P-completeness of cellular automaton Rule 110*

Turlough Neary[1] and Damien Woods[2]

[1] TASS, 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

Abstract. We show that the problem of predicting t steps of the 1D cellular automaton Rule 110 is P-complete. The result is found by showing that Rule 110 simulates deterministic Turing machines in polynomial time. As a corollary we find that the small universal Turing machines of Mathew Cook run in polynomial time, this is an exponential improvement on their previously known simulation time overhead.

1 Introduction

In this paper we solve an open problem regarding the computational complexity of Rule 110 which is one of the simplest cellular automata. We show that the prediction problem for Rule 110 is P-complete. Rule 110 is a nearest neighbour, one dimensional, binary cellular automaton [1]. 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

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

The problem of Rule 110 prediction is defined as follows.

Definition 1 (Rule 110 prediction). *Given an initial Rule 110 configuration, a cell index i and a natural number t written in unary. Is cell p_i in state* 1 *at time t?*

This problem is in P as a Turing machine simulates the cellular automaton in $O(t^2)$ steps by repeatedly traversing from left to right. From Matthew Cook's [2] result one infers a NC lower bound on the problem. Cook showed that Rule 110 simulates Turing machines via the following sequence of simulations

$$\text{Turing machine} \mapsto \text{2-tag system} \mapsto \text{cyclic tag system} \mapsto \text{Rule 110} \qquad (1)$$

* BCRI preprint 04/2006, April 2006. http://www.bcri.ucc.ie/

where $A \mapsto B$ denotes that A is simulated by B. The universality of 2-tag systems [3] is well-known and Cook supplied the latter two simulations. Each of these simulations runs in polynomial time (that is, B runs in a number of steps that is polynomial in the number of A's steps) with the exception of the exponentially slow 2-tag system simulation of Turing machines [3]. This slowdown is due to the 2-tag system's unary encoding of Turing machine tape contents. Thus via Equation (1), Rule 110 is an exponentially slow simulator of Turing machines and so it has remained open as to whether RULE 110 PREDICTION is P-complete.

In this work we replace the tag system with a *clockwise Turing machine* to give the following chain of simulations

$$\begin{aligned} &\text{Turing machine} \mapsto \text{clockwise Turing machine} \\ &\mapsto \text{cyclic tag system} \mapsto \text{Rule 110} \end{aligned} \tag{2}$$

Each simulation runs in polynomial time and the reduction from Turing machine to Rule 110 is computable by a logspace Turing machine. Thus our work shows that Rule 110 simulates Turing machines efficiently, giving the following result.

Theorem 1. RULE 110 PREDICTION is logspace complete for P.

Rule 110 is a very simple (2 state, nearest neighbour, one dimensional) cellular automaton, and Matthew Cook [2] gave four small universal Turing machines[3] that simulate its computation. Their size given as (number of states, number of symbols), are respectively $(2, 5)$,$(3, 4)$,$(4, 3)$ and $(7, 2)$. In terms of program size these machines are a significant improvement on previous small universal Turing machine results [4–9]. However in terms of time complexity Cook's machines offer no improvement over the exponentially slow machines of Rogozhin et al. [4–7]. A corollary of our work is that Matthew Cook's small universal Turing machines are polynomial time simulators of Turing machines.

The prediction problem for a number of classes of cellular automata has been shown to be P-complete. However Rule 110 is the simplest so far, in the sense that previous P-completeness results have been shown for more general cellular automata (e.g. more states, neighbours or dimensions). For example prediction of cellular automata of dimension $d \geqslant 1$ with an arbitrary number of states is known to be P-complete [10]. Lindgren and Nordahl [11] show that prediction for one dimensional nearest neighbour cellular automata is P-complete for seven states and Ollinger's result [12] improves this to six. If the update rule depends on the states of five neighbours then four states are sufficient [11, 10]. Moore [13] shows that prediction of binary majority voting cellular automata is P-complete for dimension $d \geqslant 3$. On the other hand, the prediction problem for a variety

[3] Cook's small "universal Turing machines" deviate from the usual Turing machine definition in the following way: their blank tape consists of an infinitely repeated word to the left and another to the right. Intuitively this change of definition seems to make quite a difference to program size, especially since Cook encodes a program in one of these repeated words. This has no bearing on our P-completeness result as we require only a bounded initial configuration for RULE 110 PREDICTION.

of linear and quasilinear cellular automata is in NC [14, 15]. The question of whether RULE 110 PREDICTION is P-complete has been asked, either directly or indirectly, in a number of previous works (for example [14–16]).

2 Clockwise Turing machines

A clockwise Turing machine is like a standard single-tape Turing machine [17] except for the following details: (i) the tape is assumed to be circular, (ii) the tape head moves only clockwise on the tape, (iii) the machine's transition function is of the form $f : Q \times \Sigma \to (\Sigma \cup \Sigma\Sigma) \times Q$. Here Q and Σ are the machine's finite set of states and tape symbols respectively. A transition rule $t = (q_x, \sigma, v, q_y) \in f$, is executed as follows. If the write value v is an element of Σ then the tape cell containing the read symbol is overwritten by this value and the head moves clockwise to the next cell. Otherwise if $v \in \Sigma\Sigma$ then the tape cell containing the read symbol is replaced with two cells that each contain one of v's symbols and the head moves clockwise to the next cell.

It is not difficult to give a clockwise Turing machine R_M that simulates a single-tape Turing machine M with a quadratic time overhead. We can think of M's right moves as clockwise moves by R_M with $v \in \Sigma$. However if M is increasing its tape length by reading a blank symbol and moving right, then we proceed differently. In this case R_M inserts two symbols, $v = \sigma r$, where σ is M's write symbol. Then R_M moves clockwise, traversing the entire tape, until it meets the 'rightmost end of tape marker' symbol r. If M runs in time $T(n)$ then R_M simulates a right move by M in $O(T(n))$ time.

