The Collatz Process Embeds a Base Conversion Algorithm

Tristan Stérin(✉) and Damien Woods(✉)

Hamilton Institute, Department of Computer Science,
Maynooth University, Maynooth, Ireland
{tristan.sterin,damien.woods}@mu.ie
https://dna.hamilton.ie

Abstract. The Collatz process is defined on natural numbers by iterating the map $T(x) = T_0(x) = x/2$ when $x \in \mathbb{N}$ is even and $T(x) = T_1(x) = (3x+1)/2$ when x is odd. In an effort to understand its dynamics, and since Generalised Collatz Maps are known to simulate Turing Machines [Conway, 1972], it seems natural to ask what kinds of algorithmic behaviours it embeds. We define a quasi-cellular automaton that exactly simulates the Collatz process on the square grid: on input $x \in \mathbb{N}$, written horizontally in base 2, successive rows give the Collatz sequence of x in base 2. We show that vertical columns simultaneously iterate the map in base 3. This leads to our main result: the Collatz process embeds an algorithm that converts any natural number from base 3 to base 2. We also find that the evolution of our automaton computes the parity of the number of 1s in any ternary input. It follows that predicting about half of the bits of the iterates $T^i(x)$, for $i = O(\log x)$, is in the complexity class NC^1 but outside AC^0. These results show that the Collatz process is capable of some simple, but non-trivial, computation in bases 2 and 3, suggesting an algorithmic approach to thinking about prediction and existence of cycles in the Collatz process.

Keywords: Collatz map · Model of computation · Reachability problem

1 Introduction

The Collatz process is defined on natural numbers by iterating the map $T(x) = T_0(x) = x/2$ when $x \in \mathbb{N}$ is even and $T(x) = T_1(x) = (3x+1)/2$ when x is odd. The Collatz conjecture states that for all $x \in \mathbb{N}$ a finite number of iterations lead to 1. We know that 1-variable generalised Collatz maps (iterated linear equations of a single natural number variable with arbitrary mod) are capable of running arbitrarily algorithms [8], modulo an exponential simulation

Research supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No 772766, Active-DNA project, and Science Foundation Ireland (SFI), grant number 18/ERCS/5746.

scaling, which motivates our point of view in this paper: how complicated are the dynamics of the Collatz process? Perhaps the reason this process has resisted understanding is that it embeds algorithm(s) that solve problems with high computational complexity? Or perhaps by showing that this is not the case, we can get a handle on its dynamics?

We study the structure of iterations of the Collatz process both in binary and in ternary. We define a 2D quasi-cellular automaton (CQCA) that consists of a local rule, and a non-local[1] rule. A natural number is encoded as a binary string input to the CQCA, whose subsequent dynamics exactly execute the Collatz process with one horizontal row per odd iterate. Simultaneously, vertical columns simulate all iterations (both odd and even), but in base 3.

Results. Our main result, Theorem 11, is that the CQCA embeds a base conversion algorithm that can convert any natural number x from base 3 to base 2. This base conversion algorithm is natural and efficient, running in $\Theta(\log x)$ Collatz iterations. The result puts strict constraints on the short-term dynamics of the Collatz process and enables us to characterize the complexity of natural prediction problems on the dynamics of the CQCA: predicting about half of the bits of the iterates $T^i(x)$, for $i = O(\log x)$, is in the complexity class NC^1 but outside AC^0 (see Fig. 4). Our algorithmic perspective is suggestive of an open problem: *small Collatz cycles*: Does there exist $x > 2 \in \mathbb{N}$ such that $x = T^{\leq \lceil \log_2 x \rceil}(x)$?

The proofs of our main result and supporting lemmas use the fact that the *local* (CA-like) component of the CQCA can be thought of as simultaneously iterating two finite state transducers (FSTs). One FST computes $x/2$ in ternary on vertical columns, the other FST computes $3x+1$ in binary on horizontal rows, as shown in Fig. 3. Interestingly, the two FSTs are dual in the sense that states of one are symbols of the other, and arrows of one are read/write instructions of the other. In addition, intuitively, the *non-local* component of the CQCA initiates, or "bootstraps", these two FSTs by providing the location of the least significant bit to each.

Related and Future Work. Since the operations $3x + 1$ and $x/2$ have natural base 2 interpretations, studying the process in binary has been fruitful [4,7,11,24]. For example, in binary, predecessors in the Collatz process are characterised by regular expressions [25]: for each $x, k \in \mathbb{N}$ there is a regular expression, of size exponential in k, that characterises the binary representations of all $y \in \mathbb{N}$ that lead to x via k applications of T_1 and any number of T_0.

Cloney, Goles and Vichniac give a unidimensional "quasi" CA that simulates the Collatz process [6]. Their system works in base 2, for any $x \in \mathbb{N}$ given as an input in base 2, successive downward rows encode the iterates $x, T(x), T^2(x), \ldots$, in binary. The choice of whether to apply $x/2$ and $3x + 1$ is done explicitly based on the least significant bit, hence the rule is not local. In a similar spirit,

[1] The CQCA could easily be altered to remove the non-local rule and have it be a CA, but the obvious ways to do so would involve using more states and make the mapping to the Collatz process less direct. The CQCA has the freezing property [9,27]; states change obey a partial order, with a constant number of changes permitted per cell.

Bruschi [3] defines two distinct non-local 1D CA-like rules, one which works in base 2 and the other in base 3. Finally, in base 6, the Collatz process can be expressed as a local CA because carry propagation does not occur [13,15]. However, because these systems do not include carries in the automaton state's space, our base conversion result, and parity-based observations on the structure of Collatz iterates, are not apparent, nor is the CQCA-style embedding of dual FSTs that compute $3x + 1$ and $x/2$.

Base conversion is a problem which has been studied in several ways: it is known to be computable by iterating Finite State Transducers [2], it is known to be in the circuit complexity class NC^1 [10] and the complexity of computing base conversion of real numbers (infinite expansion) has been explored [1,2,12]. While we know that our framework generalises well to the Collatz process running on infinite binary strings (i.e. the extension [17,19,23] of T to $\mathbb{Z}_2$, see Remark 9), we leave to future work to show how our base conversion result applies in that case: for instance, can the CQCA convert any element $x \in \mathbb{Z}_3 \cap \mathbb{Z}_2 \cap \mathbb{Q}$ from $\mathbb{Z}_3$ (i.e. infinite ternary expansion) to $\mathbb{Z}_2$ (i.e. infinite binary expansion)?

Generalised Collatz maps, and related iterated dynamical systems, of both one and two variables, simulate Turing machines [8,14,16,20–22]. With two variables these maps simulate Turing machines in real time, just one map application per Turing machine step, and so have a P-complete prediction problem. The situation is less clear with 1-variable generalised Collatz maps (1D-GCMs), of which Collatz is an instance. Conway [8] showed that 1D-GCMs simulate Turing machines, but via an exponential slowdown, and it remains as a frustrating open problem whether the simulation can be made to run in polynomial time.[2] As with the 2-variable case, Turing Machines simulate 1D-GCMs in polynomial-in-n time (giving an upper bound), but a matching lower bound for predicting n steps of a 1D-GCM on an n-digit input remains elusive. Is the problem in NC, or even in NL? Is it P-complete? This point of view provides the motivation for a quest to understand the kinds of algorithmic behaviour that might be embedded in 1D-GCMs, and in their prime example, the Collatz map. Here, we show an AC^0 lower bound on that prediction problem in the CQCA model, and an NC^1 upper bound on roughly half the bits produced over $O(n)$ iterations. Finding a separation between the Collatz process itself, and 1D-GCMs, is an obvious next step. The structure we find in Collatz iterations leads us to guess that the CQCA n-step prediction problem might be simpler than the analogous problem for 1D-GCMs. We leave that challenging problem for future work.

