A small fast universal Turing machine

Turlough Neary[1] and Damien Woods[2]

[1] Theoretical Aspects of Software Systems Research Group,
Department of Computer Science,
National University of Ireland Maynooth, Ireland.
tneary@cs.may.ie
[2] Department of Mathematics and Boole Centre for Research in Informatics,
University College Cork, Ireland.
d.woods@bcri.ucc.ie

Communicated by: J. Paul Gibson

Technical report: NUIM-CS-2005-TR-12

Key words: small universal Turing machine, tag system, polynomial time, universality.

Abstract. We present a small time-efficient universal Turing machine with 5 states and 6 symbols. This Turing machine simulates our new variant of tag system. It is the smallest known universal Turing machine that simulates Turing machine computations in polynomial time.

1 Introduction

We present a new small deterministic polynomial time universal Turing machine (UTM). This Turing machine (TM) simulates our new variant of tag system and is the smallest known UTM that simulates TMs in polynomial time. This is an improvement on our previous results [11].

In the early literature small UTMs directly simulated TMs [16, 6]. Subsequently the technique of indirect simulation via other universal models was successfully used to construct small UTMs. In the early 1960s Minsky used 2-tag systems to create a 7-state, 4-symbol machine [9]. Minsky's technique was more recently used to create the smallest known UTMs.

Let UTM(m, n) be the class of deterministic universal TMs with m states and n symbols. Rogozhin [14] constructed UTMs in the classes UTM$(10, 3)$, UTM$(7, 4)$, UTM$(5, 5)$, UTM$(4, 6)$ and UTM$(2, 18)$, Kudlek and Rogozhin [8] constructed a UTM in UTM$(3, 9)$ and Baiocchi [1] constructed UTMs in UTM$(19, 2)$ and UTM$(7, 4)$. Baiocchi's 4-symbol UTM uses only 25 transition rules (TRs) whereas Rogozhin's uses 26. Due to their unary encoding of the TM tape contents 2-tag systems are exponentially slow simulators of TMs [2]. Hence the simulations of Minsky, Rogozhin, Kudlek and Baiocchi all suffer from an exponential time complexity overhead. Fig. 1 is a state-symbol plot, here we see that these machines induce a curve that we call the *exponential time curve*. Previously we have constructed polynomial time UTMs in UTM$(3, 11)$, UTM$(5, 7)$, UTM$(6, 6)$, UTM$(7, 5)$ and UTM$(8, 4)$ [11]. Fig. 1 illustrates the *polynomial time curve* that is induced by these results. The halting problem has been proved decidable for all

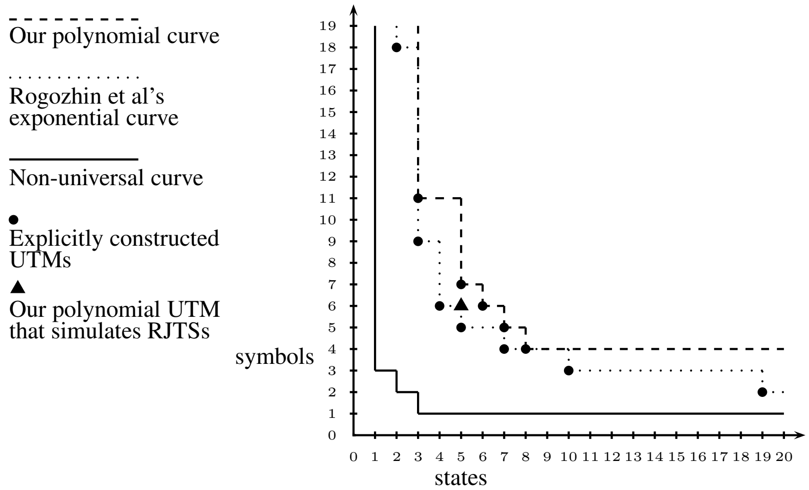


Fig. 1. State-symbol plot of small UTMs. The plot shows our new small polynomial time UTM, the polynomial time curve induced by our previous UTMs, Rogozhin et al's exponential time curve, and the non-universal TM curve. A corollary from our new small polynomial time UTM is that there are polynomial time UTMs in UTM(m, n) for all $m \geq 5$, $n \geq 6$.

deterministic TMs in the classes TM$(2, 2)$ [12, 7], TM$(3, 2)$ [13], TM$(2, 3)$ (Pavlotskaya, unpublished), TM$(1, m)$ [4] and TM$(m, 1)$ (trivial) for $m \geq 1$. These results induce the *non-universal curve* in Fig 1.

Our main result states that there exist a deterministic polynomial time UTM in the class UTM$(5, 6)$. In Fig. 1 this UTM is represented by a triangle. This UTM simulates a new variant on the tag system which we call the bi-tag system (BTS). BTSs simulate TMs in polynomial time. Hence our UTM in UTM$(5, 6)$ avoids the exponential time overhead introduced by simulating standard 2-tag systems. Our machine is smaller than the machines in [11] and represents a new algorithm for small UTMs.

2 Preliminaries

Some discusion and definitions relating to the concepts of *simulation* and *simulate in polynomial time* can be found in [11, 14, 15].

We consider deterministic TMs with a single bi-infinite tape and a single tape head [5]. A TM is a tuple $M = (Q, \Sigma, B, f, q_1, H)$. Here Q and Σ are the finite sets of states and tape symbols respectively. Also, $B \in \Sigma$ is the blank symbol, $q_1 \in Q$ is the start state, and $H \subseteq Q$ is the set of halt states. The transition function $f : Q \times \Sigma \rightarrow \Sigma \times \{L, R\} \times Q$ is defined for all $q \in Q - H$. If $q \in H$ then the function f is undefined on at least one element of $q \times \Sigma$. We write f as a list of TRs. Each TR is a quintuple $t = (q_x, \sigma_1, \sigma_2, D, q_y)$, with initial state q_x, read symbol σ_1, write symbol σ_2, tape head direction D and next state q_y.

Throughout the paper we let M be a deterministic single tape TM with $|Q|$ states, $|\Sigma|$ symbols, and time complexity $T(n)$ on input length n. Also $U_{5,6}$ denotes our UTM in class UTM$(5, 6)$. In regular expressions $\cup$, * and parentheses have their usual meanings [5].

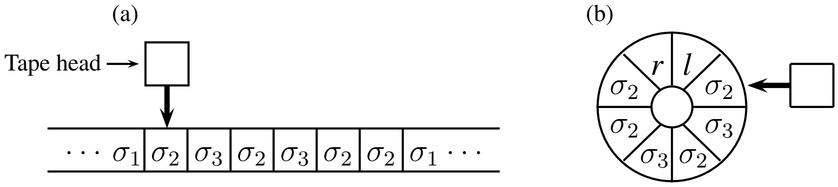


Fig. 2. (a) Example TM tape contents. The TM's blank symbol is σ_1. (b) CTM encoding of the TM tape contents in (a), the symbols r and l encode the infinite sequence of blank symbols to the right and left of M's encoded tape contents respectively.