A left move (when reading a non-blank symbol) by M is simulated by a single traversal of the circular tape that leaves a marker and then shifts each symbol one step clockwise. Upon reaching the marker the left move simulation is complete. A left move by M, when reading a blank at the leftmost tape end, is simulated using a similar strategy to that above. Proof details are to be found in a previous paper [8].

Lemma 1. *Let M be a single-tape Turing machine that runs in time $T(n)$. Then there is a clockwise Turing machine R_M that simulates the computation of M in time $O(T^2(n))$.*

In the next section we prove that cyclic tag systems simulate clockwise Turing machines. In order to simplify this proof we state the result for clockwise Turing machines that have a binary tape alphabet $\Sigma = \{a, b\}$. As with standard Turing machines, using a binary alphabet causes at most a constant factor increase in the time, space and number of states.

3 Cyclic tag systems

Cyclic tag systems were used by Cook [2] to show that Rule 110 is universal.

Definition 2 (cyclic tag system). *A cyclic tag system $C = \alpha_0, \ldots, \alpha_{p-1}$, is a list of binary words $\alpha_m \in \{0, 1\}^*$ called appendants.*

A *configuration* of a cyclic tag system consists of (i) a *marker* that points to a single appendant α_m in C, and (ii) a word $w = w_0 \dots w_{|w|-1} \in \{0,1\}^*$. We call w the *data word*. Intuitively the list C is a *program* with the marker pointing to instruction α_m. In the initial configuration the marker points to appendant α_0 and w is the binary input word.

Definition 3 (computation step of a cyclic tag system). *A computation step is deterministic and acts on a configuration in one of two ways:*

- *If $w_0 = 0$ then w_0 is deleted and the marker moves to appendant $\alpha_{(m+1 \mod p)}$.*
- *If $w_0 = 1$ then w_0 is deleted, the word α_m is appended onto the right end of w, and the marker moves to appendant $\alpha_{(m+1 \mod p)}$.*

We write $c_1 \vdash c_2$ when configuration c_2 is obtained from c_1 via a single computation step. We let $c_1 \vdash^i c_2$ denote a sequence of exactly i computation steps. A cyclic tag system completes its computation if (i) the data word is the empty word or (ii) it enters a forever repeating sequence of configurations. The complexity measures of time and space are defined in the obvious way.

Example 1. (cyclic tag system computation) Let $C = 00, 01, 11$ be a cyclic tag system with input word 011. Below we give the first four steps of the computation. In each configuration C is given on the left with the marked appendant highlighted in bold font.

$$\begin{array}{rllllllll} & \mathbf{00}, 01, 11 & 011 & \vdash & 00, \mathbf{01}, 11 & 11 & \vdash & 00, 01, \mathbf{11} & 101 \\ \vdash & \mathbf{00}, 01, 11 & 0111 & \vdash & 00, \mathbf{01}, 11 & 111 & \vdash & \dots & \end{array}$$

3.1 Cyclic tag systems simulate clockwise Turing machines

Much of the proof of Theorem 1 is given by the folowing lemma.

Lemma 2. *Let R be a binary clockwise Turing machine with $|Q|$ states that runs in time $T(n)$. Then there is a cyclic tag system C_R that simulates the computation of R in time $O(|Q|T^2(n) \log T(n))$.*

Proof. Let $R = (Q, \{a, b\}, f, q_1, q_{|Q|})$ where $Q = \{q_1, \dots, q_{|Q|}\}$ are the states, $\{a, b\}$ is the binary alphabet, f is the transition function, and $q_1, q_{|Q|} \in Q$ are the initial and final states respectively. In the sequel $\sigma_j \in \{a, b\}$. The bulk of the proof is concerned with simulating a single (but arbitrary) transition rule of R.

Encoding We define the cyclic tag system (program) to be of the form $C_R = \alpha_0, \dots, \alpha_{2z-1}$ where $z = 30|Q| + 61$. Given an initial configuration of R (consisting of current state $q_i \in Q$, read symbol σ_1, and tape contents $\sigma_1 \dots \sigma_s \in \{a, b\}^*$) we encode this as a configuration of C_R as follows

$$\boldsymbol{\alpha_0}, \dots, \alpha_{2z-1} \quad \langle 1, q_i \rangle \langle \sigma_1 \rangle \dots \langle \sigma_s \rangle \mu^{s'} \tag{3}$$

Here $\mu = 10^{z-1}$ and

$$s' = 2^{\lceil \log_2 s \rceil} \tag{4}$$

encoded object	encoded object length	initial marker index	index y of appendant	appendant α_y
$\langle 1, q_i\rangle = 0^{30i+20}10^{2z-30i-21}$	$2z$	0	$30i+20$	$\langle\, 1', q_i\rangle$
$\langle 1, q_i\rangle = 0^{30i+20}10^{2z-30i-21}$	$2z$	z	$z+30i+20$	$0^z\langle 1', q_{i,s<s'}\rangle$
$\langle 1, q_{i,s<s'}\rangle = 0^{30i+25}10^{2z-30i-26}$	$2z$	0	$30i+25$	$\langle 1', q_{i,s<s'}\rangle$
$\langle 1, q_{i,s<s'}\rangle = 0^{30i+25}10^{2z-30i-26}$	$2z$	z	$z+30i+25$	$0^z\langle 1', q_{i,s<s'}\rangle$
$\langle a\rangle = 010^{2z-2}$	$2z$	0	1	$\langle a\rangle$
$\langle a\rangle = 010^{2z-2}$	$2z$	z	$z+1$	$\langle a\rangle$
$\langle b\rangle = 0^210^{2z-3}$	$2z$	0	2	$\langle b\rangle$
$\langle b\rangle = 0^210^{2z-3}$	$2z$	z	$z+2$	$\langle b\rangle$
$\langle \not{a}\rangle = 0^310^{2z-4}$	$2z$	0	3	$\langle \not{a}\rangle$
$\langle \not{a}\rangle = 0^310^{2z-4}$	$2z$	z	$z+3$	$\langle \not{a}\rangle$
$\langle \not{b}\rangle = 0^410^{2z-5}$	$2z$	0	4	$\langle \not{b}\rangle$
$\langle \not{b}\rangle = 0^410^{2z-5}$	$2z$	z	$z+4$	$\langle \not{b}\rangle$
$\mu = 10^{z-1}$	z	0	0	$\not{\mu}$
$\mu = 10^{z-1}$	z	z	z	μ'
$\mu' = 0^610^{2z-7}$	$2z$	0	6	μ'
$\mu' = 0^610^{2z-7}$	$2z$	z	$z+6$	μ'
$\not{\mu} = 0^510^{2z-6}$	$2z$	0	5	$\not{\mu}$
$\not{\mu} = 0^510^{2z-6}$	$2z$	z	$z+5$	$\not{\mu}$

