Small polynomial time universal Turing machines

Turlough Neary*
Department of Computer Science, NUI Maynooth, Ireland.
tneary@cs.may.ie

Abstract

We present new small polynomial time universal Turing machines with state-symbol pairs of $(9, 3)$ and $(18, 2)$. These machines simulate our new variant of tag system, the bi-tag system and are the smallest known universal Turing machines with 3-symbols and 2-symbols respectively.

1 Introduction

"What is the smallest possible universal Turing machine (UTM)?" Shannon [14] was the first to ask this question. In 1962 Minsky [6] created a 7-state, 4-symbol UTM that simulates Turing machines (TMs) via 2-tag systems. Minsky's technique was more recently used by Rogozhin et al. [13, 5, 1] to create the smallest known UTMs[1].

Recently the simulation overhead of TMs by 2-tag systems was improved from exponential [2] to polynomial [15]. Hence the UTMs of Minsky and Rogozhin et al. now simulate TMs[2] in $O(T^8(\log T)^4)$ time. Figure 1 is a state-symbol plot, here we see that these machines induce a curve which we call the $O(T^8(\log T)^4)$ time curve. Previously we gave UTMs that directly simulate TMs in $O(T^2)$ time [10]. Figure 1 illustrates the $O(T^2)$ time curve that is induced by these results. Let UTM(m, n) be the class of deterministic UTMs with m states and n symbols. It is known that the following classes are empty: UTM$(2, 2)$ [11], UTM$(3, 2)$ [12], UTM$(2, 3)$ (Pavlotskaya, unpublished), UTM$(1, n)$ [4] and UTM$(n, 1)$ (trivial) for $n \geqslant 1$. These results induce the non-universal curve in Figure 1.

Previously we gave a UTM in UTM$(5, 6)$ that simulates TMs via bi-tag systems (BTSs) in $O(T^6)$ time [8]. This UTM is given by the hollow triangle in Figure 1. Our new UTMs in UTM$(9, 3)$ and UTM$(18, 2)$ also simulate TMs via BTSs in $O(T^6)$ time. This result improve the state of the art in small efficient UTMs. Each of our new UTMs is represented by a solid triangle in Figure 1.

*This work was made possible by funding from IRCSET

[1]Here we only consider TMs that obey the standard definitions. Recently Cook [3] has constructed UTMs that are smaller than those discussed here. However Cook's UTMs are generalisations of standard TMs: their blank tape consists of an infinitely repeated word to the left and another to the right. It should be noted that Cook's UTMs are exponentially slow; in a recent paper [9] we have improved their simulation time overhead to polynomial.

[2]We assume simulated TMs complete their computation in time T and have a single tape.