3 Clockwise TM

Definition 1 (clockwise TM). *A clockwise TM (CTM) is a tuple $C = (Q, \Sigma, f, q_1, q_{|Q|})$. Q and Σ are the finite sets of states and tape symbols respectively, $q_1 \in Q$ is the start state and $q_{|Q|} \in Q$ is the halt state. The transition function $f : Q \times \Sigma \to \{\Sigma \cup \Sigma\Sigma\} \times Q$ is undefined on state $q_{|Q|}$ and is defined for all $q \in Q$, $q \neq q_{|Q|}$.*

We write f as a list of clockwise transition rules (CTRs). Each CTR is a quadruple $t = (q_x, \sigma_1, w, q_y)$, with initial state q_x, read symbol σ_1, write value $w \in \{\Sigma \cup \Sigma\Sigma\}$ and next state q_y. A CTM has a circular tape and its operation is similar to that of a TM. A CTR is executed as follows: If the write value w is from Σ then the tape cell containing the read symbol is overwritten by w, if w is from $\Sigma\Sigma$ then the cell containing the read symbol becomes 2 cells each of which contain one symbol from w. The machine's state becomes q_y and the tape head moves clockwise by one tape cell. CTMs are deterministic.

Lemma 1. *The computation of TM M is simulated using space $O(T(n))$ and time $O(T^2(n))$ by a CTM C_M.*

Proof. Let N be a TM that has the following restrictions: (i) the blank symbol σ_1 does not appear as input to N, (ii) N may read the blank symbol but is not permitted to write it to the tape, (iii) N has exactly one final state. Due to the restrictions placed on N we know that when N reads a blank symbol it is either at the left or right end of its tape contents. We wish to simulate a TM $M = (\{q_1, \ldots, q_{|Q|}\}, \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}, \sigma_1, f, q_1, \{q_{|Q|}\})$ without restrictions. M is converted to a restricted TM N that requires one extra state and one extra symbol, and executes one extra computation step. We define $N = (\{q_1, \ldots, q_{|Q|+1}\}, \{\sigma_1, \ldots, \sigma_{|\Sigma|+1}\}, \sigma_1, f, q_1, \{q_{|Q|+1}\})$. We construct a CTM C_M that simulates M via N. $C_M = (Q_C, \Sigma_C, f_C, q_1, q_{|Q|+1})$ where Q_C, Σ_C, f_C are defined below.

$$\Sigma_C = \{\sigma_2, \ldots, \sigma_{|\Sigma|+1}, r, l, \gamma\}$$

The symbol γ is a special marker symbol and symbols r and l encode the infinite sequence of blank symbols to the right and left of N's encoded tape contents respectively (see Fig. 2).

$$Q_C = \{q_1, q_{1,2}, \ldots, q_{1,|\Sigma|+1}, q_{1,r}, q_{1,r'}, q_{1,l}, q_2, q_{2,2}, \ldots, q_{2,|\Sigma|+1}, q_{2,r}, q_{2,r'}, q_{2,l}, \ldots, q_{|Q|+1}\}$$

We can think of right moves of N's tape head as clockwise moves. Here we give right move TRs followed by the CTRs that simulate them.

$$q_x, \sigma_k, \sigma_j, R, q_y \quad : \quad q_x, \sigma_k, \sigma_j, q_y \tag{1}$$

$$q_x, \sigma_1, \sigma_j, R, q_y \quad : \quad q_x, l, l\sigma_j, q_y \tag{2}$$

where $\sigma_k, \sigma_j \neq \sigma_1$. The CTR in Equation (2) simulates N printing the write symbol σ_j over the blank symbol immediately to the left of it's tape contents. The CTR's write value $l\sigma_j$ also preserves l; the encoding of the infinite sequence of blank symbols to the left of the tape contents.

The final right moving case is when N's tape head is over the blank symbol immediately to the right of its tape contents. Initially C_M's tape head is over r and immediately after simulation of the TR, C_M's tape head is again over r. Immediately below are the CTRs that simulate this case.

$$q_x, \sigma_1, \sigma_j, R, q_y \quad : \quad \begin{array}{ll} q_x, r, \sigma_j r, q_{y,r'} & (*) \\ q_{y,r'}, \kappa, \kappa, q_{y,r'} & (**) \end{array}$$

where $\kappa \in \Sigma_C - \{\gamma, r\}$. CTR (*) prints N's encoded write symbol σ_j and sends C_M's control into state $q_{y,r'}$. State $q_{y,r'}$ moves C_M's tape head to the cell containing r. This completes the simulation of the TR.

Left moving TRs are more difficult to simulate as C_M's tape head moves only clockwise. C_M begins by marking the current location of the tape head with the symbol γ. C_M now moves each symbol clockwise by one cell. When C_M's tape head reads γ the left move is complete. This process moves the tape head anti-clockwise relative to the tape contents. Immediately below is given the CTRs that mark the tape head's location with the symbol γ.

$$\begin{array}{lcl} q_x, \sigma_1, \sigma_j, L, q_y & : & q_x, l, l\gamma, q_{y,j} \\ q_x, \sigma_1, \sigma_j, L, q_y & : & q_x, r, \gamma\sigma_j, q_{y,r} \\ q_x, \sigma_k, \sigma_j, L, q_y & : & q_x, \sigma_k, \gamma, q_{y,j} \end{array}$$

The CTRs that move each symbol clockwise by one cell are of the form:

$$q_{y,\upsilon}, \rho, \upsilon, q_{y,\rho}$$

where $\upsilon, \rho \in \Sigma_C - \{\gamma\}$. When C_M's tape head reads γ then C_M is in a state of the form $q_{y,\rho}$ and the unique CTR defined by the state-symbol pair $(q_{y,\rho}, \gamma)$ will begin simulation of the next TR. This TR is of the form $(q_y, \sigma_1, \sigma_k, D, q_z)$ if $\rho = r, l$ and of the form $(q_y, \rho, \sigma_k, D, q_z)$ if $\rho \neq r, l$.

Input to N is encoded for C_M by a finite state transducer. Given this encoded input C_M simulates the sequence of $T(n)$ TRs in N's computation and halts in state $q_{|Q|+1}$ the encoding of N's halt state $q_{|Q|+1}$. C_M uses space of $O(T(n))$. A single computation step of N is simulated in $O(T(n))$ steps of C_M. Thus the computation time of C_M is $O(T^2(n))$. □

4 BTS

In this section we present the BTS, our new variant on the tag system, and prove that it simulates TMs. The operation of a BTS is similar to that of a standard tag system [10]. The application of each production in a tag system is dependent on exactly 1 symbol. BTSs use productions whose application is dependent on either 1 or 2 symbols. Also BTSs are deterministic.