Table 1.1. (Stage 1. Halve counter). Every second μ is marked off by being changed to $\not{\mu}$.

are used for a 'tape length' counter. The values of appendants α_j are given during the proof below. States q_i and tape symbols $\{a, b\}$ of R are encoded as:

$$\langle 1, q_i\rangle = 0^{30i+20}10^{2z-30i-21}$$
$$\langle a\rangle = 010^{2z-2}$$
$$\langle b\rangle = 0^210^{2z-3}$$

Our simulation algorithm consists of a number of stages. In a cyclic tag system configuration the current stage x is identifiable by the notation $\langle x, q_i\rangle$.

How to read the tables We define the cyclic tag system C_R via a number of tables that specify encoded objects (e.g. encoded symbols, states) in the data word and the appendants they map to. Each table row gives an "encoded object" followed by the "encoded object length". The "initial marker index" gives the location of the program marker immediately before the encoded object is read. Each encoded object indexes an appendant α_y, where y is specified by the "index y of appendant" column and α_y is specified by the "appendant α_y" column.

To aid the reader we carefully describe the initial steps in the simulation of a transition rule. We encode a configuration that is arbitrary except for its tape length (which is 3). Initially the marker is pointing at appendant α_0 and the data word is $\langle 1, q_i\rangle\langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\mu\mu\mu\mu \in \{0,1\}^{12z}$. The leftmost $2z$ symbols

in the data word encode the current state q_i. From Table 1.1 this is $\langle 1, q_i\rangle = 0^{30i+20}10^{2z-30i-21}$. The computation begins by deleting the $30i+20$ leftmost 0 symbols while moving the marker rightwards through the appendants, one step for each deletion. The leftmost data symbol is now 1, this is deleted and causes the appendant α_{30i+20} to be appended onto the rightmost end of the data word. From Table 1.1 we see that $\alpha_{30i+20} = \langle 1', q_i\rangle$. Then $2z-30i-21$ contiguous 0 symbols are deleted while moving the marker one step for each deletion. Since $|\langle 1, q_i\rangle| = 2z$ and there are exactly $2z$ appendants in C_R, the marker is once again positioned at α_0. We write these $2z$ steps as

$$\begin{aligned} & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} \quad \langle 1, q_i\rangle\langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\mu\mu\mu\mu \\ \vdash^{2z} \; & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} \quad \langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\mu\mu\mu\mu\langle 1', q_i\rangle \end{aligned}$$

Algorithm overview Our cyclic tag system algorithm has three stages. Stages 1 and 2 isolate the encoded read symbol of R which is located immediately to the right of $\langle 1, q_i\rangle$. These stages make use of the tape-length counter specified by Equations (3) and (4). In Stage 1 every second μ is marked and then in Stage 2 every second $\langle\sigma\rangle$ is marked. This process is iterated until all μ objects are marked ($1+\log_2 s'$ iterations). The first six configurations of Fig. 1 illustrate this process. The encoded read symbol is now isolated as it is the only unmarked encoded tape symbol. The computation then enters Stage 3 which uses the encoded current state and (isolated) encoded read symbol to index an appendant that encodes the write symbol(s) and next state. In the final two configurations of Fig. 1 the new encoded current state and write value are appended and the counter is doubled to maintain the equality in Equation (4).

Stage 1. Halve counter The counter value is specified by Equation (4) as the number of μ (or later, μ') objects. This value is halved by marking half of the μ objects (changing μ to $\not\mu$) using Table 1.1. In this table we see that $|\mu| = z$ so exactly two μ objects are read for a single traversal of the marker through all $2z$ appendants. Every second μ indexes $\not\mu$ and every other μ indexes μ'. The encoded state $\langle 1, q_i\rangle$ indexes $\langle 1', q_i\rangle$ or $\langle 1', q_{i,s<s'}\rangle$, which sends control to Table 1.2.