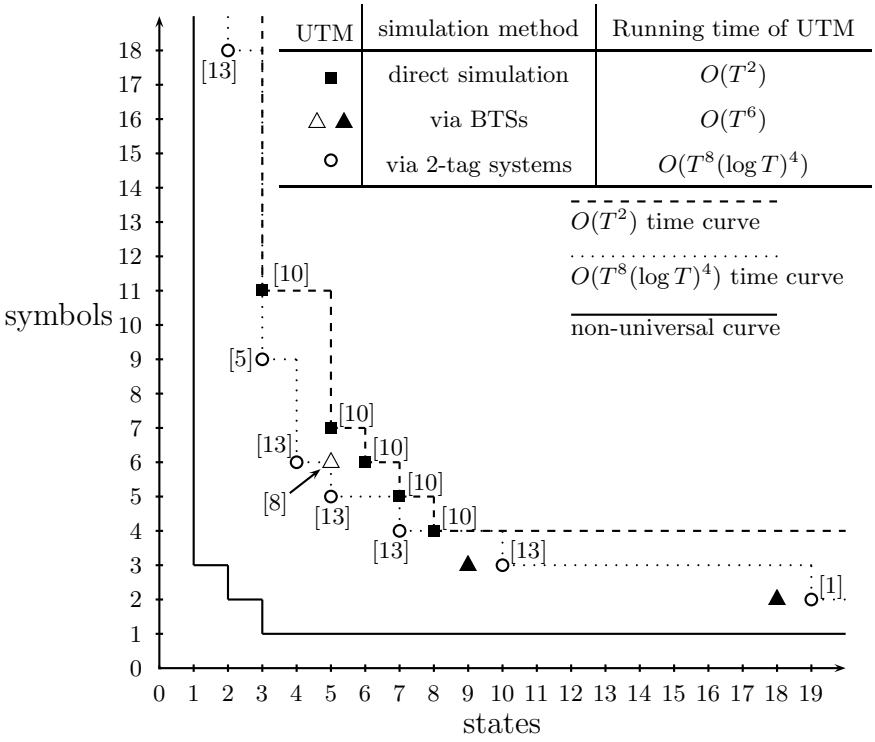

Figure 1: Current state-symbol plot of small UTMs. The table gives the graphical representation of each type of UTM followed by the method by which it simulates TMs and the computation time (see footnote 2) for each type of UTM. Each of our new UTMs is represented by a solid triangle. The plot shows the $O(T^2)$ and $O(T^8(\log T)^4)$ time curves induced by previously constructed UTMs.

2 BTS

The operation of a BTS [8] is similar to that of a 2-tag system [7]. The application of each production in a 2-tag system is dependent on exactly 1 symbol. BTSs use productions whose application is dependent on either 1 or 2 symbols.

Definition 1 (BTS)**.** A BTS is defined by the tuple (A, E, e_h, P). Here A and E are disjoint finite sets of symbols and $e_h \in E$ is the halt symbol. P is the finite set of productions. Each production is of one of the following three forms:

$$P(a) = a, \quad P(e, a) = AE, \quad P(e, a) = AAE$$

where $a \in A$ and P is defined on all elements of $\{A \cup ((E - \{e_h\}) \times A)\}$ and undefined on all elements of $\{e_h\} \times A$. BTSs are deterministic.

A configuration of a BTS is a word of the form $s = A^*(AE \cup EA)A^*$ called the data word. If a configuration s_2 is obtained from s_1 via the application of a single production we write $s_1 \vdash s_2$. In the Definition 2 let $a \in A$ and $e \in E$.

Definition 2 (BTS computation step)**.** A P production is applied in one of two ways:

(i) If $s = as'$ then $as' \vdash s'P(a)$.

(ii) If $s = eas'$ then $eas' \vdash s'P(e, a)$.

3 UTMs

We denote our UTMs in UTM(9, 3) and UTM(18, 2) as $U_{9,3}$ and $U_{18,2}$ respectively. In this section we give $U_{9,3}$ (Table 2) and $U_{18,2}$ (Table 3) and explain how they simulate BTSs. First we give the input encoding to our UTMs. Let R denote a BTS that is to be simulated. The encoding of R as a word is denoted $\langle R\rangle$. The encodings of $a \in A$ and $e \in E$ are denoted $\langle a\rangle$ and $\langle e\rangle$ respectively. The encodings of $P(a)$ and $P(e, a)$ are denoted as $\langle P(a)\rangle$ and $\langle P(e, a)\rangle$ respectively. Let R be a BTS $R = (A, E, e_h, P)$ where $A = \{a_1, \ldots, a_g\}$ and $E = \{e_1, \ldots, e_h\}$

Definition 3. The encoding of a configuration of R is of the form

$$^{\omega}c\langle R\rangle c^*(\langle A\rangle m)^*\Big(\langle A\rangle m\langle E\rangle \cup \langle E\rangle\langle A\rangle m\Big)(\langle A\rangle m)^* r c^{\omega} \tag{1}$$

where $(\langle A\rangle m)^*\Big(\langle A\rangle\langle E\rangle \cup \langle E\rangle\langle A\rangle\Big)(\langle A\rangle c)^*$ encodes R's data word via Equation (3), $c^{\omega} = ccc\ldots$, $^{\omega}c = \ldots c$, and $\langle R\rangle$ is given by Equations (2) and (3), and Table 1.

$$\begin{aligned}\langle R\rangle =& H v\langle P(e_{h-1}, a_g)\rangle v\langle P(e_{h-1}, a_{g-1})\rangle \ldots v\langle P(e_{h-1}, a_1)\rangle \\ &\vdots \\ & v\langle P(e_1, a_g)\rangle v\langle P(e_1, a_{g-1})\rangle \ldots v\langle P(e_1, a_1)\rangle \\ & v^2\langle P(a_g)\rangle v^2\langle P(a_{g-1})\rangle \ldots v^2\langle P(a_1)\rangle V\end{aligned} \tag{2}$$