Definition 2 (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 called halt symbol. P is the finite set of productions where each production is of one of the following three forms:*

$$A \rightarrow A$$
$$E \times A \rightarrow A \times E$$
$$E \times A \rightarrow A \times A \times E$$

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

Definition 3 (BTS configuration). *A configuration of a BTS is a word of the form $s = A^*(AE \cup EA)A^*$.*

We may think of s as the data word of a BTS. The computation of a BTS proceeds by deleting symbols from the left end of s and appending new symbols to the right end of s. 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 sequel $P(a)$ denotes the value mapped to by the symbol $a \in A$ and $P(ea)$ denotes the value mapped to by the pair of symbols $e \in E$ and a. In the following definition let $a \in A$, $e \in E$.

Definition 4 (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 $eas' \vdash s'P(ea)$.

A BTS computation is a finite sequence of computation steps that are consecutively applied to an initial data word. If e_h is the leftmost symbol in the data word then the computation halts.

Example 1. (BTS computation.) Let BTS $R_1 = (\{a_0, a_1\}, \{e_0, e_1, e_2\}, e_2, P)$ where $P = \{a_0 \rightarrow a_0, a_1 \rightarrow a_1, e_0a_0 \rightarrow a_1e_0, e_0a_1 \rightarrow a_1e_2, e_1a_0 \rightarrow a_0e_0, e_1a_1 \rightarrow a_1e_2\}$. Given the word $a_1e_0a_0$, the computation of R_1 proceeds as follows:

$$a_1e_0a_0 \vdash e_0a_0a_1 \vdash a_1a_1e_0 \vdash a_1e_0a_1 \vdash e_0a_1a_1 \vdash a_1a_1e_2 \vdash a_1e_2a_1 \vdash e_2a_1a_1$$

The computation halts as the halt symbol e_2 is the leftmost symbol. □

Lemma 2. *Let C be a CTM that runs in time $T(n)$ on input length n. The computation of C is simulated using space $O(T(n))$ and time $O(T^2(n))$ by a BTS R_C.*

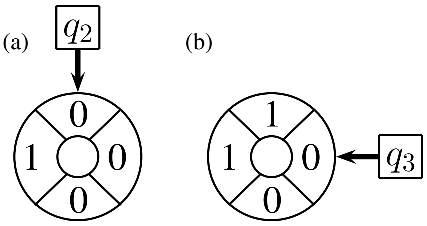
Before giving the proof of Lemma 2 we explain the proof idea. Each A symbol of R_C encodes a symbol of C's tape alphabet. Each E symbol of R_C encodes a state of C. The location of the E symbol in the data word represents the location of C's tape head, as illustrated in Fig. 3.

Each CTR of C is simulated in the following way. The change of state, symbol and tape head position is simulated by executing a P production over the $E \times A$ pair that encodes the current state and read symbol (see Fig. 3(c)). A production is then applied to each symbol in the data word. This moves the new $E \times A$ pair to the left of the data word, in order to prepare for the simulation of the next CTR.

Proof. Let CTM C=$(\{q_1, \ldots, q_{|Q|}\},\ \{\sigma_1, \ldots, \sigma_{|\Sigma|}\},\ f,\ q_1,\ q_{|Q|})$. We construct a BTS R_C that simulates C's computation.

$$R_C = (A_C,\ E_C,\ e_{|Q|},\ P_C)$$

(a) (b)



(c)	e_2 0 0 0 1	apply $e_2 0 \rightarrow 1 e_3$
	0 0 1 1 e_3	apply $0 \rightarrow 0$
	0 1 1 e_3 0	apply $0 \rightarrow 0$
	1 1 e_3 0 0	apply $1 \rightarrow 1$
	1 e_3 0 0 1	apply $1 \rightarrow 1$
(d)	e_3 0 0 1 1	TR simulation complete

Fig. 3. BTS simulating the CTR $(q_2, 0, 1, q_3)$. CTM states q_2 and q_3 are encoded as e_2 and e_3 respectively. The e symbols also mark the location of the simulated tape head. (a) Configuration of the CTM before execution of the CTR. (b) Configuration of the CTM after execution of the CTR. (c) BTS encoding of configuration in (a). (d) BTS encoding of configuration in (b).

where A_C, E_C, P_C are defined below.

$$A_C = \{a_1, \ldots, a_{|\Sigma|}\}$$

C's tape symbols $\sigma_1, \ldots, \sigma_{|\Sigma|}$ are encoded as $a_1, \ldots, a_{|\Sigma|}$ respectively.

$$E_C = \{e_1, \ldots, e_{|Q|}\}$$

C's states $q_1, \ldots, q_{|Q|}$ of R are encoded as $e_1, \ldots, e_{|Q|}$ respectively and the encoded halt state $e_{|Q|}$ is the halt symbol of R_C.

$$P_C = \{a_1 \rightarrow a_1, \ldots, a_{|\Sigma|} \rightarrow a_{|\Sigma|}\} \cup P'_C$$

P'_C is the productions defined on $(E - \{e_{|Q|}\}) \times A$. There is one production in P'_C for each CTR in C. CTRs fall in two categories, those that write a single symbol from Σ and those that write a pair of symbols from $\Sigma\Sigma$. The two possible CTRs, and their encodings as productions, are as follows

$$\begin{array}{rcl} (q_x, \sigma_i, \sigma_j, q_y) & : & e_x a_i \rightarrow a_j e_y \\ (q_x, \sigma_i, \sigma_j \sigma_k, q_y) & : & e_x a_i \rightarrow a_j a_k e_y \end{array}$$

We have constructed a BTS R_C that simulates C. R_C uses $O(T(n))$ space. To simulate a computation step of C, a production is applied to each symbol in the data word that encodes the current configuration of C. This takes $O(T(n))$ steps and yields a new data word that encodes the next configuration of C's computation. In this way R_C simulates $T(n)$ steps of C's computation in time $O(T^2(n))$. The simulation halts when the halt symbol $e_{|Q|}$ that encodes the halt state becomes the leftmost symbol in the data word. $\square$

Given a single tape TM M that runs in time $T(n)$, we conclude from the previous two lemmata that M is simulated by a BTS in time $O(T^4(n))$. However this overhead is easily improved to $O(T^3(n))$ as the next theorem shows.