We continue the above simulation (we later generalise to an arbitrary number of tape symbols).

$$\begin{aligned} & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} \quad \langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\mu\mu\mu\mu\langle 1', q_i\rangle \\ \vdash^{2z} \; & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} \quad \langle\sigma_2\rangle\langle\sigma_3\rangle\mu\mu\mu\mu\langle 1', q_i\rangle\langle\sigma_1\rangle \\ \vdash^{4z} \; & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} \quad \mu\mu\mu\mu\langle 1', q_i\rangle\langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle \\ \vdash^{z} \; & \alpha_0, \ldots, \boldsymbol{\alpha_z}, \ldots, \alpha_{2z-1} \quad \mu\mu\mu\langle 1', q_i\rangle\langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\not\mu \\ \vdash^{z} \; & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} \quad \mu\mu\langle 1', q_i\rangle\langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\not\mu\mu' \\ \vdash^{2z} \; & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} \quad \langle 1', q_i\rangle\langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\not\mu\mu'\not\mu\mu' \end{aligned}$$

The algorithm tests if the counter is 0 by checking if exactly one unmarked μ was read. If so $\langle 3, q_i\rangle$ is appended and we enter Stage 3. Otherwise $\langle 2, q_i\rangle$ is appended and we enter Stage 2. Table 1.2 simulates this 'if' statement.

encoded object	encoded object length	initial marker index	index y of appendant	appendant α_y
$\langle 1', q_i\rangle = 0^{30i+21}10^{2z-30i-12}$	$2z+10$	0	$30i+21$	$\langle 2, q_i\rangle$
$\langle 1', q_i\rangle = 0^{30i+21}10^{2z-30i-12}$	$2z+10$	z	$z+30i+21$	$\langle 3, q_i\rangle$
$\langle 1', q_{i,s<s'}\rangle = 0^{30i+26}10^{2z-30i-17}$	$2z+10$	0	$30i+26$	$\langle 2, q_{i,s<s'}\rangle$
$\langle 1', q_{i,s<s'}\rangle = 0^{30i+26}10^{2z-30i-17}$	$2z+10$	z	$z+30i+26$	$\langle 3, q_{i,s<s'}\rangle$
$\langle a\rangle = 010^{2z-2}$	$2z$	10	11	$\langle a'\rangle$
$\langle a\rangle = 010^{2z-2}$	$2z$	$z+10$	$z+11$	$\langle a\rangle$
$\langle b\rangle = 0^210^{2z-3}$	$2z$	10	12	$\langle b'\rangle$
$\langle b\rangle = 0^210^{2z-3}$	$2z$	$z+10$	$z+12$	$\langle b\rangle$
$\langle \not a\rangle = 0^310^{2z-4}$	$2z$	10	13	$\langle \not a\rangle$
$\langle \not a\rangle = 0^310^{2z-4}$	$2z$	$z+10$	$z+13$	$\langle \not a\rangle$
$\langle \not b\rangle = 0^410^{2z-5}$	$2z$	10	14	$\langle \not b\rangle$
$\langle \not b\rangle = 0^410^{2z-5}$	$2z$	$z+10$	$z+14$	$\langle \not b\rangle$
$\mu' = 0^610^{2z-7}$	$2z$	10	16	μ'
$\mu' = 0^610^{2z-7}$	$2z$	$z+10$	$z+16$	μ'
$\not\mu = 0^510^{2z-6}$	$2z$	10	15	$\not\mu$
$\not\mu = 0^510^{2z-6}$	$2z$	$z+10$	$z+15$	$\not\mu$

Table 1.2. (Stage 1. Check counter value). Here $\langle 1', q_i\rangle$ or $\langle 1', q_{i,s<s'}\rangle$ is used to check if the counter is 0.

As we continue our simulation we note from Table 1.2 that the word $\langle 1', q_i\rangle$ is of length $2z+10$. Hence the marker is at appendant α_{10} after $\langle 1', q_i\rangle$ is read:

$$\begin{aligned} & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} && \langle 1', q_i\rangle\langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\not\mu\mu'\not\mu\mu' \\ \vdash^{2z+10} & \alpha_0, \ldots, \boldsymbol{\alpha_{10}}, \ldots, \alpha_{2z-1} && \langle\sigma_1\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\not\mu\mu'\not\mu\mu'\langle 2, q_i\rangle \\ \vdash^{14z} & \alpha_0, \ldots, \boldsymbol{\alpha_{10}}, \ldots, \alpha_{2z-1} && \langle 2, q_i\rangle\langle\sigma_1'\rangle\langle\sigma_2'\rangle\langle\sigma_3'\rangle\not\mu\mu'\not\mu\mu' \end{aligned}$$

Immediately above is the first configuration of Stage 2.

Stage 2. Mark half of the encoded tape symbols The ultimate aim of this stage is to isolate the encoded read symbol. Each iteration of this stage uses Table 2.1 to mark off every second (even numbered) encoded tape symbol $\langle\sigma_j\rangle$. As we continue our simulation we note from Table 2.1 that $|\langle 2, q_i\rangle| = 2z+10$. Hence the marker is at appendant α_{20} after reading $\langle 2, q_i\rangle$.

$$\begin{aligned} & \alpha_0, \ldots, \boldsymbol{\alpha_{10}}, \ldots, \alpha_{2z-1} && \langle 2, q_i\rangle\langle\sigma_1'\rangle\langle\sigma_2'\rangle\langle\sigma_3'\rangle\not\mu\mu'\not\mu\mu' \\ \vdash^{2z+10} & \alpha_0, \ldots, \boldsymbol{\alpha_{20}}, \ldots, \alpha_{2z-1} && \langle\sigma_1'\rangle\langle\sigma_2'\rangle\langle\sigma_3'\rangle\not\mu\mu'\not\mu\mu'0^{2z-20}\langle 1, q_i\rangle \\ \vdash^{z} & \alpha_0, \ldots, \boldsymbol{\alpha_{z+20}}, \ldots, \alpha_{2z-1} && \langle\sigma_2'\rangle\langle\sigma_3'\rangle\not\mu\mu'\not\mu\mu'0^{2z-20}\langle 1, q_i\rangle\langle\sigma_1\rangle \\ \vdash^{z} & \alpha_0, \ldots, \boldsymbol{\alpha_{20}}, \ldots, \alpha_{2z-1} && \langle\sigma_3'\rangle\not\mu\mu'\not\mu\mu'0^{2z-20}\langle 1, q_i\rangle\langle\sigma_1\rangle\langle\not\phi_2\rangle \\ \vdash^{9z} & \alpha_0, \ldots, \boldsymbol{\alpha_{z+20}}, \ldots, \alpha_{2z-1} && 0^{2z-20}\langle 1, q_i\rangle\langle\sigma_1\rangle\langle\not\phi_2\rangle\langle\sigma_3\rangle\not\mu\mu\not\mu\mu \\ \vdash^{2z-20} & \alpha_0, \ldots, \boldsymbol{\alpha_z}, \ldots, \alpha_{2z-1} && \langle 1, q_i\rangle\langle\sigma_1\rangle\langle\not\phi_2\rangle\langle\sigma_3\rangle\not\mu\mu\not\mu\mu \end{aligned}$$