We contend that our approach gives a fresh perspective that could lead to progress in understanding the Collatz process. We illustrate with two examples. Firstly, the CQCA runs in a maximally parallel fashion along a 135° diagonal (see Fig. 2(b)), thus our main base conversion result (Theorem 11) implies that such a diagonal encodes a natural number being converted from base 3 to 2. Hence, it implies that the Collatz process can be interpreted as running along successive

[2] The problem is closely related to the 2-counter machine problem: when simulating Turing machines, are 2-counter machines exponentially slower than 3-counter machines? See, for example, [14,18,22].

CQCA *diagonals* (rather than rows/columns), giving a new perspective on its dynamics, that we leave to future work. Secondly, by Theorem 23, and illustrated in Fig. 4, if we pick any cell along a column, the entire upper rectangle (above and to the left of the cell) is tightly characterised by our results: it is a base 3-to-2 conversion diagram (computable in NC^1, but not in AC^0). This leaves as future work the lower rectangle (below and to the left) and, we believe, embeds the full complexity of computing n forward Collatz steps and answering the *small Collatz cycles* conjecture. Section 4, Fig. 4 and Fig. 5 contain some additional discussion.

A simulator, `simcqca`, was built in order to run the CQCA[3]. The reader is invited to experience the results of this paper through the simulator.

2 The Collatz 2D Quasi-Cellular Automaton

The Collatz Process. Let $\mathbb{N} = \{0, 1, \dots\}$, $\mathbb{N}^+ = \{1, 2, \dots\}$ and let $\mathbb{Z}$ be the integers. The Collatz process consists of iterating the Collatz map $T : \mathbb{N} \to \mathbb{N}$ where $T(x) = T_0(x) = x/2$ if x is even and $T(x) = T_1(x) = (3x+1)/2$ if x is odd. The Collatz conjecture states that for any $x \in \mathbb{N}$ the process will reach 1 after a finite number of iterations. The cyclic Collatz conjecture states that the only strictly positive cycle is $(1, 2, 1, \dots)$.

CQCA Definition. The CQCA is pictorially defined in Fig. 1 and more formally defined in Sect. 2.3 of the full version of this paper [26]. A *configuration* is defined on $\mathbb{Z}^2$, where each cell in $\mathbb{Z}^2$ has a state $(s, c) \in S = \{0, 1, \perp\}^2 \setminus \{(\perp, 0), (\perp, 1)\}$ containing *sum bit* s and *carry bit* c, said to be *defined* if 0/1 or *undefined* otherwise ($\perp$). We say that the cell is *defined* if both the sum and carry defined, *half-defined* if sum is defined and carry undefined, and *undefined* if sum and carry are undefined. Note, in all Figures, cell colour distinguishes between a carry bit being undefined or being 0, and empty light/dark grey cells have sum bit 0, as defined in Fig. 1(a).

A configuration update step is parallel and synchronous: First, the non-local rule is applied (at most 1 row per step is updated by the non-local rule) then, on the updated configuration, the local rule is applied everywhere it can be. The non-local rule implements the $+1$ part of the $3x + 1$ operation as follows: on any horizontal row which has only half-defined cells, with exactly one cell ρ having sum bit 1, then the neighbour immediately to the right of ρ gets, *ex nihilo*, a carry bit equal to 1 (shown in red in Fig. 1(a), see [26] for a formal definition). The local rule works as shown in Fig. 1(a): for any undefined cell $e \in \mathbb{Z}^2$, with a half-defined north neighbour with sum s_1, and defined north-east neighbour with sum s_0 and carry c_0, e's sum bit becomes $s_2 := s_0 + c_0 + s_1 \mod 2$. Simultaneously, the carry, $c_1 := (s_0 + c_0 + s_1 \geq 2)$, is placed on the cell to the north of e. Figure 2(b) shows that the "natural" evolution frontier of the CQCA is along a 135° diagonal. Let the unit cardinal vectors in $\mathbb{Z}^2$ be $\mathtt{EAST} = (1, 0)$, $\mathtt{WEST} = (-1, 0)$, $\mathtt{NORTH} = (0, 1)$, $\mathtt{SOUTH} = (0, -1)$.

[3] Comprehensive instructions and tutorial at: https://github.com/tcosmo/simcqca.

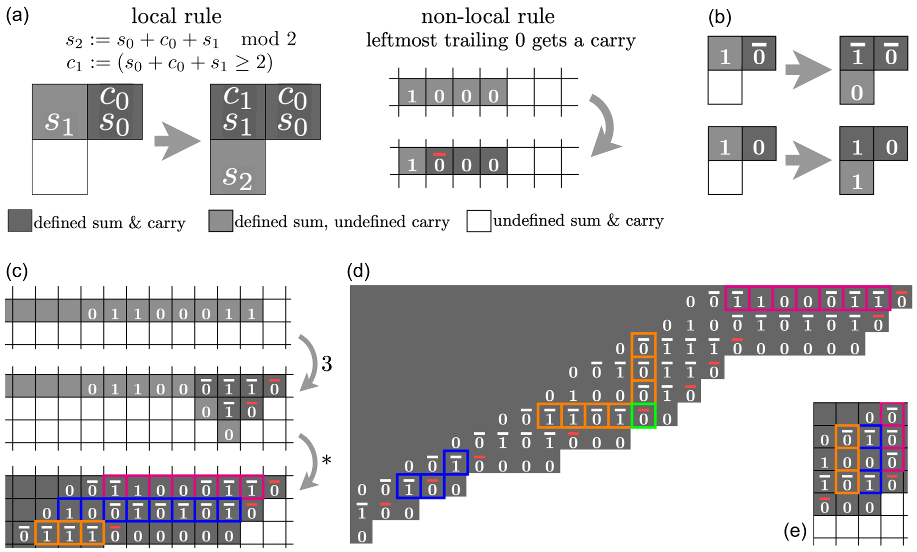
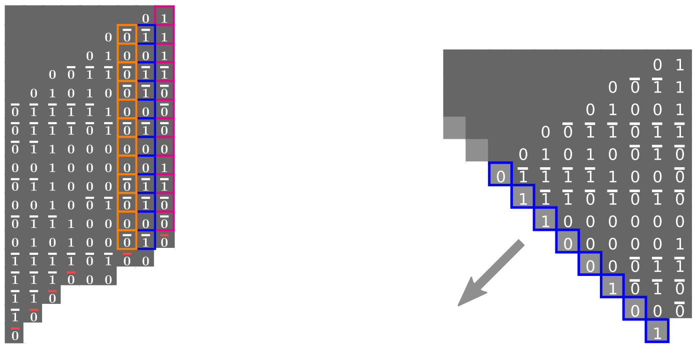

(a)
local rule
$s_2 := s_0 + c_0 + s_1 \mod 2$
$c_1 := (s_0 + c_0 + s_1 \geq 2)$
non-local rule
leftmost trailing 0 gets a carry
(b)
defined sum & carry
defined sum, undefined carry
undefined sum & carry
(c)
3
*
(d)
(e)