Theorem 1. *The computation of TM M is simulated using space $O(T(n))$ and time $O(T^3(n))$ by a BTS R_M.*

Proof. From Lemmata 1 and 2 a BTS simulates the computation of M via a CTM C_M. From Lemma 1 C_M simulates M in time $O(T^2(n))$. However C_M uses $O(T(n))$ space, hence R_M uses $O(T(n))$ space. R_M applies $O(T(n))$ productions to simulate a CTR of C_M. Thus R_M executes $O(T^2(n))$ CTRs to simulates M via C_M in time $O(T^3(n))$. $\square$

5 $U_{5,6}$

In this section we give $U_{5,6}$ our UTM which simulates BTSs. We begin by giving the input encoding to $U_{5,6}$ followed by the definition of $U_{5,6}$. We let R denote a BTS that is to be simulated by $U_{5,6}$. The encoding of R as a word is denoted $\langle R\rangle$. Analogously the encodings of $a \in A$ and $e \in E$ are denoted $\langle a\rangle$ and $\langle e\rangle$ respectively. The encodings of productions $P(a)$ and $P(ea)$ are denoted as $\langle P(a)\rangle$ and $\langle P(ea)\rangle$ respectively. Let R be a BTS $R = (A, E, e_h, P)$ where

$$\begin{aligned} A &= \{a_1, \ldots, a_g\} \\ E &= \{e_1, \ldots, e_h\} \end{aligned}$$

Definition 5 (encoding of R's input symbols). *The input symbols $a_i \in A$ and $e_j \in E$ are encoded as follows $\langle a_i\rangle = \overleftarrow{b}^{4i-3}$ and $\langle e_j\rangle = \overleftarrow{b}^{4jg}$.*

Definition 6 (encoding of a configuration of R). *The encoding of a configuration of R is of the form*

$$^{\omega}d\langle R\rangle d^{*}(\langle A\rangle c)^{*}\Big(\langle A\rangle\langle E\rangle \cup \langle E\rangle\langle A\rangle\Big)(\langle A\rangle c)^{*}d^{\omega} \tag{3}$$

where $(\langle A\rangle c)^{*}\Big(\langle A\rangle\langle E\rangle\cup\langle E\rangle\langle A\rangle\Big)(\langle A\rangle c)^{*}$ encodes R's data word via Definition 5, $d^{\omega} = ddd\ldots$, $^{\omega}d = \ldots ddd$, and $\langle R\rangle$ is the encoding of R:

$$\begin{aligned} \langle R\rangle = \overrightarrow{b}c&\lambda^2\langle P(e_{h-1}a_g)\rangle\lambda^2\langle P(e_{h-1}a_{g-1})\rangle\ldots\lambda^2\langle P(e_{h-1}a_1)\rangle \\ &\lambda^2\langle P(e_{h-2}a_g)\rangle\lambda^2\langle P(e_{h-2}a_{g-1})\rangle\ldots\lambda^2\langle P(e_{h-2}a_1)\rangle \\ &\quad\vdots \\ &\lambda^2\langle P(e_1a_g)\rangle\lambda^2\langle P(e_1a_{g-1})\rangle\ldots\lambda^2\langle P(e_1a_1)\rangle \\ &\lambda^3\langle P(a_g)\rangle\lambda^3\langle P(a_{g-1})\rangle\ldots\lambda^3\langle P(a_1)\rangle\lambda \end{aligned} \tag{4}$$

where

$$\langle P(\phi)\rangle \;=\; \begin{cases} d\lambda d^{4i-3} & \text{if } \phi = a_i \\ \lambda d^{4mg}d\lambda d^{4k-3} & \text{if } \phi = e_ja_i,\ e_ja_i \rightarrow a_ke_m \\ d^{4mg}d\lambda d^{4k-3}d\lambda d^{4v-3} & \text{if } \phi = e_ja_i,\ e_ja_i \rightarrow a_va_ke_m \end{cases} \tag{5}$$

where $a_i, a_k, a_v \in A$ and $e_j, e_m \in E$.

Finally, the position of $U_{5,6}$'s tape head is over the symbol immediately to the right of $\langle R\rangle d^{*}$.

In the encoding of an *initial configuration* of R the d^{*} term in Equation (3) is replaced with the empty word ϵ.

We wish to highlight an abuse of notation. Recall from Section 4 that $P(\cdot) \in A \cup A \times E \cup A \times A \times E$. From Defintion 5 we have $\langle a_i\rangle, \langle e_j\rangle \in \{\overleftarrow{b}\}^{*}$, however in Equation (5) we have $\langle P(a_i)\rangle, \langle P(e_ja_i)\rangle \in \{d, \lambda\}^{*}$. In the sequel this ambiguity will be made clear from the context.

5.1 $U_{5,6}$ algorithm overview

Here we give a brief description of the simulation algorithm by explaining how $U_{5,6}$ locates and simulates a production. Using a unary indexing method, $U_{5,6}$ locates the encoded production to be simulated. The encoded production ($\langle P(\phi)\rangle$ from Equation (5)) is indexed (pointed to) by the number of $\overleftarrow{b}$ symbols contained in the leftmost encoded symbol or pair of symbols (Definition 5). If the leftmost encoded symbol is $\langle a_i \rangle = \overleftarrow{b}^{4i-3}$ then the value $4i-3$ is used to index $\langle P(\phi)\rangle$. If the leftmost encoded symbol is $\langle e_j \rangle = \overleftarrow{b}^{4jg}$, and $\langle a_i \rangle = \overleftarrow{b}^{4i-3}$ is adjacent then the value $4jg+4i-3$ is used to index $\langle P(\phi)\rangle$. The number of $\overleftarrow{b}$ symbols in the encoded symbol or pair of symbols is equal to the number of λ markers between the leftmost encoded symbol and $\langle P(\phi)\rangle$. To locate $\langle P(\phi)\rangle$, $U_{5,6}$ simply neutralises the rightmost λ in $\langle R\rangle$ (i.e. changes λ to δ) for each $\overleftarrow{b}$ in the leftmost encoded symbol or pair of symbols. This process continues until the marker c that separates two encoded symbols is read. This indexed production $\langle P(\phi)\rangle$ is then printed immediately to the right of encoded data word. This printing completes the execution of $\langle P(\phi)\rangle$.

