The Representation Role for Basic Operations Embodied in Cellular Automata: A Suitability Example for Addition in Redundant Numeral Systems vs Conventional Ones

Salvatore Di Gregorio[1,2(✉)]

[1] Department of Mathematics and Computer Science, University of Calabria, 87036 Rende, CS, Italy
salvatore.digregorio@unical.it
[2] ISAC - CNR, Lamezia Terme Zona Industriale, 88046 Lamezia Terme, CZ, Italy

Abstract. Cellular Automata (CA) are both a parallel computational paradigm and an archetype for modelling complex systems, that evolve on the basis of local interactions. CA can embody different numeral representations and perform related basic arithmetical operations. However, conventional numeral representations are thought as intrinsically sequential in such operations, which implies that CA parallelism is underexploited when CA evolution mimics the sequentiality of calculation, while some redundant numeral representations could exalt the CA parallelism in a space/time trade-off, where the time complexity of some operations is constant on input length. The problem then arises when the result of an operation must be utilized in the conventional representation since, usually, the migration toward an advantageous redundant numeric representation is costless, but the inverse one implies necessarily a cost that cancels the benefits in terms of computation time. This paper explores the properties of the conventional binary positional representation embodied in a CA together with the addition operation and the corresponding ones of a redundant binary positional representation, the rules and time cost for the passage from conventional numeral system to redundant one and vice versa. The results permit to individuate the CA computation context, when redundancy could be exploited advantageously. It regards cases where a longest sequence of additions (or operations based on addition, e.g., fast Fourier transforms) has to be performed in well-defined short times as for the automatic control of mobile devices.

Keywords: Cellular Automata
Non-conventional positional numeral binary systems · Addition

1 Introduction

Cellular Automata (CA) were born with a paradox: von Neumann [1] embodied in a cellular space of finite states automata a modified Turing Machine in

G. Mauri et al. (Eds.): ACRI 2018, LNCS 11115, pp. 307–318, 2018.
https://doi.org/10.1007/978-3-319-99813-8_28

order to guarantee universal computation in the self-reproduction mechanisms. In such a way, a purely parallel computing device supports a purely sequential computation. CA are both a parallel computational paradigm and an archetype for modelling 'systems', that are extended in space and evolve on the basis of local interactions [2]. Using CA is suitable in such a type of context, even if a substantially sequential computational behavior could be easily hidden in many cases.

This question reveals distinctly itself for the case of the same numerical operations performed inside CA in numeral systems, that are related to the same set of numbers, but differ in their representation. Here another factor, the representation, comes into play, but the question cannot simply be treated in terms of time cost efficiency because in solving a particular problem, a type of representation could be mandatory for expressing solutions and/or the input data could be available only in a specific representation. Therefore it is necessary to investigate efficient translation methods in order to communicate between two or more worlds with different representations, but with the same basic operations. If we look at the single operation, e.g. the addition for the conventional vs redundant numeral systems, it is important to know if there is advantage in passing from a representation to another and returning to the previous one.

Nevertheless a criterion of computational cost effectiveness for a specific problem may be defined only if we consider the algorithmic features of the problem in terms of sequence of basic operations in the context of possible diverse representations and the eventual computational costs for passages from a representation to another one and vice versa. So the question does not regard the single operation but a specific problem, all having to be related for homogeneity to a single computational paradigm, that are in our case CA, where sequentiality can coexist with the structure parallelism.

In this paper, we consider the conventional binary representations vs a possible corresponding redundant binary representation for the addition operation on the set of natural numbers $\mathbb{N}$ and their implementation inside CA, furthermore opportune operations are evaluated in the same context for passing from a one representation to the other and vice versa.

The CA approach to 'fast' addition of binary numbers of Sheth et al. [3] is revisited as reference point for conventional representation of $\mathbb{N}$. This operation of binary addition was implemented on a Cellular Automata Machine (CAM-8 machine) [4]. A corresponding redundant representation, that is here presented together with the related addition operation, was studied and developed for basic arithmetic operations of integer numbers at the University of Calabria in some 'Laurea' theses and reports, e.g. [6], a similar representation for the set of integer numbers $\mathbb{Z}$ was adopted for addition implementation on the same CAM-8 by Clementi et al. in [5]. Mechanisms of translation between conventional and redundant representation of $\mathbb{N}$ on CA is investigated. Hardware implementations as in [7,8] are not here considered, but they can be deduced straightforwardly in manifold ways, FPGA integrated circuits, e.g. [9], could be more significant for using CA redundant arithmetic also in broader contexts. Anyway, the aim

of this paper is a comparison between CA embodying two different numeral representations and efficient passage mechanisms from one to other and vice versa.

A CA performing the addition operations in the conventional binary representation for $\mathbb{N}$ (CBN) is presented in the next section, the third section introduces a CA, that performs addition operations in a redundant binary representation for $\mathbb{N}$ (RBN), RBN properties are defined, rules of passage between CBN CA and RBN CA are established. Conclusions and comments end the paper.

2 CA for Addition in the Conventional Binary Representation

Intuitively a homogeneous CA can be seen as a d-dimensional space, partitioned in cells of uniform size, each one embedding an identical finite states automaton, the elementary automaton (ea).