Fig. 1. CQCA definition and examples. (a) CQCA local rule, and non-local rule, for sum bit (s_2: written as 0 or 1), and carry bit (c_1: over-bar for $c_1 = 1$, omitted for $c_1 = 0$), where the bits c_0, s_0 and s_1 are already defined. Cell colour distinguishes between carry bit being undefined or 0, and empty light/dark grey cells have sum bit 0. (b) Two examples of the local rule. (c) Example run of the CQCA on base 2 (horizontal) input $w = 1100011$, showing the initial configuration $c_0[w]$ (Definition 2), then 3 steps, then > 20 steps. The CQCA evolves in the south-west direction, with successive rows corresponding to odd Collatz iterates in binary (Lemma 10), in this case $(\mathbf{99}, \mathbf{149}, 224, 112, 56, 28, 14, \mathbf{7}, \dots)$, i.e., $[\![1100011]\!]_2 = 99$ (magenta), $[\![10010101]\!]_2 = 149$ (blue) and $[\![111]\!]_2 = 7$. (d) Part of the limit configuration $c_\infty[w]$ (Lemma 4) associated to $w = 1100011$. The trivial cycle (1,2), in blue, has been reached. In orange, an instance of the base conversion theorem (Theorem 11): north of the green cell (*anchor cell*) reads in base 3′ (Definition 1), $[\![\bar{0}\bar{0}\bar{0}]\!]_{3'} = [\![111]\!]_3 = 13$ and is equal to the base 2 number represented by the sum bits to the west of the green cell: $[\![1101]\!]_2 = 13$. (e) Example run of the CQCA on vertical base 3 input $\alpha = 111$ encoded in base 3′ as $\bar{0}\bar{0}\bar{0}$ (Definition 1). Each column is a successive T-iterate, from magenta to orange, we read: $[\![\bar{0}\bar{0}\bar{0}]\!]_{3'} = 13$, $[\![\bar{1}0\bar{1}]\!]_{3'} = 20 = T(13)$, $[\![\bar{0}00]\!]_{3'} = 10 = T(20)$ (Lemma 10), see Fig. 2(a) for a larger vertical example. (Color figure online)

Base 2, 3 and 3′. Let $\{0,1\}^*$ be the set of finite binary strings, $\{0,1,2\}^*$ be the set of finite ternary strings. We index strings from their rightmost symbol, and $|\cdot|$ denotes string length, meaning we write, for instance, $w = w_{|w|-1} \dots w_1 w_0 \in \{0,1\}^*$. We use the standard interpretation of strings from $\{0,1\}^*$ and $\{0,1,2\}^*$ as, respectively, base 2 and base 3 encodings of natural numbers where the rightmost symbol is the least significant digit. We write $[\![\cdot]\!]_2 : \{0,1\}^* \to \mathbb{N}$ and $[\![\cdot]\!]_3 : \{0,1,2\}^* \to \mathbb{N}$ for those interpretations. For instance, $[\![110]\!]_2 = [\![20]\!]_3 = 6$.

The CQCA uses a particular encoding for base 3 strings, called base 3′, over the four-symbol alphabet $\{0, \bar{0}, 1, \bar{1}\}$. The CQCA states (0,0), (0,1), (1,0),



(a) Columns in the CQCA iterate the Collatz process in base $3'$.

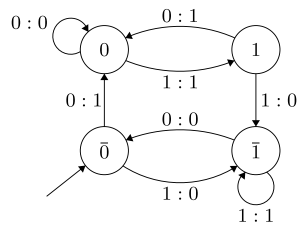
(b) The evolution frontier of the CQCA is a $135°$ diagonal.

Fig. 2. Evolution of a column input in the CQCA. Colours as in Fig. 1. (a) Portion of $c_\infty[\alpha]$, the limit configuration of the CQCA on column input $\alpha = 111211101211$. Successive columns iterate the Collatz process in base $3'$ (Lemma 10). We read: $[\![\bar{1}0\bar{1}\bar{1}0\bar{1}00\bar{1}\bar{1}0\bar{1}]\!]_{3'} = 408314 = T(272209)$ (blue) and $[\![\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}]\!]_{3'} = 204157 = T(408314)$ (orange). (b) Evolution of $c_0[\alpha]$ after $|\alpha|$ CQCA steps, highlighting the "natural" frontier of evolution of the CQCA as a $135°$ diagonal (blue cells). Cells to the north-east of the diagonal are defined, cells to the south-west are undefined and cells on the diagonal are half-defined. (Color figure online)

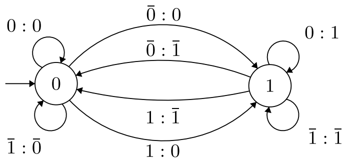
$(1,1) \in S$ respectively represent $0, \bar{0}, 1, \bar{1}$, using a (sum-bit, carry-bit) notation. The function $[\![\cdot]\!]_{3' \to 3} : \{0, \bar{0}, 1, \bar{1}\}^* \to \{0,1,2\}^*$ converts base $3'$ to base 3 in a straightforward symbol-by-symbol fashion: $0 \mapsto 0$, $\bar{0} \mapsto 1$, $1 \mapsto 1$ and $\bar{1} \mapsto 2$. For instance, $[\![\bar{0}\bar{0}01\bar{1}]\!]_{3' \to 3} = 11012$. We write $[\![\cdot]\!]_{3'} = [\![[\![\cdot]\!]_{3' \to 3}]\!]_3$ as the interpretation of base $3'$ strings as natural numbers. However, because there are two choices for encoding the trit 1 in base $3'$, converting from base 3 to base $3'$ requires a definition:

Definition 1 (Base 3 to $3'$ encoding). The function $[\![\cdot]\!]_{3 \to 3'} : \{0,1,2\}^* \to \{0, \bar{0}, 1, \bar{1}\}^*$ encodes a base 3 word α as a base $3'$ word as follows: 0 is encoded as 0, 2 is encoded as $\bar{1}$, and 1 is encoded as 1 when the rightmost neighbour different from 1 is a 2, and as $\bar{0}$ otherwise. E.g., $[\![111211101211]\!]_{3 \to 3'} = 111\bar{1}\bar{0}\bar{0}\bar{0}01\bar{1}\bar{0}\bar{0}$.

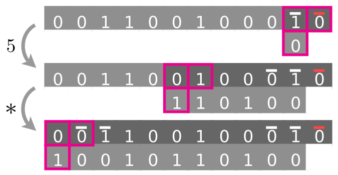
Transducers and Duality. The CQCA rule has a local and non-local component. The *local* component simulates two FSTs: the $3x+1$ FST in binary along horizontal rows and the $x/2$ FST in ternary along vertical columns (Fig. 3). Intuitively, the *non-local* component of the CQCA provides the least significant bit to these FSTs, in other words, it runs $x/2$ in binary (removing a trailing 0) and $3x+1$ in ternary (adding a trailing 1). Interestingly, these two FSTs are closely related, we say they are *dual*: states of one are symbols of the other and that arrows of one are read/write instructions of the other. The proof of our main result Theorem 11, and of Lemmas 7 and 10, use the ability of the CQCA to simulate these FSTs simultaneously, one horizontally and the other vertically.



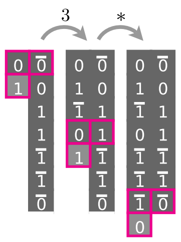
(a) The $3x+1$ binary FST.



(b) The $x/2$ base $3'$ FST.



(c) Horizontal applications of the local CQCA rule simulate the $3x+1$ FST.



(d) Vertical applications of the local rule simulate the $x/2$ FST.