Definition 7 **($U_{5,6}$).** *The TM $U_{5,6}$ is defined as $U_{5,6} = (\{u_1, u_2, u_3, u_4, u_5\}, \{\overrightarrow{b}, \overleftarrow{b}, \lambda, \delta, c, d\}, d, f, u_1, \{u_2, u_3, u_4, u_5, u_6\})$ where f is given by the following transition rules.*

$u_1, \overrightarrow{b}, \overleftarrow{b}, R, u_1$	$u_2, \overrightarrow{b},$	$u_3, \overrightarrow{b},$
$u_1, \overleftarrow{b}, \overrightarrow{b}, L, u_1$	$u_2, \overleftarrow{b}, \overleftarrow{b}, L, u_2$	$u_3, \overleftarrow{b}, \overleftarrow{b}, R, u_3$
$u_1, \lambda, \delta, R, u_1$	$u_2, \lambda, \delta, L, u_5$	$u_3, \lambda,$
$u_1, \delta, \lambda, L, u_1$	$u_2, \delta, \delta, L, u_2$	$u_3, \delta, \delta, R, u_3$
u_1, c, c, L, u_2	u_2, c, c, L, u_2	u_3, c, c, R, u_3
$u_1, d, \overrightarrow{b}, L, u_1$	$u_2, d, \overleftarrow{b}, R, u_4$	u_3, d, c, L, u_2

$u_4, \overrightarrow{b},$	$u_5, \overrightarrow{b},$
$u_4, \overleftarrow{b}, \overleftarrow{b}, R, u_4$	$u_5, \overleftarrow{b}, d, R, u_5$
$u_4, \lambda,$	$u_5, \lambda, \lambda, R, u_5$
$u_4, \delta, \delta, R, u_4$	$u_5, \delta, \lambda, R, u_5$
u_4, c, c, R, u_4	u_5, c, d, R, u_1
$u_4, d, \overleftarrow{b}, L, u_2$	$u_5, d, \overleftarrow{b}, R, u_3$

We give an example of $U_{5,6}$ simulating a production. This simulation is of a $P(a_i)$ production on an arbitrary data word. In this example we present $U_{5,6}$'s algorithm as three cycles. In the configurations below the current state of $U_{5,6}$ is highlighted in bold font to the left of $U_{5,6}$'s tape contents. The position of $U_{5,6}$'s tape head is given by an underline. The start state of $U_{5,6}$ is u_1 and $U_{5,6}$'s tape head is over the symbol directly to the right of $\langle R\rangle$ (as in Definition 6).

Example 2 *($U_{5,6}$ simulating production $P(a_i)$).* In this example R's data word is $a_i A^* E A^*$. Encoding this data word using Definitions 5 and 6 gives the word $\overleftarrow{b}^{4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*$. This gives a configuration of $U_{5,6}$ of the form:
$\boldsymbol{u_1}$ $\overrightarrow{b} \ldots \lambda^3\langle P(a_i)\rangle\lambda^3 \ldots \lambda^3\langle P(a_1)\rangle\lambda d^*\underline{\overleftarrow{b}}\,\overleftarrow{b}^{4i-4}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd \ldots$

Cycle 1 (index next production)

$$u_1, \overrightarrow{b}, \overleftarrow{b}, R, u_1$$
$$u_1, \overleftarrow{b}, \overrightarrow{b}, L, u_1$$
$$u_1, \lambda, \delta, R, u_1$$
$$u_1, \delta, \lambda, L, u_1$$
$$u_1, c, c, L, u_2$$
$$u_1, d, \overrightarrow{b}, L, u_1$$

In Cycle 1, $U_{5,6}$ reads the leftmost encoded symbol and locates the next encoded production to execute. Then $U_{5,6}$'s tape head scans from left to right and when it reads a $\overleftarrow{b}$ it changes it to a $\overrightarrow{b}$. Then $U_{5,6}$'s tape head scans left to neutralise a λ marker (changes it to a δ). This process is repeated until $U_{5,6}$ reads a c. This signals the end of Cycle 1 and the beginning of Cycle 2. In the configurations below we replace $\langle P(a_i)\rangle$ with $d\lambda d^{4i-3}$ and $\langle P(a_1)\rangle$ with $d\lambda d$, their respective encodings via Equation (5). After the initial configuration we have:
$\boldsymbol{u_1}\ \overrightarrow{b} \ldots \lambda^3 d\lambda d^{4i-3}\lambda^3 \ldots \lambda^3 d\lambda d\lambda\underline{d^*}\,\overrightarrow{b}\,\overleftarrow{b}^{4i-4}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd\ldots$
$\boldsymbol{u_1}\ \overrightarrow{b} \ldots \lambda^3 d\lambda d^{4i-3}\lambda^3 \ldots \lambda^3 d\lambda d\underline{\lambda}\,\overrightarrow{b}^*\,\overrightarrow{b}\,\overleftarrow{b}^{4i-4}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd\ldots$
$\boldsymbol{u_1}\ \overrightarrow{b} \ldots \lambda^3 d\lambda d^{4i-3}\lambda^3 \ldots \lambda^3 d\lambda d\delta\underline{\overrightarrow{b}^*}\,\overrightarrow{b}\,\overleftarrow{b}^{4i-4}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd\ldots$
$\boldsymbol{u_1}\ \overrightarrow{b} \ldots \lambda^3 d\lambda d^{4i-3}\lambda^3 \ldots \lambda^3 d\lambda d\delta\overleftarrow{b}^*\,\overleftarrow{b}\,\underline{\overleftarrow{b}^{4i-4}}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd\ldots$
In the configuration immediately above a λ marker has been neutralised (changed to a δ). The above process is repeated until the c to the right of $\langle a_i\rangle = \overleftarrow{b}^{4i-3}$ is read.
$\boldsymbol{u_1}\ \overrightarrow{b} \ldots \lambda^3 d\lambda d^{4i-3}\delta^3 \ldots \delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^*\,\overleftarrow{b}^{4i-3}\underline{c}(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd\ldots$
$\boldsymbol{u_2}\ \overrightarrow{b} \ldots \lambda^3 d\lambda d^{4i-3}\delta^3 \ldots \delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^*\,\underline{\overleftarrow{b}^{4i-3}}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd\ldots$
In the configuration immediately above $U_{5,6}$ has entered Cycle 2. The encoded production $\langle P(a_1)\rangle$ was located by neutralising $4i-3$ markers.

Cycle 2 (print production)

$u_2, \overleftarrow{b}, \overleftarrow{b}, L, u_2$	$u_4, \overleftarrow{b}, \overleftarrow{b}, R, u_4$	$u_3, \overleftarrow{b}, \overleftarrow{b}, R, u_3$	$u_5, \lambda, \lambda, R, u_5$
$u_2, \lambda, \delta, L, u_5$	$u_4, \delta, \delta, R, u_4$	$u_3, \delta, \delta, R, u_3$	$u_5, d, \overleftarrow{b}, R, u_3$
$u_2, \delta, \delta, L, u_2$	u_4, c, c, R, u_4	u_3, c, c, R, u_3	
u_2, c, c, L, u_2	$u_4, d, \overleftarrow{b}, L, u_2$	u_3, d, c, L, u_2	
$u_2, d, \overleftarrow{b}, R, u_4$			

Cycle 2 prints the encoded production indexed in Cycle 1 immediately to the right of the encoded data word. $U_{5,6}$ scans left in state u_2 and records the next symbol of the encoded production to be printed. If $U_{5,6}$ reads a d it enters state u_4 scans right and prints a $\overleftarrow{b}$ at the right end of the encoded data word. If $U_{5,6}$ reads the subword $d\lambda$ it scans right in state u_3 and prints a c at the right end of the encoded data word. This process is repeated until the end of the encoded production is detected causing $U_{5,6}$ to enter Cycle 3. The end of the encoded production is detected by $U_{5,6}$

reading the subword $\lambda\lambda$.

