Remarks on the computational complexity of small universal Turing machines

Damien Woods*
School of Mathematics
Boole Centre for Research in Informatics,
University College Cork, Ireland
http://www.bcri.ucc.ie/~dw5

Turlough Neary*
TASS, Department of Computer Science,
National University of Ireland, Maynooth, Ireland
http://cs.may.ie/~tneary

Abstract

This paper surveys some topics in the area of small universal Turing machines and tag systems. In particular we focus on recent results concerning the computational complexity of such machines.

Keywords: small universal Turing machines, polynomial time, tag systems.

1 Introduction

Shannon [22] was the first to consider the question of finding the smallest possible universal Turing machine (UTM), where size (s, l) is the number of states s and symbols l. Early attempts [5, 25] gave small UTMs that efficiently (with a polynomial time slowdown) simulate Turing machines.

In the early 1960s Cocke and Minsky [2] showed that 2-tag systems simulate Turing machines, but in an exponentially slow fashion. Minsky [12] found a 7-state, 4-symbol UTM that simulates 2-tag systems in polynomial time. So this small UTM simulates Turing machines via the following sequence of simulations

$$\text{Turing machine} \rightarrow \text{2-tag system} \rightarrow \text{small UTM} \tag{1}$$

where $A \rightarrow B$ denotes that A is simulated by B. Later Rogozhin [21] and others [1, 7, 19] used Minsky's technique of simulation via (1) to find small UTMs for a range of state-symbol pairs (see Figure 1). All of these small UTMs efficiently simulate 2-tag systems. However since the 2-tag simulation of Turing machines is exponentially slow, it remained open as to whether these

*Both authors are supported by the Irish Research Council for Science, Engineering and Technology

UTMs could be made to run in polynomial time. We have recently resolved this situation by showing that 2-tag systems efficiently simulate Turing machines.

Theorem 1 ([26]). Given a Turing machine M that computes in time t then there is a 2-tag system T_M that simulates the computation of M and computes in polynomial time $O(t^4(\log t)^2)$.

We immediately get the following interesting result.

Corollary 2 ([26]). The small UTMs of Minsky, Rogozhin and others [1, 7, 12, 19, 21] are polynomial time, $O(t^8(\log t)^4)$, simulators of Turing machines.

This is an exponential improvement on the previously known simulation time overhead and improves a forty year old result in the area of small UTMs.

Before this result was known it was entirely plausible that there was a trade-off between UTM program size complexity, and time/space complexity; the smallest UTMs seemed to be exponentially slow. However our result shows there is currently little evidence for such a claim.

Since this result, Neary has found $(9, 3)$ and $(18, 2)$ UTMs [13]. These machines are efficient simulators of Turing machines and simulate via bi-tag systems [14]. Bi-tag systems are a variant of tag system that use the technique of reading either one or two symbols per timestep in order to efficiently simulate Turing machines.

Prior to Theorem 1 the smallest known polynomial time UTMs are to be found in [16]. However these efficient machines are not as small those of Rogozhin et al., hence Theorem 2 represents a significant size improvement for efficient UTMs. On the other hand, the machines in [16] remain significant to this day since (i) they are the smallest known UTMs that *directly* simulate Turing machines and (ii) they have only a quadratic slowdown when simulating 1-tape machines.

Figure 1 illustrates the present situation.

1.1 Unanswered questions

Suppose we change the usual definition of UTM to allow for repeated blank words. Does this situation permit smaller universal machines? As yet there is no definite answer to this question, however we conjecture that it is indeed the case. As evidence we cite the small machines of Cook.[1] As Figure 1 shows, Cook's machines are significantly smaller than the 'non-repeated blank word' machines. A second piece of evidence is the fact that Cook uses the repeated blank word to store a (possibly universal) cyclic tag system program.

To illustrate potential difficulties of proving decidability for certain state-symbol classes we note that Margenstern [10] gives a number of small machines that simulate the famous Collatz $3x + 1$ problem (Margenstern gives $(2, 10)$,

[1] In the present paper, the phrase "small UTMs" refers to Turing machines that obey the standard definitions. Recently M. Cook [3] has found universal machines that are smaller than all others discussed in the present paper (see Figure 1). However Cook's UTMs are generalisations of standard Turing machines: their blank tape consists of an infinitely repeated word to the left and another to the right. Incidentally Cook's machines simulate Rule 110, which he showed to be universal via a magnificent construction in the same paper. Via Cook's construction, his universal machines are exponentially slow; in a recent paper [15] we have improved their simulation time overhead to polynomial.

