# Parity problem with a cellular automaton solution

K. M. Lee, Hao Xu, and H. F. Chau*

*Department of Physics, University of Hong Kong, Pokfulam Road, Hong Kong*

The parity of a bit string of length $N$ is a global quantity that can be efficiently computed using a global counter in $O(N)$ time. But is it possible to find the parity using cellular automata with a set of local rule tables without using any global counter? Here, we report a way to solve this problem using a number of $r=1$ binary, uniform, parallel, and deterministic cellular automata applied in succession for a total of $O(N^2)$ time.

Cellular automaton (CA) is a simple local interaction model used to study the evolution and self-organization of various physical and biological systems [1]. And at the same time, CA can also be viewed as a restrictive model of parallel computation without common global memory. In fact, recently there is an increasing interest in using CA to perform certain computational tasks [2]. It is therefore natural to ask if CA can be used to perform a task that depends on the global information of an input state.

An example of this kind is called the density classification problem (DCP). In this problem, we are given an one dimensional array of bit string in periodic boundary conditions. We are required to apply some CA rules so as to evolve the state to all zero if the number of zeros in the input state is greater than that of ones. Similarly, we have to evolve the state to all one if the number of zeros in the input state is less than that of ones.

After a number of fail attempts, Land and Belew proved that no single CA can solve the DCP without making some mistake [3]. And yet later on, Capcarrere and co-workers argued that a single CA rule can solve the DCP provided that we modify both the required output of the automaton and the boundary conditions used [4,5]. Nonetheless, we do not completely agree with their approach for we have to scan through the states of the entire final bit string, in general, before knowing the answer. This requires either global memories or a table that scales with the system size in the read out; hence it somehow defeats the purpose of restricting ourselves to the use of CA in the first place. (In contrast, the read out in the original DCP can be determined by looking at any one or two bits in the final string.) Although DCP cannot be solved using one CA rule, Fukś showed that this can be done by applying two CA rules in succession. More precisely, he found a solution to the DCP by applying the CA rule 184 a fixed number of times depending only on the string length and then followed by the CA rule 232 a fixed number of times again depending only on the string length [6]. A number of related problems depending on the global density of a string have also found to have CA solutions [7].

Another challenge for CA is the parity problem (PP), namely, to evolve a given input bit string $\sigma$ using a sequence of CA rules to all $P_\sigma$s where $P_\sigma$ is the parity of $\sigma$. (Then the parity of the input bit string can be determined by looking at

any one of the final bit string.) This problem appears to be much harder than the DCP because the output is altered simply by a flip in any one of the input bits. In fact, Sipper proved that no single $r=1$ CA rule can solve the PP with fixed boundary conditions [8].

In this paper, we show that PP can be solved by applying a number of $r=1$ CA rules in succession. The term CA in this paper shall mean a local synchronous uniform deterministic binary CA rule with parallel update in periodic boundary conditions. That is, the state of each bit in the next time step depends deterministically only on the state of its finite neighborhood. The states of all sites are updated in parallel and the rule table of the CA is covariant under translation along the bit string. Besides, we restrict ourselves to consider only those sequence of CAs that solves the PP exactly without making any misclassification.

To have a feeling of the difficulty of this problem, two remarks are in place. First, except for the boundary conditions, relaxing any of the above conditions makes the PP trivial. For example, applying the CA rule 60 first to the 2nd bit, then the 3rd bit, and so on till the last bit in the bit string, the value of the last bit is the parity of our input string. Second, one set of local CA rules is not sufficient to solve the PP for input string of *any* length. The reason is simple: if such a set could solve the PP, it would evolve an odd parity bit string $\sigma$ to all one. The uniformity condition implies that this set would evolve the concatenated even parity bit string $\sigma\sigma$ to all one as well. But this is absurd.

Thus, we need to invoke more than one CA rules to solve the PP. In fact, all we need is a few $r=1$ CA rules reported below in Wolfram's notations [9].

(1) $R_{222}$ is the Wolfram elementary CA rule 222.

$$R_{222}: \quad \frac{000\ 001\ 010\ 011\ 100\ 101\ 110\ 111}{0\quad 1\quad 1\quad 1\quad 1\quad 0\quad 1\quad 1}.$$

This rule replaces the two ending zeros of a string of zeros by ones, if the number of zeros in the string is more than two. Since we are using periodic boundary conditions, the (global) parity of the configuration does not change although locally parity does change. If we apply the rule $\lfloor N/2 \rfloor$ times, then there will be no more consecutive zeros.