Fig. 3. Panels (a) and (b) present the $3x+1$ binary Finite State Transducer (FST) and the $x/2$ base $3'$ FST. Instructions "$s_1 : s_2$" mean "read s_1, write s_2". The FSTs are *dual* to one another: states of one are symbols of the other and arrows of one are read/write instructions of the other. Panels (c) and (d) show the relation with the local CQCA rule: (c) horizontal CQCA applications correspond to simulating the $3x+1$ FST and (d) vertical CQCA applications to simulating the $x/2$ FST. The same colour code as Fig. 1 is used. More precisely, in (c), on input $[\![00110010001]\!]_2 = 401$ and initial state $\bar{0}$ (rightmost 0 with red carry), we get output $[\![10010110100]\!]_2 = 1204 = 3 \times 401 + 1$. In (d) on input $[\![\bar{0}011\bar{1}\bar{1}\bar{0}]\!]_{3'} = 862$ and initial state 0 (top-left most sum bit 0), we get output $[\![01\bar{1}0\bar{1}\bar{1}\bar{1}]\!]_{3'} = 431 = 862/2$. For clarity of exposition, we are "illegally" running the CQCA in a vertical (c), or horizontal (d), sequential mode—in the legal, or "natural", parallel mode, see Fig. 2(b), more cells would have been computed than are shown. The non-local component of the CQCA rule provides the FSTs with the (red) carry at the least significant digit. (Color figure online)

3 Base Conversion in the Collatz Process

Definition 2 (Binary initial configuration $c_0[w]$). For any binary input $w \in \{0,1\}^*$, we define $c_0[w] \in S^{\mathbb{Z}^2}$ to be the initial configuration of the CQCA with w written on the horizontal ray $y = 0, x < 0$ as follows: for $1 \le i \le |w|$ we set $c_0[w](-i,0) = (w_{i-1}, \perp)$, for $i > |w|$ we set $c_0[w](-i,0) = (0,\perp)$ and for all other positions $(x,y) \in \mathbb{Z}^2$ we set $c_0[w](x,y) = (\perp,\perp)$.

Definition 3 (Ternary initial configuration $c_0[\alpha]$). For any ternary input $\alpha \in \{0,1,2\}^*$ we define $c_0[\alpha] \in S^{\mathbb{Z}^2}$ to be the initial configuration of the CQCA with $\alpha' = [\![\alpha]\!]_{3 \to 3'}$ (Definition 1) written on the vertical ray $x = 0, y > 0$ as follows: for $1 \le i \le |\alpha|$ we set $c_0[\alpha](0,i) = \texttt{state}(\alpha'_{i-1})$ where $\texttt{state} : \{0,\bar{0},1,\bar{1}\} \to S$ is such that $\texttt{state}(0) = (0,0)$, $\texttt{state}(\bar{0}) = (0,1)$, $\texttt{state}(1) = (1,0)$ and $\texttt{state}(\bar{1}) = (1,1)$. Also, for all $x < 0$ we set $c_0[\alpha](x,|\alpha|) = (0,\perp)$ and finally at all other positions we set $c_0[\alpha](x,y) = (\perp,\perp)$.

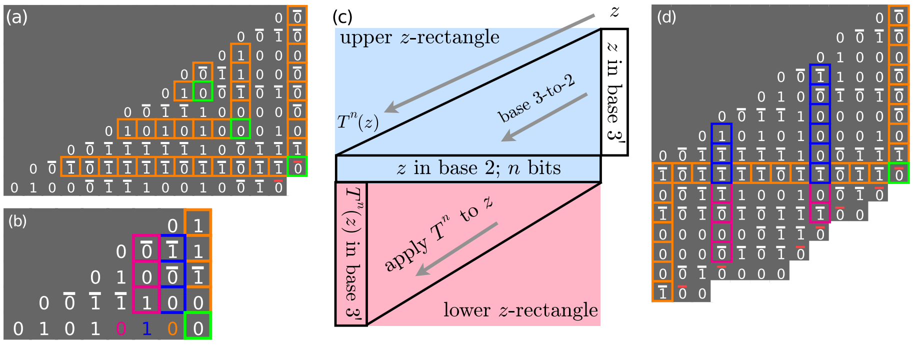


Fig. 4. Structure of the CQCA, and Collatz iterations, implied by our results. (a) Three instances of Theorem 11. From the innermost to the outermost they read $[\![\bar{0}]\!]_{3'} = [\![1]\!]_3 = [\![1]\!]_2 = 1$, then $[\![11\bar{1}0]\!]_{3'} = [\![1120]\!]_3 = [\![101010]\!]_2 = 42$, and finally $[\![\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}0\bar{1}]\!]_{3'} = [\![101111011011]\!]_2 = 3035$. Anchor cells are in green. (b) Zoom-in of base conversion diagram of $[\![11\bar{1}0]\!]_{3'}$ (middle example in (a)). Each column, excluding its bottom cell, represents successive integer divisions by 2, e.g., from orange to magenta columns we read $[\![1120]\!]_3 = 42$, $[\![210]\!]_3 = 21$ and $[\![101]\!]_3 = 10$. At each step, the sum bits of the bottom row "store" the parity information of the previous column that was divided by 2: the orange bit gives the parity of the orange column and so on. Checking parity in base 3′ is checking the parity of the number of 1s (both sum and carry bits), hence outside AC^0 (Corollary 15, Theorem 19). (c) For any input $z \in \mathbb{N}$ the schematic shows $n = \lceil \log_2 x \rceil$ iterations of the Collatz map T. The values $x, T(x), T^2(x), \ldots, T^n(x)$ appear as n columns, written in ternary (base 3′). The configuration is cut by a horizontal line, whose cells encode z in binary (Theorem 11). The 'base conversion' upper z-rectangle is simple to define in terms of z, and is computable in NC^1. The lower z-rectangle embeds the full complexity of computing n iterations of T, but with input being in base 2 and output in base 3′. (d) The influence of $z = [\![\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}0\bar{1}]\!]_{3'} = [\![11011102]\!]_3 = 3035$ on its next $n = \lceil \log_2 z \rceil$ Collatz iterates. The column in orange on the right is the base 3′ encoding of z. By Lemma 10, the column in orange on the left is the base 3′ encoding of $T^n(z) = T^{12}(z) = [\![2020002]\!]_3 = 1622$. By Theorem 11, the orange horizontal sum bits give the base 2 representation of z. Hence the cells outlined in blue (upper z-rectangle) in $T^4(z)$ (second outlined column to the right) and $T^9(z)$ (third outlined column), are entirely determined by the base 3′ to base 2 conversion diagram of z, only the cells outlined in magenta (lower z-rectangle) are independent from the base conversion algorithm. (Color figure online)

Each initial configuration has a well-defined unique limit configuration:

Lemma 4 (Limit configurations $c_\infty[w]$ and $c_\infty[\alpha]$). Let $w \in \{0,1\}^*$ be a finite binary string and $\alpha \in \{0,1,2\}^*$ be a finite ternary string. Then, in the CQCA evolution from the initial configurations $c_0[w] \in S^{\mathbb{Z}^2}$, and $c_0[\alpha] \in S^{\mathbb{Z}^2}$, both sum and carry bits in any cells are final: if set, they never change. Hence, limit configurations $c_\infty[w] = \lim_{i\to\infty} F^i(c_0[w])$ and $c_\infty[\alpha] = \lim_{i\to\infty} F^i(c_0[\alpha])$ exist.[4]

Proof. Both the local and non-local rules of the CQCA can only be applied either on undefined cells or half-defined cells. Furthermore, if applied on an undefined cell, the cell becomes half-defined or defined, and when applied on a half-defined cell it becomes defined. Hence, the following partial order $(\perp,\perp) < (s,\perp) < (s,c)$ with $s, c \in \{0,1\}$ holds on the states of the CQCA: it has the freezing property [9,27] and cells can change state at most twice. Limit configurations are well-defined by taking the final state of each cell. $\square$

Example 5. Figure 1(c) (top) and (d) respectively show $c_0[w]$ and a portion of $c_\infty[w]$ for $w = 1100011$. Figure 2(a) shows a portion of $c_\infty[\alpha]$ for $\alpha = 111211101211$; by Definition 1 we have $[\![111211101211]\!]_{3\to 3'} = 111\bar{1}\bar{0}\bar{0}\bar{0}01\bar{1}\bar{0}\bar{0}$.