$\boldsymbol{u_2}\ \overrightarrow{b}\ldots\lambda^3 d\lambda d^{4i-4}\underline{d}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd\ldots$

$\boldsymbol{u_4}\ \overrightarrow{b}\ldots\lambda^3 d\lambda d^{4i-4}\overleftarrow{b}\,\underline{\delta^3}\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* dd\ldots$

$\boldsymbol{u_4}\ \overrightarrow{b}\ldots\lambda^3 d\lambda d^{4i-4}\overleftarrow{b}\,\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^* \underline{d}d\ldots$

$\boldsymbol{u_2}\ \overrightarrow{b}\ldots\lambda^3 d\lambda d^{4i-4}\overleftarrow{b}\,\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle\underline{(\langle A\rangle c)^*}\,\overleftarrow{b}\,d\ldots$

$\boldsymbol{u_2}\ \overrightarrow{b}\ldots\lambda^3 d\lambda \underline{d^{4i-4}}\,\overleftarrow{b}\,\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}\,d\ldots$

In the configuration immediately above a $\overleftarrow{b}$ is printed at the right end of the encoded data word. Immediately after the above process is repeated a futher $4i-4$ times we have the following:

$\boldsymbol{u_2}\ \overrightarrow{b}\ldots\lambda^3 d\underline{\lambda}\,\overleftarrow{b}^{\,4i-3}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}dd\ldots$

$\boldsymbol{u_5}\ \overrightarrow{b}\ldots\lambda^3 \underline{d}\delta\,\overleftarrow{b}^{\,4i-3}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}dd\ldots$

$\boldsymbol{u_3}\ \overrightarrow{b}\ldots\lambda^3\overleftarrow{b}\,\underline{\delta}\,\overleftarrow{b}^{\,4i-3}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}dd\ldots$

$\boldsymbol{u_3}\ \overrightarrow{b}\ldots\lambda^3\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,4i-3}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}\underline{d}d\ldots$

$\boldsymbol{u_2}\ \overrightarrow{b}\ldots\lambda^3\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,4i-3}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\underline{\overleftarrow{b}^{\,4i-3}}cd\ldots$

In the configuration immediately above a c marker is printed at the right end of the encoded data word. On the next scan left $U_{5,6}$ encounters the subword $\lambda\lambda$ which causes $U_{5,6}$ to enter Cycle 3.

$\boldsymbol{u_2}\ \overrightarrow{b}\ldots\lambda\lambda\underline{\lambda}\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,4i-3}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}cd\ldots$

$\boldsymbol{u_5}\ \overrightarrow{b}\ldots\lambda\underline{\lambda}\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,4i-3}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}cd\ldots$

$\boldsymbol{u_5}\ \overrightarrow{b}\ldots\lambda\lambda\underline{\delta}\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,4i-3}\delta^3\ldots\delta^3\overleftarrow{b}\,\delta\,\overleftarrow{b}\,\delta\,\overleftarrow{b}^{\,*}\,\overleftarrow{b}^{\,4i-3}c(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}cd\ldots$

Cycle 3 (restore tape)

$$u_5, \overleftarrow{b}, d, R, u_5$$
$$u_5, \delta, \lambda, R, u_5$$
$$u_5, c, d, R, u_1$$

Cycle 3 restores R's encoded table of behaviour after an encoded production has been indexed and printed. $U_{5,6}$ scans right restoring each $\overleftarrow{b}$ to a d and each δ to a λ. When $U_{5,6}$ reads the c at the left end of the encoded data word, it changes this c to a d. This cycle is now complete and Cycle 1 is entered.

$\boldsymbol{u_5}\ \overrightarrow{b}\ldots\lambda^3 d\lambda d^{4i-3}\lambda^3\ldots\lambda^3 d\lambda d\lambda d^* d^{4i-3}\underline{c}(\langle A\rangle c)^*\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}cd\ldots$

$\boldsymbol{u_1}\ \overrightarrow{b}\ldots\lambda^3 d\lambda d^{4i-3}\lambda^3\ldots\lambda^3 d\lambda d\lambda d^* d^{4i-3}d\underline{(\langle A\rangle c)^*}\langle E\rangle(\langle A\rangle c)^*\,\overleftarrow{b}^{\,4i-3}cd\ldots \quad \text{(I)}$

The simulation of the encoded production is now complete. $U_{5,6}$ is ready to begin the indexing and execution of the next encoded production. Configuration (I) satisfies Definition 6. □

Remark 1. In Configuration (I) we see that the word d^* separating $\langle R\rangle$ and the encoded data word increased in length by $4i-3$. During simulation of each production the word encoding the leftmost symbol is changed to a word contsisting only of ds. Hence the number of ds that separate $\langle R\rangle$ and the encoded data word increases as the simulation continues.

Example 3 *($U_{5,6}$ simulating production $P(e_j a_i)$).* In this example R's data word is $e_j a_i A^*$. Encoding this data word using Definitions 5 and 6 gives the word $\overleftarrow{b}^{4jg}\overleftarrow{b}^{4i-3}c(\langle A\rangle c)^*$. This gives a configuration of $U_{5,6}$ of the form:

$$\boldsymbol{u_1}\ \overrightarrow{b}\ldots\lambda^2\langle P(e_j a_i)\rangle\ldots\lambda^3\langle P(a_1)\rangle\lambda d^*\underline{\overleftarrow{b}^{4jg}}\overleftarrow{b}^{4i-3}c(\langle A\rangle c)^* dd\ldots \qquad \text{(II)}$$

The production $P(e_j a_i) = a_k e_m$ is encoded as $\langle P(e_j a_i)\rangle = \lambda d^{4mg} d\lambda d^{4k-3}$ via Equation (5). Thus from the configuration immediately above get:

$$\boldsymbol{u_1}\ \overrightarrow{b}\ldots\lambda^2\lambda d^{4mg}d\lambda d^{4k-3}\ldots\lambda^3 d\lambda d\lambda d^*\underline{\overleftarrow{b}^{4jg}}\overleftarrow{b}^{4i-3}c(\langle A\rangle c)^* dd\ldots$$

$$\boldsymbol{u_1}\ \overrightarrow{b}\ldots\lambda^2\lambda d^{4mg}d\lambda d^{4k-3}\ldots\delta^3\overleftarrow{b}\,\delta\overleftarrow{b}\,\delta\overleftarrow{b}^*\overleftarrow{b}^{4jg}\overleftarrow{b}^{4i-3}\underline{c}(\langle A\rangle c)^* dd\ldots$$