encoded object	encoded object length	initial marker index	index y of appendant	appendant α_y
$\langle 2, q_i\rangle = 0^{30i+12}10^{2z-30i-3}$	$2z+10$	10	$30i+22$	$0^{2z-20}\langle 1, q_i\rangle$
$\langle 2, q_{i,s<s'}\rangle = 0^{30i+17}10^{2z-30i-8}$	$2z+10$	10	$30i+27$	$0^{2z-20}\langle 1, q_{i,s<s'}\rangle$
$\langle a'\rangle = 0^7 10^{z-8}$	z	20	27	$\langle a\rangle$
$\langle a'\rangle = 0^7 10^{z-8}$	z	$z+20$	$z+27$	$\langle \not{a}\rangle$
$\langle b'\rangle = 0^8 10^{z-9}$	z	20	28	$\langle b\rangle$
$\langle b'\rangle = 0^8 10^{z-9}$	z	$z+20$	$z+28$	$\langle \not{b}\rangle$
$\langle \not{a}\rangle = 0^3 10^{2z-4}$	$2z$	20	23	$\langle \not{a}\rangle$
$\langle \not{a}\rangle = 0^3 10^{2z-4}$	$2z$	$z+20$	$z+23$	$\langle \not{a}\rangle$
$\langle \not{b}\rangle = 0^4 10^{2z-5}$	$2z$	20	24	$\langle \not{b}\rangle$
$\langle \not{b}\rangle = 0^4 10^{2z-5}$	$2z$	$z+20$	$z+24$	$\langle \not{b}\rangle$
$\mu' = 0^6 10^{2z-7}$	$2z$	20	26	μ
$\mu' = 0^6 10^{2z-7}$	$2z$	$z+20$	$z+26$	μ
$\not{\mu} = 0^5 10^{2z-6}$	$2z$	20	25	$\not{\mu}$
$\not{\mu} = 0^5 10^{2z-6}$	$2z$	$z+20$	$z+25$	$\not{\mu}$

Table 2.1. (Stage 2. Mark half of the encoded tape symbols). Rows 3 to 6 are used to mark off every second $\langle\sigma\rangle$.

If we are simulating a transition rule that has write value from $\Sigma\Sigma = \{a, b\}^2$, and the tape length is a power of 2, then we must double the counter value in order to satisfy Equation (4). This doubling occurs in Stage 3. However the tape length test happens in Stage 1 using Table 1.1 as follows.

Suppose that the encoded tape length is not a power of 2 and thus $s < s'$. Then, on some iteration, Stage 2 reads an odd number, strictly greater than 1, of unmarked encoded tape symbols. If this occurs then $\langle 1, q_i\rangle$ indexes the appendant $\langle 1', q_{i,s<s'}\rangle$. To see this, notice that in Stage 2 the tape symbols a, b are encoded as $\langle a'\rangle, \langle b'\rangle$ where $|\langle a'\rangle| = |\langle b'\rangle| = z$. If C_R reads an odd number of these then the initial marker index is at z. Suppose otherwise that the encoded tape length is a power of 2. Then $\langle 1, q_i\rangle$ always indexes the appendant $\langle 1', q_i\rangle$ in Stage 1. In summary, if $\langle 1', q_{i,s<s'}\rangle$ is not appended before Stage 3 begins then the number of tape symbols is a power of 2 and $s = s'$.

The simulation continues as follows:

$$
\begin{aligned}
& \alpha_0, \ldots, \boldsymbol{\alpha_z}, \ldots, \alpha_{2z-1} && \langle 1, q_i\rangle\langle\sigma_1\rangle\langle\phi_2\rangle\langle\sigma_3\rangle\not{\mu}\mu\not{\mu}\mu \\
\vdash^{2z}\ & \alpha_0, \ldots, \boldsymbol{\alpha_z}, \ldots, \alpha_{2z-1} && \langle\sigma_1\rangle\langle\phi_2\rangle\langle\sigma_3\rangle\not{\mu}\mu\not{\mu}\mu 0^z\langle 1', q_{i,s<s'}\rangle \\
\vdash^{13z}\ & \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} && \langle 1', q_{i,s<s'}\rangle\langle\sigma_1\rangle\langle\phi_2\rangle\langle\sigma_3\rangle\not{\mu}\mu'\not{\mu}\not{\mu} \\
\vdash^{16z+10}\ & \alpha_0, \ldots, \boldsymbol{\alpha_{10}}, \ldots, \alpha_{2z-1} && \langle 2, q_{i,s<s'}\rangle\langle\sigma_1'\rangle\langle\phi_2\rangle\langle\sigma_3'\rangle\not{\mu}\mu'\not{\mu}\not{\mu} \\
\vdash^{14z+10}\ & \alpha_0, \ldots, \boldsymbol{\alpha_{20}}, \ldots, \alpha_{2z-1} && 0^{2z-20}\langle 1, q_{i,s<s'}\rangle\langle\sigma_1\rangle\langle\phi_2\rangle\langle\phi_3\rangle\not{\mu}\mu\not{\mu}\not{\mu} \\
\vdash^{17z-20}\ & \alpha_0, \ldots, \boldsymbol{\alpha_z}, \ldots, \alpha_{2z-1} && \langle 1', q_{i,s<s'}\rangle\langle\sigma_1\rangle\langle\phi_2\rangle\langle\phi_3\rangle\not{\mu}\not{\mu}\not{\mu}\not{\mu} \\
\vdash^{16z+10}\ & \alpha_0, \ldots, \boldsymbol{\alpha_{z+10}}, \ldots, \alpha_{2z-1} && \langle 3, q_{i,s<s'}\rangle\langle\sigma_1\rangle\langle\phi_2\rangle\langle\phi_3\rangle\not{\mu}\not{\mu}\not{\mu}\not{\mu}
\end{aligned}
$$

