: Small weakly universal Turing machines, arXiv:0707.4489v1 [cs.CC] (2007)
68. 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)
69. Neary, T., Woods, D.: Four small universal Turing machines. Fundamenta Informaticae 91(1), 123–144 (2009)
70. Nozaki, A.: On the notion of universality of Turing machine. Kybernetika Academia Praha 5(1), 29–43 (1969) (English translated version)
71. Ollinger, N.: The Quest for Small Universal Cellular Automata. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 318–329. Springer, Heidelberg (2002)
72. Ollinger, N., Richard, G.: Four states are enough! Theoretical Computer Science 412(1-2), 22–32 (2011)
73. Ottmann, T.: Eine universelle Turingmaschine mit zweidimensionalem band. Elektronische Informationsverarbeitung und Kybernetik 11(1-2), 27–38 (1975) (in German)
74. Ottmann, T.: Einfache universelle mehrdimensionale Turingmaschinen. Habilitationsschrift, Karlsruhe (1975)
75. Pavlotskaya, L.: Solvability of the halting problem for certain classes of Turing machines. Mathematical Notes 13(6), 537–541; Translated from Matematicheskie Zametki, 13(6), 899–909 (1973)
76. Pavlotskaya, L.: O minimal'nom chisle razlichnykh kodov vershin v grafe universal'noj mashiny T'juringa. Disketnyj Analiz, Sbornik Trudov Instituta Matematiki SO AN SSSR 27, 52–60 (1975); On the minimal number of distinct codes for the vertices of the graph of a universal Turing machine (in Russian)
77. Pavlotskaya, L.: Dostatochnye uslovija razreshimosti problemy ostanovki dlja mashin T'juring. Avtomaty i Mashiny, 91–118 (1978); Sufficient conditions for the halting problem decidability of Turing machines (in Russian)
78. Pavlotskaya, L.: On machines, universal by extensions. Theoretical Computer Science 168(2), 257–266 (1996)
79. Post, E.L.: Formal reductions of the general combinatorial decision problem. American Journal of Mathmatics 65(2), 197–215 (1943)
80. Post, E.L.: Recursive unsolvability of a problem of Thue. Journal of Symbolic Logic 12(1), 1–11 (1947)
81. Post, E.L.: Absolutely unsolvable problems and relatively undecidable propositions – account of an anticipation. In: Davis, M. (ed.) The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions, pp. 340–406. Raven Press, New York (1965); Corrected republication. Dover publications, New York (2004)
82. Priese, L.: Towards a precise characterization of the complexity of universal and nonuniversal Turing machines. SIAM J. Comput. 8(4), 508–523 (1979)
83. Rado, T.: On non-computable functions. Bell System Technical Journal 41(3), 877–884 (1962)
84. Richard, G.: A particular universal cellular automaton, HAL research report (oai:hal.archives-ouvertes.fr:hal-00095821_v1) (2006)
85. Robinson, R.M.: Undecidability and nonperiodicity for tilings of the plane. Inventiones Mathematicae 12(3), 177–209 (1971)

86. Robinson, R.M.: Minsky's small universal Turing machine. International Journal of Mathematics 2(5), 551–562 (1991)
87. Rogozhin, Y.: Sem' universal'nykh mashin T'juringa. In: Fifth All Union Conference on Mathematical Logic, Akad. Naul SSSR. Otdel. Inst. Mat., Novosibirsk, p. 27 (1979); Seven universal Turing machines (in Russian)
88. Rogozhin, Y.: Sem' universal'nykh mashin T'juringa. Systems and Theoretical Programming, Mat. Issled 69, 76–90 (1982); Seven universal Turing machines (in Russian)
89. Rogozhin, Y.: Universal'naja mashina T'juringa s 10 sostojanijami i 3 simvolami. Izvestiya Akademii Nauk Respubliki Moldova, Matematika 4(10), 80–82 (1992); A universal Turing machine with 10 states and 3 symbols (in Russian)
90. Rogozhin, Y.: About Shannon's problem for Turing machines. Computer Science Journal of Moldova 1(3), 108–111 (1993)
91. Rogozhin, Y.: Small universal Turing machines. Theoretical Computer Science 168(2), 215–240 (1996)
92. Rogozhin, Y.: A universal Turing machine with 22 states and 2 symbols. Romanian Journal of Information Science and Technology 1(3), 259–265 (1998) (in Russian)
93. Rogozhin, Y., Verlan, S.: On the Rule Complexity of Universal Tissue P Systems. In: Freund, R., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2005. LNCS, vol. 3850, pp. 356–362. Springer, Heidelberg (2006)
94. Rothemund, P.W.K.: A DNA and restriction enzyme implementation of Turing Machines. In: Lipton, R.J., Baum, E.B. (eds.) DNA Based Computers: Proceeding of a DIMACS Workshop. DIMACS, vol. 2055, pp. 75–119. Princeton University, AMS (1996)
95. Shannon, C.E.: A universal Turing machine with two internal states. Automata Studies, Annals of Mathematics Studies 34, 157–165 (1956)
96. Siegelmann, H.T., Margenstern, M.: Nine switch-affine neurons suffice for Turing universality. Neural Networks 12(4-5), 593–600 (1999)
97. Schroeppel, R.: A two counter machine cannot calculate 2^n. Technical Report AIM-257, A.I. memo 257, Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA (1972)
98. Takahashi, H.: Keisankikai II. Iwanami, Tokyo (1958); Computing machine II (in Japanese)
99. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 2(42), 230–265 (1936)
100. Wagner, K.: Universelle Turingmaschinen mit n-dimensionale band. Elektronische Informationsverarbeitung und Kybernetik 9(7-8), 423–431 (1973); Universal Turing machines with n-dimensional tapes (in German)
101. Wang, H.: A variant to Turing's theory of computing machines. Journal of the Association for Computing Machinery 4(1), 63–92 (1957)
102. Wang, H.: Tag systems and lag systems. Mathematical Annals 152(4), 65–74 (1963)
103. Watanabe, S.: On a minimal universal Turing machine. Technical report, MCB Report, Tokyo (1960)
104. Watanabe, S.: 5-symbol 8-state and 5-symbol 6-state universal Turing machines. Journal of the ACM 8(4), 476–483 (1961)
105. Watanabe, S.: 4-symbol 5-state universal Turing machine. Information Processing Society of Japan Magazine 13(9), 588–592 (1972)

106. Wolfram, S.: Statistical mechanics of cellular automata. Reviews of Modern Physics 55(3), 601–644 (1983)
107. Wolfram, S.: A new kind of science. Wolfram Media, Inc. (2002)
108. Woods, D., Neary, T.: On the time complexity of 2-tag systems and small universal Turing machines. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 439–446. IEEE, Berkeley (2006)
109. Woods, D., Neary, T.: Small Semi-Weakly Universal Turing Machines. In: Durand-Lose, J., Margenstern, M. (eds.) MCU 2007. LNCS, vol. 4664, pp. 303–315. Springer, Heidelberg (2007)
110. Woods, D., Neary, T.: The complexity of small universal Turing machines: A survey. Theoretical Computer Science 410(4-5), 443–450 (2009)
111. Woods, D., Neary, T.: Small semi-weakly universal Turing machines. Fundamenta Informaticae 91(1), 179–195 (2009)