In the configuration immediately above we skip to the end of Cycle 1. The indexing of $\langle P(e_j a_i)\rangle$ is now complete. In Configuration (II) no c marker separates the word $\overleftarrow{b}^{4jg} = \langle e_j\rangle$ and the word $\overleftarrow{b}^{4i-3} = \langle a_i\rangle$ hence both are used to index $\langle P(e_j a_i)\rangle$. Now $U_{5,6}$ prints $\langle P(e_j a_i)\rangle$ in the same manner as Example (2) to give:

$$\boldsymbol{u_5}\ \overrightarrow{b}\ldots\underline{\lambda^2}\delta\overleftarrow{b}^{4mg}\overleftarrow{b}\,\delta\overleftarrow{b}^{4k-3}\ldots\delta^3\overleftarrow{b}\,\delta\overleftarrow{b}\,\delta\overleftarrow{b}^*\overleftarrow{b}^{4jg}\overleftarrow{b}^{4i-3}c(\langle A\rangle c)^*\overleftarrow{b}^{4k-3}c\overleftarrow{b}^{4mg}dd\ldots$$

The configuration immediately above ends Cycle 2. The printing of $\langle P(e_j a_i)\rangle$ at the right end of the encoded data word is now complete. Note that a c marker is not printed to the right of the word $\overleftarrow{b}^{4mg} = \langle e_m\rangle$. Thus when this $\langle e_m\rangle$ becomes the leftmost encoded symbol, both its value and the encoded symbol immediately to it's right are used to index the appropriate encoded production. Skiping to the end of Cycle 3 gives:

$$\boldsymbol{u_5}\ \overrightarrow{b}\ldots\lambda^3 d^{4mg}d\lambda d^{4k-3}\ldots\lambda^3 d\lambda d\lambda d^* d^{4jg}d^{4i-3}\underline{d(\langle A\rangle c)^*}\overleftarrow{b}^{4k-3}c\overleftarrow{b}^{4mg}dd\ldots$$

The simulation of production $P(e_j a_i)$ is now complete. □

Lemma 3. *$U_{5,6}$ halts when the encoded halt symbol becomes the leftmost encoded symbol in the encoded data word.*

Proof. The halt symbol e_h is encoded via Definition 5 as $\langle e_h\rangle = \overleftarrow{b}^{4hg}$. The arbitrary data word $a_i A^*$ to the right of e_h is encoded via Definition 6 as $\overleftarrow{b}^{4i-3}c(\langle A\rangle c)^*$. When the halt symbol e_h is the leftmost symbol we get a configuration of the form:

$$\boldsymbol{u_1}\ \overrightarrow{b}c\lambda\ldots\lambda d^*\underline{\overleftarrow{b}^{4hg}}\overleftarrow{b}^{4i-3}c(\langle A\rangle c)^* dd\ldots$$

$$\boldsymbol{u_1}\ \overrightarrow{b}c\delta\ldots\delta\overleftarrow{b}^*\overleftarrow{b}^{4hg}\overleftarrow{b}\,\underline{\overleftarrow{b}}\,\overleftarrow{b}^{4i-5}c(\langle A\rangle c)^* dd\ldots \qquad \text{(III)}$$

In the configuration immediately above we skip to the point in the simulation where every λ in $\langle R\rangle$ is neutralised. The number of λ markers in $\langle R\rangle$ is $4hg+1$ (see Equation (4)). Hence every $\overleftarrow{b}$ in $\langle e_h\rangle$ and a single $\overleftarrow{b}$ from $\langle a_i\rangle$ are read to neutralise all λ markers in $\langle R\rangle$. When $U_{5,6}$ reads the second $\overleftarrow{b}$ in $\langle a_i\rangle$ it scans left to give:

$$\boldsymbol{u_1}\ \overrightarrow{b}\,\underline{c}\delta\ldots\delta\overrightarrow{b}^*\overrightarrow{b}^{4hg}\overrightarrow{b}\,\overrightarrow{b}\,\overleftarrow{b}^{4i-5}c(\langle A\rangle c)^* dd\ldots$$

$$\boldsymbol{u_2}\ \underline{\overrightarrow{b}}c\delta\ldots\delta\overrightarrow{b}^*\overrightarrow{b}^{4hg}\overrightarrow{b}\,\overrightarrow{b}\,\overleftarrow{b}^{4i-5}c(\langle A\rangle c)^* dd\ldots$$

There is no TR for the state-symbol pair $(u_2, \overrightarrow{b})$ so the computation halts. Above we assume that $\langle a_i\rangle$ has more than one $\overleftarrow{b}$. However a_1 is encoded by a single $\overleftarrow{b}$. This unique case is not a problem. In Configuration (III) a c is read by $U_{5,6}$ instead of a $\overleftarrow{b}$ sending $U_{5,6}$'s control in to state u_2. Then $U_{5,6}$ scans left in state u_2 and halts when it reads the $\overrightarrow{b}$. □

Lemma 4. *$U_{5,6}$ simulates any BTS.*

Proof. Examples 2 and 3 give simulations of the two types of productions. Example 2 is a simulation of an arbitrary $P(a_i)$ production on an arbitrary data word. Example 3 is specific in the sense that it is a production of the form $P(e_j a_i) = a_k e_m$. However Example 3 is easily extended to verify the other case of $P(e_j a_i) = a_v a_k e_m$ as it requires only the printing of a further A symbol. From Lemma 3 we know that when the encoded halt symbol becomes the leftmost symbol the computation of $U_{5,6}$ halts. Hence $U_{5,6}$ simulates the sequence of productions in a BTS's computation. The encoded output is easily decoded via Definition 5. □

Theorem 2. *$U_{5,6}$ uses space $O(|\Sigma|T^3(n)+|Q||\Sigma|^2T^2(n)+|Q|^2|\Sigma|^4)$ and time $O(|\Sigma|^2T^6(n)+|Q||\Sigma|^3T^5(n)+|Q|^2|\Sigma|^4T^4(n)+|Q|^2|\Sigma|^5T^3(n)+|Q|^3|\Sigma|^6T^2(n))$ to simulate TM M.*

Proof. From Theorem 1 the BTS R_M uses space $O(T(n))$ and time $O(T^3(n))$.