UTM	$\langle P(a_i)\rangle$	$\langle P(e_j, a_i)\rangle$ $P(e_j, a_i) = a_k e_m$	$\langle P(e_j, a_i)\rangle$ $P(e_j, a_i) = a_v a_k e_m$
$U_{9,3}$	$\lambda\lambda cc\lambda c^{8i-4}$	$\lambda cc\lambda\lambda c^{8mg+2}\lambda c^{8k-4}$	$\lambda\lambda c^{8mg+2}\lambda c^{8k-4}\lambda c^{8v-4}$
$U_{18,2}$	$cb(cc)^2cb(cc)^{4i-2}$	$(cb)^2(cc)^{4gj+2}cb(cc)^{4k-2}$	$cb(cc)^{4gj+2}cb(cc)^{4k}cb(cc)^{4v-2}$

Table 1: Encoding of P productions for $U_{9,3}$ and $U_{18,2}$. Here $a_i, a_k, a_v \in A$ and $e_j, e_m \in E$ and below ϵ is the empty word.

$$\begin{aligned}&\langle a_i\rangle = b^{4i-3}, \langle e_j\rangle = b^{4jg}, m = \lambda, r = \epsilon, v = \lambda cc && \text{if UTM is } U_{9,3} \\ &\langle a_i\rangle = (bc)^{4i-1}, \langle e_j\rangle = (bc)^{4jg}, m = b^2, r = (bc)^2, v = cb && \text{if UTM is } U_{18,2}\end{aligned} \tag{3}$$

In Equation (2) $V = v$ for $U_{9,3}$ and $V = v^3$ for $U_{18,2}$ we do not give a value here for H. H is a word that causes our UTM to halt when the encoded halt symbol $\langle e_h\rangle$ is read. The position of the UTM's tape head is over the symbol immediately to the right of $\langle R\rangle c^*$.

The operation of $U_{9,3}$ and $U_{18,2}$ is similar to that of our UTM in UTM(5, 6) [8]. We explain the operation of $U_{9,3}$. Using a unary indexing method, $U_{9,3}$ locates the encoded production to be simulated. The encoded production ($\langle P(a_i)\rangle$ or $\langle P(e_j, a_i)\rangle$) is indexed (pointed to) by the number of b symbols contained in the leftmost encoded symbol or pair of symbols in the encoded data word (Equation (1)). If the leftmost encoded symbol is $\langle a_i\rangle = b^{4i-3}$ (Equation (3)) then the value $4i - 3$ is used to index $\langle P(a_i)\rangle$. If the leftmost encoded symbol is $\langle e_j\rangle = b^{4jg}$, and $\langle a_i\rangle = b^{4i-3}$ is adjacent then the value $4jg + 4i - 3$ is used

$U_{9,3}$	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9
c	bRu_1	cLu_3	cLu_3	bLu_9	cRu_6	bLu_4	λLu_4	cRu_7	bLu_5
b	cLu_2	cLu_2	bLu_4	bLu_4		bRu_6	bRu_7	cRu_9	cRu_8
λ	λRu_3	λLu_2	λRu_1	λLu_4	λLu_8	λRu_6	λRu_7	λRu_8	cRu_1

Table 2: Table of behaviour for $U_{9,3}$. The blank symbols is c. The start state is u_1.

to index $\langle P(e_j, a_i)\rangle$. The number of b symbols in the encoded symbol or pair of symbols is equal to the number of λc^* words between the leftmost encoded symbol and the encoded production to be indexed. To locate this production, $U_{9,3}$ simply changes each λc^* to λb^* for each b in the leftmost encoded symbol or pair of symbols. This process continues until the λ that separates two encoded symbols is read. The indexed production is then printed immediately to the right of the encoded data word which completes the execution of the production.

We give an example of $U_{9,3}$ indexing the encoded production $\langle P(a_1)\rangle$. In configurations below the tape head of $U_{9,3}$ is given by an underline. $U_{9,3}$'s current state is given on the left in bold. The data word $a_1e_ja_i$ is encoded via Equations (1) and (3) as $b\lambda b^{4jg}b^{4i-3}$. From Equation (1) we get the initial :

$\boldsymbol{u_1}, \ldots \langle P(a_2)\rangle(\lambda cc)^2\langle P(a_1)\rangle\lambda cc\underline{b}\lambda b^{4jg}b^{4i-3}\lambda ccc\ldots$

Below we have replaced $\langle P(a_1)\rangle$ with its encoding $\lambda\lambda cc\lambda cccc$ via Table 1.