(2) $R_{132}$ is similar to $R_{222}$, but it replaces the ending ones by zeros. It also conserves parity.

*Email address: hfchau@hkusua.hku.hk

$$R_{132}: \quad \frac{000\ 001\ 010\ 011\ 100\ 101\ 110\ 111}{0\quad 0\quad 1\quad 0\quad 0\quad 0\quad 0\quad 1}.$$

(3) $R_{76}$ will flip a configuration of all ones to all zeros.

$$R_{76}: \quad \frac{000\ 001\ 010\ 011\ 100\ 101\ 110\ 111}{0\quad 0\quad 1\quad 1\quad 0\quad 0\quad 1\quad 0}.$$

(4) $R_{254}$ preserves the configuration of all zeros. In fact, careful application of $R_{254}$ together with $R_{76}$ will put our result to the desired form.

$$R_{254}: \quad \frac{000\ 001\ 010\ 011\ 100\ 101\ 110\ 111}{0\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1}.$$

(5) $R_{184}$ is the so-called traffic rule. It tries to move a one to the right if the site at right is a zero.

$$R_{184}: \quad \frac{000\ 001\ 010\ 011\ 100\ 101\ 110\ 111}{0\quad 0\quad 0\quad 1\quad 1\quad 1\quad 0\quad 1}.$$

(6) $R_{252}$ is an auxiliary rule to change a zero right of an one to one.

$$R_{252}: \quad \frac{000\ 001\ 010\ 011\ 100\ 101\ 110\ 111}{0\quad 0\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1}.$$

Let $\sigma$ be an arbitrary input bit string of length $N$, and $P_\sigma$ its parity. We write $R\sigma$ the resulting configuration after the rule $R$ is applied to $\sigma$ once, for any CA rule $R$.

*Theorem.* Let

$$\sigma_1 \equiv (R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor})^{\lfloor N/2 \rfloor}\sigma, \tag{1}$$

which means that we apply $R_{222}$ to $\sigma$ $\lfloor N/2 \rfloor$ times, then apply $R_{132}$ $\lfloor N/2 \rfloor$ times and back to $R_{222}$, and so on. The total number of time steps is $2\lfloor N/2 \rfloor^2$.

(a) If $N$ is odd, then

$$\sigma_1 = \begin{cases} 0^N & \text{if } P_\sigma = 0, \\ 1^N & \text{if } P_\sigma = 1, \end{cases} \tag{2}$$

where $0^N$ denotes a string of $N$ consecutive zeros, that is, the state at all the sites is zero; and similarly for $1^N$. Total number of time steps is $(N-1)^2/2$.

(b) If $N = 2q$, where $q$ is odd, then

$$\sigma_1 = \begin{cases} 0^N \text{ or } 1^N & \text{if } P_\sigma = 0, \\ \text{other than } 0^N \text{ and } 1^N & \text{if } P_\sigma = 1. \end{cases} \tag{3}$$

Applying the CA rules $S \equiv R_{254}^{\lfloor N/2 \rfloor} R_{76}$, we can transform $\sigma_1$ to the form given in (a):

$$S\sigma_1 = \begin{cases} 0^N & \text{if } P_\sigma = 0, \\ 1^N & \text{if } P_\sigma = 1. \end{cases} \tag{4}$$

The total number of time steps is $(N^2/2 + N/2 + 1)$.

(c) If $N = 2^m q$, where $m \geqslant 2$ and $q$ is odd, let

$$T \equiv R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor} R_{184} R_{252} \tag{5}$$

and

$$\sigma_2 \equiv T^{m-1}\sigma_1, \tag{6}$$

then

$$\sigma_2 = \begin{cases} 0^N \text{ or } 1^N & \text{if } P_\sigma = 0, \\ \text{other than } 0^N \text{ and } 1^N & \text{if } P_\sigma = 1. \end{cases} \tag{7}$$

Similar to (b), $S\sigma_2$ is either all zeros or all ones, depending on whether $P_\sigma$ is even or odd. The total number of steps is $N^2/2 + (m-1)(N+2) + N/2 + 1$.

*Remark.* If we are allowed to change the lattice size, then we do not need (b) and (c) of the Theorem to solve the parity classification problem. Suppose we are given a bit string with even length. To find out the parity of the number of ones in it, we just concatenate a single zero bit to the bit string. The resultant bit string will be in odd length and hence we are back to (a) of the Theorem.

The following two Lemmas are required to prove this Theorem.