$(4,4)$, $(5,3)$ and $(2,11)$ machines and mentions that Baiocchi has improved these results by giving $(2,8)$, $(5,2)$ and $(2,10)$ machines). Michel subsequently gave $(2,4)$, $(3,3)$ and $(5,2)$ machines that simulate an open *Collatz-like* problem [11], which suggests that moving the non-universal curve to these points could be difficult. However, Michel [11] has conjectured that all $(4,2)$ machines are decidable.

Can we have more efficient small UTMs? For example the machines that simulate 2-tag or bi-tag systems leave 'garbage' data on the tape, because of this they use space that is linear in the time of the simulated Turing machine. Can we have small UTMs that are more space efficient? For example it would be interesting to find small UTMs that simultaneously use space that is a constant times the space used by the simulated Turing machine and time that is a polynomial of the time used by the simulated Turing machine. Such machines would satisfy van Emde Boas' [23] notion of reasonable sequential machines.

What about small UTMs with less than polynomial time complexity? For example, consider the construction of a linear time UTM. The following idea makes it seem that direct simulation of Turing machines is the most straightforward way to achieve a linear time UTM. Let M be a one tape Turing machine. The program of M is encoded as a word that is positioned at the simulated tape head location on the UTM's tape. Simulating a transition rule would involve scanning through the encoded table of behaviour, however it is not necessary to scan the entire simulated tape contents. The idea is straightforward, however trying to construct *small* linear time UTMs could be difficult.

1.2 Applications

Besides Corollary 2 there are numerous other applications of Theorem 1. Rogozhin and Verlan [20] prove the universality of splicing tissue P systems with only 8 rules by simulating 2-tag systems. Simulation via small UTMs has been used to prove universality in cellular automata [9, 10]. Our result improves the efficiency of these simulations. For another example, Levin and Venkatesan [8, 24] used a small 8-state, 5-symbol, polynomial time UTM of Watanabe's [25] to show the average case NP-completeness of a graph colouring problem. Our construction gives polynomial time UTMs that are significantly smaller than Watanabe's and thus improves (lowers) the number of colours in their construction.

References

[1] C. Baiocchi. Three small universal Turing machines. In M. Margenstern and Y. Rogozhin, editors, *Machines, Computations, and Universality*, volume 2055 of *LNCS*, pages 1–10, Chişinău, Moldova, May 2001. Springer.

[2] J. Cocke and M. Minsky. Universality of tag systems with $p = 2$. *Journal of the Association for Computing Machinery*, 11(1):15–20, Jan. 1964.

[3] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.

[4] G. T. Hermann. The uniform halting problem for generalized one state Turing machines. In *Proceedings of the ninth annual Symposium on Switching*

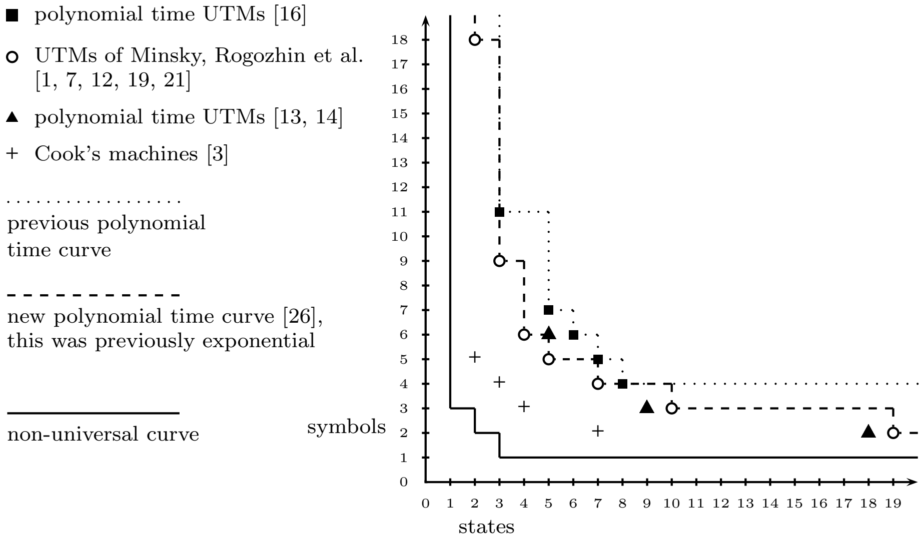

Figure 1: State-symbol plot of small UTMs. The plot shows the polynomial time curve induced by our previous UTMs [16], the previously exponential time curve of Minsky, Rogozhin and others [1, 7, 12, 21], and the non-universal Turing machine curve for which there are no UTMs [4, 6, 17, 18]. Theorem 1 [26] improves the polynomial time curve so that it coincides with the previous exponential time curve. The polynomial time UTMs from [13] represent a further improvement for 2 and 3 symbol machines.