(Space). From the proof of Lemmata 1 and 2 the BTS R_M that simulates M via C_M has $O(|Q||\Sigma|)$ of the E symbols and $O(|\Sigma|)$ of the A symbols. There is a production for each A symbol and a production for each $(E-\{e_h\})\times A$ pair. Hence there are $O(|Q||\Sigma|^2)$ productions. From Equation (5) the length of an encoded production in R_M is $O(|Q||\Sigma|^2)$. Thus the space used to store R_M is $O(|Q|^2|\Sigma|^4)$.

The space C_M uses is $O(T(n))$. To encode C_M's tape contents and current state R_M uses $O(T(n))$ of its A symbols and a single E symbol. From the previous paragraph and Definition 5 $U_{5,6}$ uses $O(|\Sigma|)$ to store each A symbol and $O(|Q||\Sigma|^2)$ to store each E symbol. Thus $U_{5,6}$ requires $O(|\Sigma|T(n)+|Q||\Sigma|^2)$ space to store R_M's data word.

After each production has executed, the space between R_M's encoded table of behaviour and data word increases (Remark 1). For every $T(n)$ productions executed, $T(n)$ encoded A symbols and a single encoded E symbol are deleted increasing this space by $O(|\Sigma|T(n)+|Q||\Sigma|^2)$. After simulating $O(T^3(n))$ productions there is $O(|\Sigma|T^3(n)+|Q||\Sigma|^2T^2(n))$ space between R_M's encoded table of behaviour and data word. The space used to store R_M and its data word is $O(|\Sigma|T(n)+|Q|^2|\Sigma|^4)$. The space used by $U_{5,6}$ after simulating $O(T^3(n))$ productions is $O(|\Sigma|T^3(n)+|Q||\Sigma|^2T^2(n)+|Q|^2|\Sigma|^4)$.

(Time). Simulating a $P(ea)$ production involves 3 Cycles. (1) Index an encoded production by neutralising $O(|Q||\Sigma|^2)$ λs: $O(|Q||\Sigma|^3T^3(n)+|Q|^2|\Sigma|^4T^2(n)+|Q|^3|\Sigma|^6)$ steps. (2) Print an encoded production of length $O(|Q||\Sigma|^2)$: $O(|Q||\Sigma|^3T^3(n)+|Q|^2|\Sigma|^4T^2(n)+|Q|^3|\Sigma|^6)$ steps. (3) Restore $U_{5,6}$'s tape: $O(|\Sigma|T^3(n)+|Q||\Sigma|^2T^2(n)+|Q|^2|\Sigma|^4)$ steps. Thus $U_{5,6}$ simulates a single $P(ea)$ production of R_M in time $O(|Q||\Sigma|^3T^3(n)+|Q|^2|\Sigma|^4T^2(n)+|Q|^3|\Sigma|^6)$. Using a similar argument a single $P(a)$ production of length $O(|\Sigma|)$ is simulated by $U_{5,6}$ in time $O(|\Sigma|^2T^3(n)+|Q||\Sigma|^3T^2(n)+|Q|^2|\Sigma|^5)$. For every $O(T(n))$ prodcutions executed R_M executes only a single $P(ea)$ production. Hence $U_{5,6}$ simulates M in time $O(|\Sigma|^2T^6(n)+|Q||\Sigma|^3T^5(n)+|Q|^2|\Sigma|^4T^4(n)+|Q|^2|\Sigma|^5T^3(n)+|Q|^3|\Sigma|^6T^2(n))$. □

In summary, $U_{5,6}$ simulates TM M in time $O(T^6(n))$ and space $O(T^3(n))$ if we ignore the number of states and symbols of M in our analysis.

This result holds for more general definitions of TMs. For example, let M' be a deterministic multitape TM. M' would be converted to a single tape TM M. The state-symbol product of M would be only a constant times greater than the state-symbol product of M', also M would be at worst polynomially slower than M'. Thus, $U_{5,6}$ simulates M' in polynomial time. We get the following immediate corollary.

Corollary 1. *There are polynomial time UTMs in UTM(m, n) for all $m \geq 5$, $n \geq 6$.*

Cook [3, 17] has recently published UTMs in UTM$(2, 5)$, UTM$(3, 4)$, UTM$(4, 3)$ and UTM$(7, 2)$ that are smaller than Rogozhin et al's. However, Cooks machines also suffer from an exponential slowdown through simulation of 2-tag systems. Cook's UTMs differ from the classical Turing machine definition [5]. Instead of having a blank symbol these machines have a blank period. Cook's UTMs require the blank tape to have an infinitely repeating word to the left and a different infinitely repeating word to the right.

We have improved the state of the art in small efficient UTMs. Fig. 1 summarises our results. Our polynomial time UTM is smaller than those in [11]. Any reduction in the size of our UTM in terms of state-symbol product would result in a polynomial time UTM of the same size as Rogozhin et al's exponential time UTMs.

References

1. C. Baiocchi. Three small universal Turing machines. In Y. R. M. Margenstern, editor, *Machines, Computations, and Universality*, volume 2055 of *LNCS*, pages 1–10, Chisinau, Moldova, May 2001. MCU, 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, January 1964.
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 IEEE, editor, *Proceedings, Ninth Annual Symposium on Switching and Automata Theory*, pages 368–372, Schenectady, New York, October 1968. FOCS, IEEE Computer Society Press.
5. J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1979.
6. N. Ikeno. A 6-symbol 10-state universal Turing machine. In *Proceedings, Institute of Electrical Communications*, Tokyo, 1958.
7. M. Kudlek. Small deterministic Turing machines. *Theoretical Computer Science*, 168(2):241–255, 1996.
8. M. Kudlek and Y. Rogozhin. Small universal Turing and circular Post machines. *Pure Mathematics and Applications*, 13(1–2):197–210, 2002.
9. 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.
10. M. Minsky. *Computation finite and infinite machines*. Prentice-Hall, New Jersey, 1967.
11. T. Neary and D. Woods. Small fast universal Turing machines. Technical Report NUIM-CS-TR-2005-11, National university of Ireland, Maynooth, 2005.
12. L. Pavlotskaya. Solvability of the halting problem for certain classes of Turing machines. *Mathematical Notes Academy of Science USSR*, 13(6):537–541, June 1973.
13. L. Pavlotskaya. Sufficient conditions for the halting problem decidability of Turing machines (in Russian). *Avtomaty i Mashiny*, pages 91–118, 1978.
14. Y. Rogozhin. Small universal Turing machines. *Theoretical Computer Science*, 168(2):215–240, 1996.
15. P. van Emde Boas. *Handbook of theoretical computer science, volume A*, chapter 1, pages 1–66. Elsevier, 1990.
16. S. Watanabe. 5-symbol 8-state and 5-symbol 6-state universal Turing machines. *ACM*, 8(4):476–483, October 1961.
17. S. Wolfram. *A new kind of science*. Wolfram Media, Inc., 2002.