*Lemma 1.* For any $N$, $\sigma_1$ could only be one of the following three forms: $0^N$, $1^N$ or $(10^{2l-1})^k$, where $l, k \geqslant 1$ are integers, $2lk = N$ and $P_\sigma = k \pmod 2$. [The notation $(10^{2l-1})^k$ means, for example, if $l=2$ and $k=3$, the bit string is 100010001000.]

*Proof.* It is obvious that both $0^N$ and $1^N$ are fixed points of $R_{222}$ and $R_{132}$. Thus, in the rest of this proof, we shall only consider configurations $\sigma$ with both zeros and ones.

As we have discussed above, for any configuration $\sigma$, the string $R_{222}^{\lfloor N/2 \rfloor}\sigma$ has no consecutive zeros. Thus, its general form will look like

$$R_{222}^{\lfloor N/2 \rfloor}\sigma = \cdots 101^{n_1}01^{n_2}01\cdots, \tag{8}$$

where the numbers of ones between two zeros $n_i$ are greater than or equal to one. (The trivial configuration $1^N$ is also possible.)

Since $R_{132}$ replaces the ending ones by zeros, if any of the $n_i$ is even, the corresponding string of ones will be completely annihilated after applying $R_{132}^{\lfloor N/2 \rfloor}$. For example,

$$R_{132}^2(\cdots 0101111010 \cdots) = R_{132}(\cdots 0100110010 \cdots)$$
$$= \cdots 0100000010 \cdots . \tag{9}$$

This example also shows that after a single pass of $R \equiv R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor}$, we cannot conclude that the number of zeros between two ones is odd. However, we can conclude that if there is a string of even number of zeros between two ones, after a single pass of $R$, the number of zeros in that string will increase by at least two. Thus, $R^{\lfloor N/2 \rfloor}\sigma$ must be of the form

$$\cdots 10^{2n_1'+1}10^{2n_2'+1}1\cdots, \tag{10}$$

where $n_i'$ are non-negative integers (or the trivial cases $0^N$ or $1^N$).

We still need to show that the $n_i'$ are all equal. This can be illustrated by an example, consider the case $N=8$,

$$R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor} (10100000) = R_{132}^{\lfloor N/2 \rfloor} (10111011)$$

$$= (00010001). \qquad (11)$$

We see that the numbers of zeros between the ones tend to equalize, from one and five zeros to three zeros in this example. More precisely, $R_{222}^{\lfloor N/2 \rfloor} (10^{2n'_1 + 1} 10^{2n'_2 + 1} 1 \cdots)$ equals $01^{n'_1 + n'_2 + 1} 01^{n'_2 + n'_3 + 1} 0 \cdots$ up to a translation. So, after applying $R$ for sufficiently long time, the resultant string will be in the form $1^N$, $0^N$ or Eq. (10). Since the first two cases are fixed points of $R$, we only need to focus on the third case. If the resultant string is in the third form, then it is easy to check that the total number of ones does not increase after each application of $R$. Hence, eventually the number of ones in the string will stay constant under repeated application of $R$. And this happens if and only if $n'_i$ are all even or all odd. In this case,

$$R(10^{2n'_1 + 1} 10^{2n'_2 + 1} 1 \cdots)$$

$$= 10^{1 + n'_2 + (n'_1 + n'_3)/2} 10^{1 + n'_3 + (n'_2 + n'_4)/2} 1 \cdots \quad (12)$$

up to a translation. Hence, the repeated application of $R$ equalizes the number of zeros between the ones; and a configuration in the form $(10^{2n+1})^l$ is a fixed point of $R$. Finally, it is straight forward to check that at most $\lfloor N/2 \rfloor$ applications of $R$ is enough to bring a bit string to the fixed points of the form $0^N$, $1^N$, or $(10^{2l-1})^k$. [One of the worst cases is the configuration $1(10)^{(N-1)/2}$ for $N$ odd.]

Since $R_{132}$ and $R_{222}$ conserve parity, $k$ (mod 2) is equal to the parity of $(10^{2l-1})^k$, which is just $P_\sigma$. This completes the proof of Lemma 1.

*Lemma 2.* With the notations of the Theorem, we have

$$T 0^N = 0^N, \qquad (13)$$

$$T 1^N = 1^N, \qquad (14)$$

$$T(10^{2l-1})^k = \begin{cases} (10^{l-1})^{2k} & \text{if } l \text{ is even,} \\ 1^N & \text{if } l = 1, \\ 0^N & \text{if } l \geq 3 \text{ is odd,} \end{cases} \quad (15)$$

where $k \geq 1$.