Next, we define how to read base 2 strings on rows and base $3'$ on columns of the CQCA:

Definition 6 (Mapping rows and columns to natural numbers). Let $x_0, y_0 \leq 0$ and $c \in S^{\mathbb{Z}^2}$ be a configuration. A finite segment of defined cells along a horizontal row x_0 of c is said to *give* word $w \in \{0,1\}^*$ if w is exactly the concatenation of the sum bits in these cells (LSB on right). An infinite horizontal row y_0 of c is said to *give* $w \in \{0,1\}^*$ if there is a $k \geq 0$ such that the defined sum bits on y_0 are $0^\infty w 0^k$.

A contiguous segment of defined cells on the column of c with x-coordinate x_0 is said to *give* the base $3'$ word $q \in \{0,\bar{0},1,\bar{1}\}^*$ if q is exactly the concatenation of the base $3'$ symbols corresponding to each cell's state[5] (where the southmost cell gives the least significant trit).

Intuitively, the following lemma says that rows of CQCA encode odd Collatz iterates in binary. There is one odd iterate per row, Fig. 1(c).

Lemma 7 (Rows simulate Collatz in base 2). Let $z \in \mathbb{N}^+$ and let $w \in \{0,1\}^* \setminus \{0\}^*$ be the standard base 2 representation of z and, by Lemma 4, let $c_\infty[w] \in S^{\mathbb{Z}^2}$ be the CQCA limit configuration on input w. Then, the horizontal row $y_0 \leq 0$ of $c_\infty[w]$ gives the base 2 representation of the $|y_0|^{\text{th}}$ odd term in the Collatz sequence of z (Definition 6).

Proof. Note that it is enough to show that row $y = -1$ of $c_\infty[w]$ gives the base 2 representation of the second odd term in the Collatz sequence of z and then inductively apply the argument to all rows $y < -1$ to get the result. We have $w \neq 0^n$, hence there is at least one 1 in w and so, on row $y = 0$ of

[4] Furthermore, although the following fact is not used in the rest of this paper, one can prove that limit configurations contain no half-defined cells: each cell is either defined or undefined.

[5] The CQCA defined states are $(0,0), (0,1), (1,0), (1,1)$ and they respectively map to base $3'$ symbols $0, \bar{0}, 1, \bar{1}$.

$c_\infty[w]$, the non-local rule of the CQCA (Fig. 1(a)) was applied exactly once, at position $x_0 = \max(\{x < 0 \mid c_\infty[w](x,0) = (s,c) \text{ with } s = 1\}) + 1$ and we have $c_\infty[w](x_0,0) = (0,1)$. Sum bits on row $y = 0$ up to column $x_0 - 1$ give the representation of $z' \in \mathbb{N}$ (Definition 6), the first odd term in the Collatz sequence of $z = [\![w]\!]_2$.

From position $x = x_0 - 1$ down to $x_0 = -\infty$, the local rule of the CQCA was applied to produce carries on row $y = 0$ and sum bits on row $y = -1$. Given the definition of the CQCA local rule, that computation corresponds exactly to applying the binary $3x+1$ FST (Fig. 3(a) and (c)): input is read in the sum bits of row $y = 0$ from $x = x_0 - 1$ down to $x = -\infty$, output is produced in the sum bits of row $y = -1$, the initial state is $\bar{0}$, corresponding to $c_\infty[w](x_0,0) = (0,1)$. Hence, by Lemma 5 in the full version of this paper [26], which asserts the correctness of the $3x+1$ FST computation, sum bits of row $y = -1$ from $x = -\infty$ to $x = x_0 - 1$ give the binary representation of $3z' + 1$; starting with a $(0)^\infty$ prefix and with LSB to the east. By ignoring the $m \geq 1$ trailing zeros on row $y = -1$ (there is at least one because $3z' + 1$ is even), we get the representation of $(3z' + 1)/2^m$, the second odd iterate in the Collatz sequence of z. Note the non $(0)^\infty$ part of row $y = -1$ is produced in $\leq |w| + 2$ steps in the CQCA. □

Remark 8. Although Lemma 7 only deals with odd Collatz iterations, even Collatz iterations also naturally appear in the CQCA: even terms occurring between the i^{th} and the $(i+1)^{\text{th}}$ odd Collatz iterates correspond to the trailing 0s on row $y = -(i+1)$ of the CQCA. For example, on the third row in Fig. 1(c) (bottom) we read $7 = [\![111]\!]_2$ but also, in the trailing $0s$, all $2^n \cdot 7$ for $1 \leq n \leq 6$.

Remark 9. The Collatz process can be generalised to an uncountable class of numbers that includes both $\mathbb{N}$ and $\mathbb{Z}$: $\mathbb{Z}_2$, the ring of 2-adic integers which syntactically corresponds to the set of infinite binary words [5,17]. Given that generalisation, the Collatz process can be run on more exotic numbers such as $-\frac{4}{23} \in \mathbb{Z}_2 \cap \mathbb{Q}$ and the Collatz conjecture generalises as follows: all rational 2-adic integers eventually reach a cycle[6] (known as Lagarias' Periodicity Conjecture [17]). When running the $3x+1$ FST on infinite binary inputs, one can show that it is computing the $3x+1$ function on 2-adic integers (see [26], Appendix B). Hence, Lemma 7 and the CQCA framework in general, can be generalised for working with infinite binary inputs and the Collatz process in $\mathbb{Z}_2$.

Intuitively, the following lemma says that columns of the CQCA encode all Collatz iterates, even and odd, in ternary, as in Figs. 1(e) and 2(a).

Lemma 10 (Columns simulate Collatz in base 3′). Let $z \in \mathbb{N}^+$ and let $\alpha \in \{0,1,2\}^* \backslash \{0\}^*$ be the standard base 3 representation of z, and, by Lemma 4, let $c_\infty[\alpha] \in S^{\mathbb{Z}^2}$ be the CQCA limit configuration on input α. Then, the vertical column $x_0 < 0$ in $c_\infty[\alpha]$ gives the base $3'$ representation of $T^{|x_0|}(z)$ (as the defined cells down to, and excluding, the southmost cell with sum bit 0, Definition 6).

[6] The set $\mathbb{Z}_2 \cap \mathbb{Q}$ exactly corresponds to irreducible fractions with odd denominator and parity is given by the parity of the numerator. For instance, the first Collatz steps of $-\frac{4}{23}$ are: $(-\frac{4}{23}, -\frac{2}{23}, -\frac{1}{23}, \frac{10}{23}, \frac{5}{23}, \dots)$. It reaches the cycle $(\frac{5}{23}, \frac{19}{23}, \frac{40}{23}, \frac{20}{23}, \frac{10}{23}, \frac{5}{23} \dots)$.