$\boldsymbol{u_2}, \ldots \langle P(a_2)\rangle(\lambda cc)^2\lambda\lambda cc\lambda cccc\lambda c\underline{c}c\lambda b^{4jg}b^{4i-3}\lambda ccc\ldots$

$\boldsymbol{u_3}, \ldots \langle P(a_2)\rangle(\lambda cc)^2\lambda\lambda cc\lambda cccc\underline{\lambda}ccc\lambda b^{4jg}b^{4i-3}\lambda ccc\ldots$

$\boldsymbol{u_1}, \ldots \langle P(a_2)\rangle(\lambda cc)^2\lambda\lambda cc\lambda cccc\lambda bbb\underline{\lambda}b^{4jg}b^{4i-3}\lambda ccc\ldots$

$\boldsymbol{u_3}, \ldots \langle P(a_2)\rangle(\lambda cc)^2\lambda\lambda cc\lambda cccc\lambda bbb\lambda\underline{b^{4jg}}b^{4i-3}\lambda ccc\ldots$

$U_{9,3}$ now prints a single b immediately to the right of the encoded data word for each cc pair it reads (with the exception of the leftmost 4 c symbols) in $\langle P(a_1)\rangle$. When $U_{9,3}$ reads the substring $c\lambda cc$ in $\langle P(a_1)\rangle$ it scans right and prints a λ to the right of the newly printed $\langle a_1\rangle$ symbol at the right end of the encoded data word. Finally when $U_{9,3}$ reads the substring $\lambda\lambda cc$ in $\langle P(a_1)\rangle$ it is finished printing the encoded production. $U_{9,3}$ now restores $\langle R\rangle$ to its original value and begins simulating the next production.

The operation of $U_{18,2}$ is similar to that of $U_{9,3}$ however there are some difference. The encoded production is indexed by the number of bc subwords in the leftmost encoded symbol or pair of symbols. During indexing, for each bc subword in the leftmost encoded symbol or pair of symbols the leftmost c preceded by no more than one b is changed to a b. The remainder of the computation proceeds in a similar manner.

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. MCU, Springer.

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

$U_{18,2}$	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9
c	bRu_2	cRu_1	cLu_5	cLu_5	cLu_4	bRu_2	cLu_8	bRu_{12}	bLu_{10}
b	bRu_3	bRu_1	bLu_9	bLu_6	cLu_4	cLu_4	bLu_9	bLu_7	bLu_7

$U_{18,2}$	u_{10}	u_{11}	u_{12}	u_{13}	u_{14}	u_{15}	u_{16}	u_{17}	u_{18}
c	cRu_{13}	bLu_7	cRu_{11}	cLu_{15}	cRu_{13}	bLu_9	cRu_{17}		cRu_{15}
b	bRu_{15}	bRu_{12}	bRu_{11}	bRu_{14}	bRu_{13}	cRu_{16}	bRu_{15}	cRu_{18}	cRu_1

Table 3: Table of behaviour for $U_{18,2}$. The blank symbols is c. The start state is u_1.

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

[4] G. Hermann. The uniform halting problem for generalized one state Turing machines. In *Proceedings, Ninth Annual Symposium on Switching and Automata Theory*, pages 368–372, New York, Oct. 1968. IEEE.

[5] M. Kudlek and Y. Rogozhin. A universal Turing machine with 3 states and 9 symbols. In W. Kuich et al., editors, *Developments in Language Theory*, volume 2295 of *LNCS*, pages 311–318, Vienna, May 2002.

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

[7] M. Minsky. *Computation finite and infinite machines*. Prentice-Hall, 1967.

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

[9] T. Neary and D. Woods. P-completeness of cellular automaton Rule 110. In M. Bugliesi et al., editor, *ICALP 2006, Part I*, volume 4051 of *LNCS*, pages 132–143, Venice, July 2006. Springer.

[10] T. Neary and D. Woods. Small fast universal Turing machines. *TCS*, 2006. Accepted. Preliminary version available as Department of computer science, NUI Maynooth, Technical Report: NUIM-CS-TR-2005-11.

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

[12] 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).

[13] Y. Rogozhin. Small universal Turing machines. *TCS*, 168(2):215–240, 1996.

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

[15] D. Woods and T. Neary. On the time complexity of 2-tag systems and small universal Turing machines. FOCS, Berkeley, California, 2006. Accepted.