*Proof.* Eq. (13) and Eq. (14) are trivial. For Eq. (15), we consider two cases. If $l \geq 2$,

$$T(10^{2l-1})^k = (R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor} R_{184} R_{252})(10^{2l-1})^k$$

$$= (R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor} R_{184})(110^{2l-2})^k$$

$$= (R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor})(1010^{2l-3})^k$$

$$= \begin{cases} (10^{l-1} 10^{l-1})^k & \text{if } l \text{ is even,} \\ (0^{2l})^k & \text{if } l \text{ is odd.} \end{cases} \quad (16)$$

If $l = 1$,

$$T(10)^k = (R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor} R_{184} R_{252})(10)^k$$

$$= (R_{132}^{\lfloor N/2 \rfloor} R_{222}^{\lfloor N/2 \rfloor} R_{184}) 1^N$$

$$= 1^N. \qquad (17)$$

This concludes the proof of Lemma 2. ∎

*Proof of the Theorem.* (a) If $N$ is odd, among the three forms provided by Lemma 1, $(10^{2l-1})^k$ cannot be reached. Since parity is conserved, the final configuration can only be $0^N$ or $1^N$ according to the initial parity.

(b) If $N = 2q$, parity of $0^N$ and $1^N$ are even, but the parity of $(10^{2l-1})^k$ must be odd because $k$ is odd. Thus, we have proved Eq. (3). Now, by flipping the all ones to all zeros and changing all other configurations to all ones by the rules $S$, we put the final configuration to a form similar to (a).

(c) If $P_\sigma = 1$, then $P_\sigma = 1 = k$ (mod 2), and $l = 2^{m-1} q'$ where $q'$ is odd. By Lemma 2, we have

$$T^{m-1}(10^{2l-1})^k = T^{m-1}(10^{2^m q' - 1})^k$$

$$= T^{m-2}(10^{2^{m-1} q' - 1})^{2k}$$

$$= \cdots = (10^{2q' - 1})^{2^{m-1} k}. \qquad (18)$$

If $P_\sigma = 0$, then $l = 2^{m'-1} q'$ where $m' < m$ and $q'$ is odd. The above equation does not reach the last line and $T^{m-1}(10^{2l-1})^k$ is $0^N$ or $1^N$. This completes the proof of the Theorem. ∎

To summarize, we show that it is impossible for any single set of CA rules to correctly compute the parity of a bit string. But surprisingly, we find a CA solution to the PP using a sequence of $r = 1$ CAs. For an input bit string of length $N$, the worst case run time scales as $O(N^2)$. However, we have no idea if the present method has the shortest worst case run time or not. And since the PP is equivalent to the computation of the sum of a given bit string modulo two, our result implies that CA can be used to count the number of ones in a bit string modulo two. In other words, if we denote the number of ones in a bit string $\sigma$ by $\#\sigma$, then one can use a sequence of CAs to compute the least significant bit of $\#\sigma$ written in binary notation. Besides, the CA solution of the DCP for a bit string $2^n - 1$ long can then be regarded as a way to compute the most significant bit of $\#\sigma$. Therefore, it is instructive to investigate the possibility of using CAs to compute any given bit of $\#\sigma$ and hence to count the number of ones in a bit string provided that we have $O(\log N)$ copies. Along a similar line of thought, it is also worthwhile to look for CA solutions to the problem of addition over a finite ring or field.

[1] See, for example, B. Chopard and M. Droz, *Cellular Automata Modeling of Physical Systems* (Cambridge University Press, Cambridge, 1998).

[2] M. Mitchell, P.T. Hraber, and J.P. Crutchfield, Complex Syst. **7**, 89 (1993); S.C. Benjamin and N.J. Johnson, Appl. Phys. Lett. **70**, 2321 (1997); M. Sipper, Computer **32**, 18 (1999).

[3] M. Land and R.K. Belew, Phys. Rev. Lett. **74**, 5148 (1995).

[4] M.S. Capcarrere, M. Sipper, and M. Tomassini, Phys. Rev. Lett. **77**, 4969 (1996).

[5] M. Sipper, M.S. Capcarrere, and E. Ronald, Int. J. Mod. Phys. C **9**, 899 (1998).

[6] H. Fukś, Phys. Rev. E **55**, R2081 (1997).

[7] H.F. Chau, K.K. Yan, K.Y. Wan, and L.W. Siu, Phys. Rev. E **57**, 1367 (1998); H.F. Chau, L.W. Siu, and K.K. Yan, Int. J. Mod. Phys. C **10**, 883 (1999).

[8] M. Sipper, Phys. Rev. E **57**, 3589 (1998).

[9] S. Wolfram, Rev. Mod. Phys. **55**, 601 (1983).