Proof. Note that it is enough to show that column $x_0 = -1$ of $c_\infty[\alpha]$ gives the base $3'$ representation of $T(z)$, with $z = [\![\alpha]\!]_3$, and then inductively apply the argument to all columns $x < -1$ to get the result. By construction of $c_0[\alpha]$ (Definition 3) and because all bits are final (Lemma 4) we have that the sum bit of $c_\infty[\alpha](-1, |\alpha|)$ is 0 (more generally, for all $x < 0$ we have the sum bit of $c_\infty[\alpha](x, |\alpha|)$ is 0). From there, the local CQCA rule (Fig. 1) is applied to each position $(-1, y)$ with $1 \leq y \leq |\alpha|$. This application of the rule corresponds exactly to running the base $3'$ $x/2$ FST (Fig. 3(b) and (d)) by reading the input on the base $3'$ symbols of cells of column $x = 0$ (both sum and carry bits of these cells are defined in $c_0[\alpha]$), writing the base $3'$ output on the cells of column $x = -1$ starting from initial FST state 0 (corresponding to the sum bit of $c_\infty[\alpha](-1, |\alpha|)$ being 0). In that interpretation, the sum bit output to the south of a cell by the local CQCA rule corresponds to the new FST state after reading the east base $3'$ symbol, hence the sum bit s of cell $(-1, 0)$ corresponds to the state of the $x/2$ FST after reading all base $3'$ symbols of $[\![\alpha]\!]_{3\to 3'}$ (the least significant digit is at position $(0, 1)$). Two cases, with $z = [\![\alpha]\!]_3$:

1. If $z \equiv 0 \mod 2$, by Lemma 6 in the full version of this paper [26], we have that the final state of the $x/2$ FST after reading $[\![\alpha]\!]_{3\to 3'}$ is 0 and that the output word $\alpha' \in \{0, \bar{0}, 1, \bar{1}\}$ is such that $[\![\alpha']\!]_{3'} = [\![\alpha]\!]_3/2 = z/2$. Hence, we deduce that column $x = -1$, from $y = |\alpha|$ down to $y = 1$, gives the base $3'$ representation of $T(z) = z/2$ which is what we wanted.
2. If $z \equiv 1 \mod 2$, by Lemma 6 of [26], we have that the final state of the $x/2$ FST after reading $[\![\alpha]\!]_{3\to 3'}$ is 1 and that the output word $\gamma \in \{0, \bar{0}, 1, \bar{1}\}$, which is written on column $x = 1$, $y = |\alpha|$ down to $y = 1$, is such that $[\![\gamma]\!]_{3'} = ([\![\alpha]\!]_3 - 1)/2 = (z - 1)/2$. Furthermore, for $e = (0, 0) \in \mathbb{Z}^2$, the sum bit of $c_\infty[\alpha](e + \texttt{WEST})$, which corresponds to the final state of the $x/2$ FST, is $s = 1$ and $c_0[\alpha](e) = (\perp, \perp)$. Hence, the non-local rule will be applied at position e giving $c_0[\infty](e) = (0, 1) = (s_e, c_e)$. Then, at the following CQCA step, since $s_e + c_e + s = 0 + 1 + 1 \geq 2$ we get a carry bit of 1 at (e+$\texttt{WEST}$), i.e. $c_0[\infty](e + \texttt{WEST}) = (1, 1)$. It means that on column $x = -1$, with cell at position $(-1, 0) = e + \texttt{WEST}$ we add the base $3'$ symbol $\bar{1}$ on the least significant side of γ (word γ was output by the FST to the north of position $(-1, 0)$). Hence, column $x = -1$, from row $y = |\alpha|$ down to $y = 0$, interprets as: $3 \cdot [\![\gamma]\!]_{3'} + 2 = 3\frac{z-1}{2} + 2 = \frac{3z+1}{2} = T(z)$. Which is what we wanted.

Hence we get that column $x = -1$ gives the base $3'$ representation of $T(z)$. From the above points, it is immediate that the cell directly below the base $3'$ expression of $T(z)$ on column $x = -1$ has sum bit equal to 0 and that all cells below are undefined, giving the end of the Lemma statement. Note that it requires at most $|\alpha| + 2$ simulation steps for the CQCA to compute that representation (at most two extra cells to the south are used). $\square$

We now prove our main result: the natural number written in base $3'$ on a column is converted to base 2 on the row directly south-west to it, see Fig. 4.

Theorem 11 (Base 3-to-2 conversion). Let $\alpha \in \{0, 1, 2\}^*$ be a finite ternary string. By Lemma 4 let $c_\infty[\alpha] \in S^{\mathbb{Z}^2}$ be the CQCA limit configuration on input α.

Then, in $c_\infty[\alpha]$, any position $e = (x_0, y_0) \in \mathbb{Z}^2$ such that both cells $e + \texttt{NORTH}$ and $e + \texttt{WEST}$ are defined, is called an *anchor cell* and has the *base conversion property*: there exists $z \in \mathbb{N}$ such that the defined cells on column x_0 strictly to the north of e give (Definition 6) the base $3'$ representation of z and the cells on row $y = y_0$ strictly to the west of e give the base 2 representation of z.

Proof. A direct consequence of the proof of Lemma 10 is that, in $c_\infty[\alpha]$ a defined sum bit $s \in \{0, 1\}$ at position (x_1, y_0) with $x_1 < 0$ and $y_0 < |\alpha|$ gives the state of the $x/2$ FST (Fig. 3(b) and (d)) immediately after it reads all base $3'$ symbols from row $y = |\alpha|$ to $y = y_0 + 1$ on column with x-coordinate $x_1 + 1$ of $c_\infty[\alpha]$. As s is the final FST state, by Lemma 6 [26], we get that s is 0 if that base $3'$ represented number was even; otherwise (if odd) s is 1.

We also know that the output of the FST, i.e. symbols strictly to the north of (x_1, y_0) represent $\lfloor x/2 \rfloor$. Hence, one base conversion step was performed: x mod 2 is written in the sum bit s at position (x_1, y_0) and $\lfloor x/2 \rfloor$ is computed to the north of it. By induction, all the other base conversion steps are also performed to the west of (x_1, y_0) and we get the result for the anchor cell (x_0, y_0) with $x_0 = x_1 + 1$. □

Remark 12. Figure 4(a) presents several instances of Theorem 11. Note that position $(0, 0)$ is always an anchor cell in $c_0[\alpha]$ meaning that the CQCA converts $[\![\alpha]\!]_{3 \to 3'}$ from base $3'$ to base 2. Hence, the CQCA can effectively convert any base $3'$ input to base 2. Figure 4(b) presents the details of such a base conversion and shows that the CQCA base conversion algorithm is rather natural: at each step the parity of the input is computed, then the input is divided by 2. Also, the algorithm is efficient: for $x \in \mathbb{N}$, it can be shown that $\lceil \log_2(x) \rceil + \lceil \log_3(x) \rceil$ CQCA steps are sufficient to convert x from base 3 to base 2.

Remark 13. Theorem 11 also implies that limit configurations of row inputs are very similar to limit configurations of column inputs. Indeed, for $z \in \mathbb{N}$, with base 2 representation $w \in \{0, 1\}^*$ and base 3 representation $\alpha \in \{0, 1, 2\}^*$, we have: for all $x \leq 0$ and $y \leq 0$, $c_\infty[w](x, y) = c_\infty[\alpha](x, y)$. Thus, for all $x \leq -|w|$ and $y < 0$, columns of $c_\infty[\omega]$ iterate the Collatz process in ternary and the base conversion property holds for any anchor cell. Hence, the base conversion property naturally appears in the CQCA, for both row and column inputs.

Remark 14. Theorem 11 implies that cells with state $(0, 1)$ because of the non-local rule (red carries in Fig. 1 and 4) have an interesting interpretation column-wise: they implement the operation $3x + 1$ in ternary. Indeed, in base 3 the operation $3x + 1$ is trivial: it consists of appending 1 (represented here by $\bar{0}$ via Definition 1) to the base $3'$ representation of x. Thus, the CQCA "stacks" trivial ternary steps $(3x + 1)$ on the same column, and trivial binary steps ($x/2$, i.e. a shift in binary) on the same row (Remark 8).