Immediately above is the first the configuration of Stage 3.

encoded object	encoded object length	initial marker index	index y of appendant	appendant α_y
$\langle 3, q_i\rangle = 0^{30i+14}10^{z+5}$	$z+30i+20$	$z+10$	$z+30i+24$	$0^{2z-30i-30}$
$\langle 3, q_{i,s<s'}\rangle = 0^{30i+19}10^{z+10}$	$z+30i+30$	$z+10$	$z+30i+29$	$0^{2z-30i-40}$
$\langle a\rangle = 010^{2z-2}$	$2z$	$30i+30$	$30i+31$	$\langle\sigma_p\rangle\langle 1, q_k\rangle$
$\langle a\rangle = 010^{2z-2}$	$2z$	$30i+30$	$30i+31$	$D\langle\sigma_p\rangle\langle\sigma_h\rangle\langle 1, q_k\rangle$
$\langle b\rangle = 0^2 10^{2z-3}$	$2z$	$30i+30$	$30i+32$	$\langle\sigma_p\rangle\langle 1, q_k\rangle$
$\langle b\rangle = 0^2 10^{2z-3}$	$2z$	$30i+30$	$30i+32$	$D\langle\sigma_p\rangle\langle\sigma_h\rangle\langle 1, q_k\rangle$
$\langle a\rangle = 010^{2z-2}$	$2z$	$30i+40$	$30i+41$	$\langle\sigma_p\rangle\langle 1, q_k\rangle$
$\langle a\rangle = 010^{2z-2}$	$2z$	$30i+40$	$30i+41$	$\langle\sigma_p\rangle\langle\sigma_h\rangle\langle 1, q_k\rangle$
$\langle b\rangle = 0^2 10^{2z-3}$	$2z$	$30i+40$	$30i+42$	$\langle\sigma_p\rangle\langle 1, q_k\rangle$
$\langle b\rangle = 0^2 10^{2z-3}$	$2z$	$30i+40$	$30i+42$	$\langle\sigma_p\rangle\langle\sigma_h\rangle\langle 1, q_k\rangle$
$\langle \not{a}\rangle = 0^3 10^{2z-4}$	$2z$	$30i+30$	$30i+33$	$\langle a\rangle$
$\langle \not{a}\rangle = 0^3 10^{2z-4}$	$2z$	$30i+40$	$30i+43$	$\langle a\rangle$
$\langle \not{b}\rangle = 0^4 10^{2z-5}$	$2z$	$30i+30$	$30i+34$	$\langle b\rangle$
$\langle \not{b}\rangle = 0^4 10^{2z-5}$	$2z$	$30i+40$	$30i+44$	$\langle b\rangle$
$\not{\mu} = 0^5 10^{2z-6}$	$2z$	$30i+30$	$30i+35$	μ
$\not{\mu} = 0^5 10^{2z-6}$	$2z$	$30i+40$	$30i+45$	μ

Table 3.1. (Stage 3. Simulate transition rule). This table prints the encoded write value and establishes the new encoded current state $\langle 1, q_k\rangle$. If the counter does not need to be doubled this table completes simulation of the transition rule.

Stage 3. Complete simulation of transition rule In this stage an appendant α_y is indexed, based on the value of the encoded current state and encoded read symbol using Table 3.1. The printing of appendant α_y simulates the encoded write value, encoded next state, and the clockwise tape head movement.

Using Table 3.1 we read the encoded current state, either $\langle 3, q_i\rangle$ or $\langle 3, q_{i,s<s'}\rangle$, after which the initial marker index is either $30i+30$ or $30i+40$ respectively. The encoded read symbol was already isolated and uniquely retains its original value of $\langle a\rangle$ or $\langle b\rangle$; this value points at the appendant α_y (rows 3 to 10). All other (non-isolated) encoded tape symbols are of the form $\langle \not{a}\rangle$ or $\langle \not{b}\rangle$ and they point to the appendants $\langle a\rangle$ or $\langle b\rangle$ respectively.

The simulated transition rule is of the form $(q_i, \sigma_j, \sigma_p, q_k)$ or $(q_i, \sigma_j, \sigma_p\sigma_h, q_k)$, respectively encoded as the appendants $\langle\sigma_p\rangle\langle 1, q_k\rangle$ or $\langle\sigma_p\rangle\langle\sigma_h\rangle\langle 1, q_k\rangle$. In the present example we simulate the rule $(q_i, \sigma_1, \sigma_4, q_k)$:

$$
\begin{array}{rll}
 & \alpha_0, \ldots, \boldsymbol{\alpha_{z+10}}, \ldots, \alpha_{2z-1} & \langle 3, q_{i,s<s'}\rangle\langle\sigma_1\rangle\langle \not{\phi}_2\rangle\langle \not{\phi}_3\rangle \not{\mu}\not{\mu}\not{\mu}\not{\mu}\not{\mu} \\
\vdash^{z+30i+30} & \alpha_0, \ldots, \boldsymbol{\alpha_{30i+40}}, \ldots, \alpha_{2z-1} & \langle\sigma_1\rangle\langle \not{\phi}_2\rangle\langle \not{\phi}_3\rangle \not{\mu}\not{\mu}\not{\mu}\not{\mu}\not{\mu} 0^{2z-30i-40} \\
\vdash^{2z} & \alpha_0, \ldots, \boldsymbol{\alpha_{30i+40}}, \ldots, \alpha_{2z-1} & \langle \not{\phi}_2\rangle\langle \not{\phi}_3\rangle \not{\mu}\not{\mu}\not{\mu}\not{\mu}\not{\mu} 0^{2z-30i-40}\langle\sigma_4\rangle\langle 1, q_k\rangle \\
\vdash^{12z} & \alpha_0, \ldots, \boldsymbol{\alpha_{30i+40}}, \ldots, \alpha_{2z-1} & 0^{2z-30i-40}\langle\sigma_4\rangle\langle 1, q_k\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\mu\mu\mu\mu \\
 & \vdash^{2z-30i-40} \boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} & \langle\sigma_4\rangle\langle 1, q_k\rangle\langle\sigma_2\rangle\langle\sigma_3\rangle\mu\mu\mu\mu
\end{array}
$$

encoded object	encoded object length	initial marker index	index y of appendant	appendant α_y
$D = 0^{39}10^{2z}$	$2z + 40$	0	39	0^{2z-40}
$\langle 1, q_k \rangle = 0^{30k+20}10^{2z-30k-21}$	$2z$	40	$30k + 60$	$\langle 1, q_k \rangle$
$\langle a \rangle = 010^{2z-2}$	$2z$	40	41	$\langle a \rangle$
$\langle b \rangle = 0^2 10^{2z-3}$	$2z$	40	42	$\langle b \rangle$
$\mu = 10^{z-1}$	z	40	40	$\mu\mu$
$\mu = 10^{z-1}$	z	$z + 40$	$z + 40$	$\mu\mu$

Table 3.2. (Stage 3. Double counter). Each μ indexes the appendant $\mu\mu$.

Alternatively if the rule is of the form $(q_i, \sigma_1, \sigma_4\sigma_5, q_k)$ then the latter configuration is instead

$$\boldsymbol{\alpha_0}, \ldots, \alpha_{2z-1} \quad \langle \sigma_4 \rangle \langle \sigma_5 \rangle \langle 1, q_k \rangle \langle \sigma_2 \rangle \langle \sigma_3 \rangle \mu\mu\mu\mu$$

The simulation of the transition rule is now complete. The marker in C_R's program is at appendant α_0. The encoded write value is written, the new encoded state $\langle 1, q_k \rangle$ is established and the (clockwise) tape head movement is simulated.

We have given a sequence of configurations that explicitly simulate the application of a transition rule. We used arbitrary initial and next states $q_i, q_k \in Q$, and arbitrary tape symbols $\sigma_j \in \{a, b\}$.

The simulation is specific in the sense that the length of the tape data is fixed. The computation of C_R remains similar for any length of tape data that is not a power of 2. If the tape length is a power of 2, and thus $s = s'$, then C_R enters Stage 3 via $\langle 3, q_i \rangle$ instead of $\langle 3, q_{i,s<s'} \rangle$. On the one hand, if the tape data does not increase in length, the remainder of the computation proceeds in a similar manner to the above simulation. On the other hand, if the tape data increases in length [i.e. we are simulating a transition rule of the form $(q_i, \sigma_j, \sigma_p\sigma_h, q_k)$] then rows 4 or 6 of Table 3.1 are executed. The appendants in these rows contain the subword D. After reading D (using Table 3.2) the marker points at α_{40} which causes each μ in the counter to index the appendant $\mu\mu$. This doubles the counter's value and completes the simulation of the transition rule.

The simulation is also specific in the sense that the encoded state is the leftmost object in the data word when we begin simulating a transition rule. This generalises to an arbitrary encoded state position. To see this notice that the encoded state directs control flow of the algorithm through Stages 1 to 3. The order of executing the stages is unaffected by the relative *position* of the encoded state in the data word.

We have shown how C_R simulates an arbitrary transition rule of R. To simulate halting C_R enters a repeating sequence of configurations. The halt state $q_{|Q|}$ is encoded in the normal way as $\langle 1, q_{|Q|} \rangle = 0^{30|Q|+20}10^{2z-30|Q|-21}$. We define the appendant at index $30|Q| + 20$ to be $\langle 1, q_{|Q|} \rangle$. Therefore $\langle 1, q_{|Q|} \rangle$ indexes a copy of itself. Also after $\langle 1, q_{|Q|} \rangle$ is read, each encoded tape symbol indexes a copy of itself. This causes C_R to enter a forever repeating sequence of configurations.

Space analysis At time $T(n)$ there are $O(T(n))$ encoded objects (state and symbols) in C_R's data word; each of length $O(|Q|)$. Thus C_R uses $O(|Q|T(n))$ space.

Time analysis Simulating a transition rule involves 3 stages. Each stage executes in $O(|Q|T(n))$ steps. To simulate a single transition rule the counter is halved $O(\log T(n))$ times, (i.e. Stages 1 and 2 are executed $O(\log T(n))$ times) and Stage 3 is executed once. Thus $O(|Q|T(n)\log(T(n))$ time is sufficient to simulate a transition rule and $O(|Q|T^2(n)\log T(n))$ time is sufficient to simulate the computation of R. □

A consequence of the previous lemma is that Rule 110 simulates Turing machines in polynomial time. Matthew Cook's [2] universal Turing machines (see footnote on page 2) simulate Rule 110 in quadratic time, which in turn (using Cook's construction) simulates Turing machines in exponential time. We have improved this time bound to polynomial.