and Automata Theory (FOCS), pages 368–372, Schenectady, New York, Oct. 1968. IEEE Computer Society Press.

[5] N. Ikeno. A 6-symbol 10-state universal Turing machine. In *Proceedings, Institute of Electrical Communications*, Tokyo, 1958.

[6] M. Kudlek. Small deterministic Turing machines. *Theoretical Computer Science*, 168(2):241–255, 1996.

[7] 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*, volume 2295 of *LNCS*, pages 311–318, Vienna, May 2001. Springer.

[8] L. A. Levin and R. Venkatesan. An average case NP-complete graph problem. `arXiv:cs.CC/0112001`, v2, 2002.

[9] K. Lindgren and M. Nordahl. Universal computation in simple one-dimensional cellular automata. *Complex Systems*, 4(3):299–318, 1990.

[10] M. Margenstern. Frontier between decidability and undecidability: a survey. *Theoretical Computer Science*, 231(2):217–251, Jan. 2000.

[11] P. Michel. Small Turing machines and generalized busy beaver competition. *Theoretical Computer Science*, 326:45–56, Oct. 2004.

[12] M. Minsky. Size and structure of universal Turing machines using tag systems. In *Recursive Function Theory: Proceedings, Symposium in Pure Mathematics*, volume 5, pages 229–238, Provelence, 1962. AMS.

[13] T. Neary. Small polynomial time universal Turing machines. In *MFCSIT'06 (this volume)*, Ireland. NUI Cork.

[14] T. Neary and D. Woods. A small fast universal Turing machine. Technical Report NUIM-CS-TR-2005-12, Department of Computer Science, NUI Maynooth, 2005.

[15] T. Neary and D. Woods. P-completeness of cellular automaton rule 110. In M. Bugliesi et al., editor, *International Colloquium on Automata Languages and Programming (ICALP 2006 Part 1)*, volume 4051 of *Lecture Notes in Computer Science*, pages 132–143. Springer, July 2006.

[16] T. Neary and D. Woods. Small fast universal Turing machines. *Theoretical Computer Science*, 2006. Accepted.

[17] L. Pavlotskaya. Solvability of the halting problem for certain classes of Turing machines. *Mathematical Notes (Springer)*, 13(6):537–541, June 1973.

[18] L. Pavlotskaya. Dostatochnye uslovija razreshimosti problemy ostanovki dlja mashin T'juring. *Problemi kibernetiki*, 33:91–118, 1978. (Sufficient conditions for the halting problem decidability of Turing machines) (in Russian).

[19] R. Robinson. Minsky's small universal Turing machine. *International Journal of Mathematics*, 2(5):551–562, 1991.

[20] Y. Rogozhin and S. Verlan. On the rule complexity of universal tissue P systems. In *Proceedings, 6^{th} International Workshop on Membrane Computing*, pages 510–516, Vienna, Jul. 2005.

[21] Y. Rogozhin. Small universal Turing machines. *Theoretical Computer Science*, 168(2):215–240, 1996.

[22] C. E. Shannon. A universal Turing machine with two internal states. *Automata Studies, Annals of Mathematics Studies*, 34:157–165, 1956.

[23] P. van Emde Boas. Machine models and simulations. In van Leeuwen, ed, *Handbook of Theoretical Computer Science*, vol A, chapt 1. Elsevier, 1990.

[24] R. Venkatesan and L. Levin. Random instances of a graph coloring problem are hard. In *Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC)*, pages 217–222, Chicago, Illinois, 1988. ACM.

[25] S. Watanabe. 5-symbol 8-state and 5-symbol 6-state universal Turing machines. *Journal of the ACM*, 8(4):476–483, Oct. 1961.

[26] D. Woods and T. Neary. On the time complexity of 2-tag systems and small universal Turing machines. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2006. (Accepted).