Corollary 15 (Parity checking in Collatz). Let $\alpha \in \{0, 1, 2\}^*$ and, by Lemma 4, let $c_\infty[\alpha] \in S^{\mathbb{Z}^2}$ be the limit configuration of the CQCA on input α (written in base $3'$ on column $x = 0$). For any anchor cell (Theorem 11) at position $e \in \mathbb{Z}^2$, let $s \in \{0, 1\}$ be the sum bit of the cell at $e + \texttt{WEST}$. Then, s is the parity of the number written in base $3'$ on the column at $e + \texttt{NORTH}$ and going to the north (Definition 6): this number is even iff $s = 0$ and odd iff $s = 1$.

Proof. Immediately implied by the proof of Theorem 11: the sum bit s at position $e + \texttt{WEST}$ gives the state in which the $x/2$ FST was after reading base $3'$ symbols on the column to the north of e. By Lemma 6 in the full version of this paper [26], the bit s is the parity of the number given by that column. $\square$

Example 16. Figure 4(b) outlines instances of Corollary 15.

3.1 Computational Complexity of CQCA Prediction

In this section we leverage our previous results to make statements about the computational complexity of predicting the CQCA.

Definition 17 (Bounded CQCA prediction problem). Given (a) any ternary input α, of length $n \in \mathbb{N}$ with resulting CQCA limit configuration $c_\infty[\alpha]$, and (b) any $(x, y) \in \mathbb{Z}^2$, where $\max(|x|, |y|) = O(n)$, what is the state $c_\infty[\alpha](x, y)$?

Remark 18. The version of this prediction problem, where the question is to predict the state $c_\infty[\alpha](x, y)$ for any $(x, y) \in \mathbb{Z}^2$, is at least as hard as the Collatz conjecture which in CQCA terms states that the $(1, 2, 1, 2, \dots)$ "glider" will eventually occur, cf. blue outlined cells in Fig. 1(d).

It is straightforward to see that the bounded CQCA prediction problem is in the complexity class P, we can also give a lower bound in terms of AC^0 which is the class of problems solved by uniform[7] polynomial size, constant depth Boolean circuits with arbitrary gate fanin [22]:

Theorem 19. The bounded CQCA prediction problem is in P and not in AC^0.

Proof. To see that the problem is in P, simply encode the initial configuration as input to a two-tape Turing machine and, in $O(n^2)$ time, run the simulation forward until we have filled the plane up to distance n from the input.

To show the problem is outside AC^0, let $v \in \{0, 1\}^*$, and let $\alpha \in \{0, 1, 2\}^*$ be the base 3 word such that $\alpha = v$. Let $c_\infty[\alpha]$ be the limit configuration of the CQCA on (column) input α written in base $3'$ (as usual) on column $x = 0$. Since there are no 2-symbols in v, by Definition 1, the base $3'$ encoding of v maps 0 to 0 and 1 to $\bar{0}$, an encoding straightforward to represent in binary, and straightforward to compute in AC^0. Let b be the sum bit of $c_\infty[\alpha](-1, 0)$, i.e.

[7] Here, uniform has the meaning used in Boolean circuit complexity: that members of the circuit family for an infinite problem (set of words) are produced by a suitably simple algorithm [10]. The class AC^0 is of interest as it is "simple" enough to be strictly contained in P, that is, $\text{AC}^0 \subsetneq \text{P}$ [22]. This enables us to give lower bounds to the computational complexity, or inherent difficulty, of some problems, such as the bounded CQCA prediction problem.

$(-1, 0) + \texttt{NORTH} + \texttt{EAST}$ is the first symbol of α, hence (well) within the bound $n = O(|\alpha|)$ set by Definition 17.

Deciding whether a natural number is odd or even is equivalent to checking the parity of its number of 1s written in base 3. In base 3′, this translates to checking the parity of the total number of sum and carry bits. By Corollary 15, b is the parity of the natural number represented by α, equivalently, b is the parity of the number of 1s in v. Hence the CQCA solves the problem $\texttt{PARITY} = \{v \in \{0,1\}^* \mid v \text{ has an odd number of 1s}\}$ on the input v, with the result placed at distance $2 < |\alpha|$ from the input word α. Since PARITY sits outside the complexity class AC^0 [22], the Bounded CQCA prediction problem is not in AC^0. □

Remark 20. The proof of the previous theorem shows how to use the CQCA to solve PARITY. In fact, we can say something stronger: in any CQCA configuration, each sum bit with defined $\texttt{NORTH} + \texttt{EAST}$ neighbour solves an instance of the PARITY problem. See Fig. 4.

Definition 21 **(Upper z-rectangle prediction problem).** Let $\alpha \in \{0,1,2\}^*$, $z = [\![\alpha]\!]_3 \in \mathbb{N}$ and let $n = \lceil \log_2 z \rceil \in \mathbb{N}$. Let $c[\alpha]$ be the associated CQCA initial configuration (Definition 3) and $R = \{(x, y) \mid -n \le x \le 0, 0 \le y \le |\alpha|\} \subsetneq \mathbb{Z}^2$. The upper z-rectangle prediction problem asks: What are the states, in the limit configuration $c_\infty[\alpha]$, of all cells with positions $(x, y) \in R$.

Example 22. Figure 4(c) gives a schematic representation of R, the upper z-rectangle. Figure 4(d) and Fig. 5 each give and instance of R, respectively for $z = [\![101111011011]\!]_2 = 3035$ and $z = [\![11110010]\!]_2 = 242$.

NC^1 is the class of problems solved by uniform polynomial size, logarithmic depth (in input length) Boolean circuits with gate fanin ≤ 2 [10]. The proof of the following theorem intuitively comes from the fact that predicting the entire upper z-rectangle amounts to running $\lceil \log_3 z \rceil$ base conversions in parallel (the proof is in the full version of this paper [26], Theorem 28).

Theorem 23. The upper z-rectangle prediction problem is in NC^1, and is not in AC^0.

Open Problem 1 (Lower z-rectangle prediction problem). What is the complexity of filling out the lower z-rectangle? I.e. the rectangle defined by $M = \{(x, y) \mid -n \le x \le 0, -n \le y \le 0\} \subsetneq \mathbb{Z}^2$ (same notation as Definition 21 and shown on Fig. 4(c)). We know it is in P, and that it is not in AC^0 (Theorem 19). Can we get matching lower and upper bounds?

These prediction problems are closely related to what we call *small positive Collatz cycles*. Indeed, if predicting M (Open Problem 1) was easy, one could hope to easily answer whether there exists $x > 2$ such that $x = T^{\le \lceil \log_2(x) \rceil}(x)$:

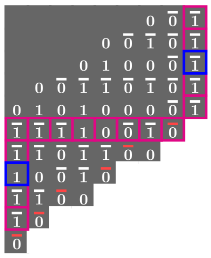


Fig. 5. A small positive cycle, almost. The rightmost magenta column encodes $z = [\![\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}]\!]_{3'} = 242$ in base $3'$. By Lemma 10, the leftmost magenta column is $T^{\lceil \log_2(x) \rceil}(x) = [\![\bar{1}\bar{1}1\bar{1}\bar{1}]\!]_{3'} = [\![22122]\!]_3 = 233$. The numbers 242 and 233 differ in only one trit (in blue): they almost generate a small positive Collatz cycle (Definition 24). Theorem 11 tells us that the region above the magenta row is characterised by base conversion, what about the region below?

Definition 24 (Small positive Collatz cycles). Let $x \in \mathbb{N}^+$. We say that x generates a small positive Collatz cycle if $x = T^m(x)$ with $0 < m \leq \lceil \log_2(x) \rceil$.

Open Problem 2. There are no small positive Collatz cycles besides $(2, 1, 2, 1, \ldots)$.