Corollary 1. *Matthew Cook's small universal Turing machines simulate Turing machines in polynomial time.*

Finally we show that the reduction from the GENERIC MACHINE SIMULATION PROBLEM (GMSP) [10] to RULE 110 PREDICTION is computable by a logspace transducer Turing machine. The GMSP is stated as: given a word x, an encoding $\langle M\rangle$ of a single-tape Turing machine M, and an integer t in unary, does M accept x within t steps?

Lemma 3. *The GMSP is logspace reducible to* RULE 110 PREDICTION.

Proof. From Section 2 the number of states of the binary clockwise Turing machine R_M is linear in the number of states and symbols of M. We encode these machines as words in a straightforward way such that for their lengths: $|\langle R_M\rangle| = O(|\langle M\rangle|)$. Also the input x_R to R_M is of length linear in $|x|$, the length of M's input. The conversion is clearly logspace computable.

We reduce the simulation problem for R_M to the analogous problem for cyclic tag systems. In the proof of Lemma 2 we showed how to construct C_{R_M}. We encode C_{R_M} as a word $\langle C_{R_M}\rangle$. The value z used in the proof of Lemma 2 is linear in $|Q|$, the number of states of R_M. There are $2z$ appendants, each of length $O(|Q|)$, giving an encoded program length of $O(|Q|^2)$. From Equation (3) the input $\langle x_R\rangle$ to C_{R_M} is of length $O(|Q||x_R|)$. Thus the encoded appendants and input are logspace constructable.

To show that a logspace transducer Turing machine generates a Rule 110 instance from $\langle C_{R_M}\rangle\#\langle x_R\rangle\#^t$ we examine Cook's Rule 110 simulation of cyclic tag systems [2]. The input is written directly as the states of $O(|\langle x_R\rangle|)$ contiguous cells beginning at, say, cell p_0. On the left of the input a constant word (representing Cook's 'ossifiers') is repeated $O(t)$ times. On the right the cyclic tag system program (list of appendants and 'leaders') is written $O(t)$ times. □

Since we already know that RULE 110 PREDICTION is in P, the proof of Theorem 1 is complete.

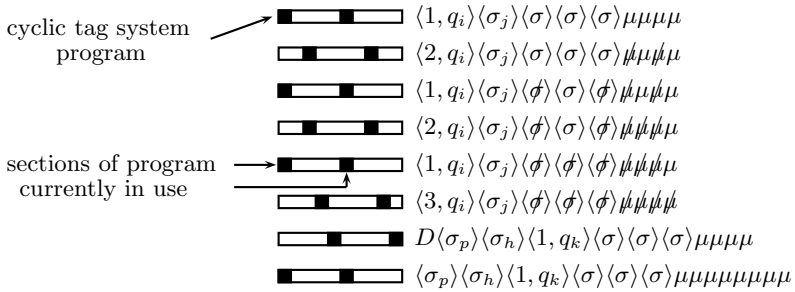

Fig. 1. Cyclic tag system simulation of transition rule $(q_i, \sigma_j, \sigma_p\sigma_h, q_k)$. The cyclic tag system program is illustrated on the left. In the data word the encoded current state $\langle x, q_i\rangle$ directs the control flow by determining the sections of the cyclic tag system program that are used in Stage x.

References

1. Wolfram, S.: Statistical mechanics of cellular automata. Reviews of Modern Physics **55** (1983) 601–644
2. Cook, M.: Universality in elementary cellular automata. Complex Systems **15** (2004) 1–40
3. Cocke, J., Minsky, M.: Universality of tag systems with $P = 2$. Journal of the ACM **11** (1964) 15–20
4. Rogozhin, Y.: Small universal Turing machines. TCS **168** (1996) 215–240
5. Baiocchi, C.: Three small universal Turing machines. In Margenstern, M., Rogozhin, Y., eds.: Machines, Computations, and Universality. Volume 2055 of LNCS., Chişinău, Moldova, MCU, Springer (2001) 1–10
6. Kudlek, M., Rogozhin, Y.: A universal Turing machine with 3 states and 9 symbols. In Kuich, W., Rozenberg, G., Salomaa, A., eds.: Developments in Language Theory (DLT) 2001. Volume 2295 of LNCS., Vienna, Springer (2002) 311–318
7. Minsky, M.: Size and structure of universal Turing machines using tag systems. In: Recursive Function Theory, Symp. in Pure Math. Volume 5., AMS (1962) 229–238
8. Neary, T., Woods, D.: A small fast universal Turing machine. Technical Report NUIM-CS-TR-2005-12, Dept. of Computer Science, NUI Maynooth (2005)
9. Neary, T., Woods, D.: Small fast universal Turing machines. TCS (To appear.)
10. Greenlaw, R., Hoover, H.J., Ruzzo, W.L.: Limits to parallel computation: P-completeness theory. Oxford university Press, Oxford (1995)
11. Lindgren, K., Nordahl, M.G.: Universal computation in simple one-dimensional cellular automata. Complex Systems **4** (1990) 299–318
12. Ollinger, N.: The quest for small universal cellular automata. In Widmayer, P., et al., eds.: International Colloquium on Automata, Languages and Programming (ICALP). Volume 2380 of LNCS., Malaga, Spain, Springer (2002) 318–329
13. Moore, C.: Majority-vote cellular automata, Ising dynamics and P-completeness. Journal of Statistical Physics **88** (1997) 795–805
14. Moore, C.: Quasi-linear cellular automata. Physica D **103** (1997) 100–132
15. Moore, C.: Predicting non-linear cellular automata quickly by decomposing them into linear ones. Physica D **111** (1998) 27–41
16. Aaronson, S.: Book review: A new kind of science. Quantum Information and Computation **2** (2002) 410–423
17. Hopcroft, J.E., Ullman, J.D.: Introduction to automata theory, languages, and computation. Addison-Wesley (1979)