Example 25. Figure 5 shows that answering the question about small positive Collatz cycle seems challenging. Indeed, the Figure illustrates that for $x = [\![\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}]\!]_{3'} = [\![22222]\!]_3 = 242$, we have $T^{\lceil \log_2(x) \rceil}(x) = [\![22122]\!]_3 = 233$. The two numbers differ only in one trit, it is *almost* a small positive Collatz cycle.

Remark 26. One can also reformulate the small positive Collatz cycles problem by saying that, although the Collatz process is able to compute base $3'$-to-2 conversion, it can never compute base 2-to-$3'$ in $\lceil \log_2(x) \rceil$ CQCA steps (except for $x = 2$). Note that the CQCA would have to produce base 3 trits, from most to least significant trit which seems very hard to in the $\lceil \log_2(x) \rceil$ time constraint.

4 Discussion: Structural Implications on Collatz Sequences

Figure 4 summarises some strong consequences of our results on the structure of Collatz sequences. First, since columns are iterating the Collatz process in base 3 (Lemma 10), the base conversion theorem implies that any $z \in \mathbb{N}$ is giving some rather specific constraints on the next $\lceil \log_2 z \rceil$ iterations of the Collatz process. Specifically, the upper z-rectangle, Fig. 4(b), which corresponds to the diagram of the conversion of z from base $3'$ to base 2, is constraining, on average, half of the trits (base 3 digits) of any iteration $T^{\leq \lceil \log_2 z \rceil (z)}(z)$, Fig. 4(d).

Second, Theorem 23 tells us that this upper z-rectangle is easy to predict, in the sense that the entire region can be computed in NC^1. The computational complexity of lower z-rectangle prediction remains open.

Third, Fig. 4(b) illustrates how the parity checking result of Corollary 15 places constraints on future iterates. A sum bit at *any* position $e \in \mathbb{Z}^2$ of a configuration $c_\infty[\alpha]$ is constrained to be the parity of the number of 1 s in the entire column (both sums and carries) whose base is at the north-east of e.

We should note that these phenomena are occurring everywhere, at each Collatz iterate. Hence, although patterns have been notoriously hard to identify in the Collatz process, our results give a new lens which reveals some detail.

Acknowledgement. Sincere thanks to Olivier Rozier for fruitful interactions, and to the anonymous reviewers for helpful comments.

References

1. Adamczewski, B., Bugeaud, Y.: On the complexity of algebraic numbers I. Expansions in integer bases. Ann. Math. **165**(2), 547–565 (2007). https://doi.org/10.4007/annals.2007.165.547
2. Adamczewski, B., Faverjon, C.: Mahler's method in several variables II: Applications to base change problems and finite automata, September 2018. https://arxiv.org/abs/1809.04826
3. Bruschi, M.: Two cellular automata for the 3x+1 map (2005). https://arxiv.org/abs/nlin/0502061
4. Capco, J.: Odd Collatz sequence and binary representations, March 2019. https://hal.archives-ouvertes.fr/hal-02062503
5. Caruso, X.: Computations with p-adic numbers. Journées Nationales de Calcul Formel. Les cours du CIRM (2018). https://hal.archives-ouvertes.fr/hal-01444183
6. Cloney, T., Goles, E., Vichniac, G.Y.: The $3x+1$ problem: a quasi cellular automaton. Complex Syst. **1**(2), 349–360 (1987)
7. Cloney, T., Goles, E., Vichniac, G.Y.: The $3x+1$ problem: a quasi cellular automaton. Complex Syst. **1**(2), 349–360 (1987)
8. Conway, J.: Unpredictable iterations. In: Number Theory Conference (1972)
9. Goles, E., Ollinger, N., Theyssier, G.: Introducing freezing cellular automata. In: Exploratory Papers of Cellular Automata and Discrete Complex Systems (AUTOMATA 2015), pp. 65–73 (2015)
10. Hesse, W., Allender, E., Barrington, D.A.M.: Uniform constant-depth threshold circuits for division and iterated multiplication. J. Comput. Syst. Sci. **65**(4), 695–716 (2002)
11. Hew, P.C.: Working in binary protects the repetends of $1/3^{\mathrm{h}}$: comment on Colussi's 'The convergence classes of Collatz function'. Theor. Comput. Sci. **618**, 135–141 (2016). https://doi.org/10.1016/j.tcs.2015.12.033
12. Jakobsen, S.K., Simonsen, J.G.: Liouville numbers and the computational complexity of changing bases. In: Anselmo, M., Della Vedova, G., Manea, F., Pauly, A. (eds.) Beyond the Horizon of Computability. CiE 2020. Lecture Notes in Computer Science, vol. 12098, pp. 50–62. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51466-2_5
13. Kari, J.: Cellular automata, the Collatz conjecture and powers of 3/2. In: Yen, H.C., Ibarra, O.H. (eds.) Developments in Language Theory. DLT 2012. Lecture Notes in Computer Science, vol. 7410, pp. 40–49. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31653-1_5

14. Koiran, P., Moore, C.: Closed-form analytic maps in one and two dimensions can simulate universal turing machines. Theoret. Comput. Sci. **210**(1), 217–223 (1999). https://doi.org/10.1016/s0304-3975(98)00117-0
15. Korec, I.: The $3x + 1$ problem, generalized pascal triangles and cellular automata. Mathematica Slovaca **42**(5), 547–563 (1992). http://eudml.org/doc/32424
16. Kurtz, S.A., Simon, J.: The undecidability of the generalized Collatz Problem. In: Cai, J.Y., Cooper, S.B., Zhu, H. (eds.) Theory and Applications of Models of Computation. TAMC 2007. Lecture Notes in Computer Science, vol. 4484, pp. 542–553. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72504-6_49
17. Lagarias, J.C.: The $3x+1$ problem and its generalizations. Am. Math. Mon. **92**(1), 3–23 (1985). http://www.jstor.org/stable/2322189
18. Minsky, M.: Computation: Finite and Infinite Machines. Prentice Hall Inc., Engelwood Cliffs (1967)
19. Monks, K.G., Yazinski, J.: The autoconjugacy of the 3x+1 function. Discrete Math. **275**(1–3), 219–236 (2004). https://doi.org/10.1016/s0012-365x(03)00125-0
20. Moore, C.: Unpredictability and undecidability in dynamical systems. Phys. Rev. Lett. **64**(20), 2354 (1990)
21. Moore, C.: Generalized shifts: unpredictability and undecidability in dynamical systems. Nonlinearity **4**(2), 199 (1991)
22. Moore, C., Mertens, S.: The Nature of Computation. Oxford University Press, Oxford (2011)
23. Rozier, O.: Parity sequences of the 3x+1 map on the 2-adic integers and Euclidean embedding. Integers **19**, A8 (2019)
24. Shallit, J., Wilson, D.A.: The “3x + 1” problem and finite automata. Bull. EATCS **46**, 182–185 (1992)
25. Stérin, T.: Binary expression of ancestors in the Collatz graph. In: Schmitz, S., Potapov, I. (eds.) 14th International Conference on Reachability Problems. LNCS, Springer (2020). (To appear). https://arxiv.org/abs/1907.00775v4
26. Stérin, T., Woods, D.: The Collatz process embeds a base conversion algorithm (2020). Expanded version. https://arxiv.org/abs/2007.06979v3
27. Vollmar, R.: On cellular automata with a finite number of state changes. In: Knödel, W., Schneider, H.J. (eds.) Parallel Processes and Related Automata/Parallele Prozesse und damit zusammenhängende Automaten. Computing Supplementum, vol. 3, pp. 181–191. Springer, Vienna (1981). https://doi.org/10.1007/978-3-7091-8596-4_13