Input for each cell is given by the states of the neighboring cells, where the neighborhood conditions are determined by a pattern invariant in time and space.

At the time (step) $t = 0$, cells are in arbitrary states and the CA evolves changing the state at discrete times simultaneously, according to the transition function $\tau : S^r \to S$, where S is the finite set of the ea states and r is the number of the neighboring cells.

The following definition (partly from Di Gregorio and Trautteur [10]) for CA is adopted in this paper:

Definition 1. *A Cellular Automaton A is a quadruple $A = \langle \mathbb{Z}^d, X, S, \tau \rangle$ where:*

- *$\mathbb{Z}^d$ is the set of cells identified by points with integer co-ordinates in a Euclidean d-dimensions space; such a formal definition may be extended to different types of spaces (e.g., Riemannian spaces), different topologies (e.g., torus in 2-dimensions spaces), or different tessellations (e.g., hexagonal tessellation for 2-dimensions;*
- *$X = \langle \xi_0, \xi_1, \ldots \xi_{r-1} \rangle$ with $\#X = r$ is the neighborhood index, that is the ordered finite set of d-dimensional vectors, that defines for a generic cell $i = \langle i_1, i_2, \ldots, i_d \rangle$ the set $N(X, i) = \langle i + \xi_0, i + \xi_1, \ldots, i + \xi_{r-1} \rangle$ of the neighboring cells (usually ξ_0 is the null vector);*
- *S is the finite set of states of the elementary automaton. A specification of S as Cartesian product of sets of sub-states: $S = S_1 \times S_2 \times \ldots \times S_s$ is introduced.*
- *$\tau : S^r \to S$ is the deterministic transition function of the elementary automaton;*

furthermore:

- *$C = \{c \,|\, c : \mathbb{Z}^d \to S\}$ is the set of possible state assignment to the CA; it is called the CA configuration set; $c(i)$ is the state of the cell i;*
- *$\gamma : C \to C \mapsto [\gamma(c)](i) = \tau(c(N(X, i)))$ for $c \in C$, is the global transition function. A configuration c is stable if $\gamma(c) = c$.*

The following two CA embody the addends as sequence of sub-states in the configurations. So numbers may be so individuated and 'writing' and 'reading' for passage from one numeral representation to another one can be specified.

2.1 CA ADD Definition and Properties

A possible CA ADD for addition of two natural numbers m and n in the conventional binary representation CBN is here defined as a 1-dimension CA with ring topology of l cells with $l > max(\lceil \log_2 m \rceil), (\lceil \log_2 n \rceil)$:

Definition 2. *$ADD = (Z_l, X, S, \tau)$ where:*

- *$Z_l = \langle l-1, l-2, \cdots, 1, 0 \rangle$ is the finite cellular space of length l with ring topology and reverse numeration of cells by formalization convenience;*
- *$X = \langle 0, -1 \rangle$ is the neighborhood: the cell itself and the 'right' one;*
- *$S = S_1 \times S_2$, the set of states with $S_1 = S_2 = 0, 1$, the four states are represented as $\{{}^{0}_{0}, {}^{0}_{1}, {}^{1}_{0}, {}^{1}_{1}\}$ where, for a configuration c, the former (upper) bit in the cell i is the i^{th} bit of the former addend m specified as m_i and the latter (lower) bit in the cell i is the i^{th} bit of the latter addend n specified as n_i, both with positional weight 2^i; m and n are respectively the upper and the lower addends of c (see Fig. 1).*
- *$\tau : S^2 \rightarrow S$ is the transition function so defined from the following equations, where two configurations c' and c'' are considered such that $c'' = \gamma(c')$:*
  1. *$m_i^{''} = m_{i-1}^{'} \wedge n_{i-1}^{'}$, $0 < i < l$; $m_0^{''} = m_{l-1}^{'} \wedge n_{l-1}^{'}$ by the ring topology;*
  2. *$n_i^{''} = m_i^{'} \oplus n_i^{'}$, $0 \leq i < l$;*

  being m' and n' respectively the former and latter addend of c', m'' and n'' respectively the upper and lower addend of c'', where $m_i^{''}$ is the carry bit with positional weight 2^i of the sum $m_{i-1}^{'} + n_{i-1}^{'}$; $n_i^{''}$ is the 'lesser' bit with positional weight 2^i of the sum $m_i^{'} + n_i^{'}$.

m (25)	→	0	0	0	1	1	0	0	1
n (15)	→	0	0	0	0	1	1	1	1
Positional weight	→	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Fig. 1. An example of ADD configuration c (highlighted) with $l = 8$; the upper sequence of bits is the former addend m, the lower sequence is the latter addend n; the positional weight of each cell is specified below, values in base 10 of m and n are on the left in brackets.

The ring topology of ADD (therefore a finite number of cells) involves that additions are performed properly, only if there is no overflow, i.e., significant length of numbers doesn't overcome $l - 1$ bits, because the last cell is neighbor to the first one; l may be large at will, so a sufficient length of bits may be always

assumed (sufficient length condition). The ADD configuration example of Fig. 1 specifies the positional weight of the cells and values of m and n in the base 10 numeration.

Theorem 1. *Let c be a generic configuration of ADD with length l and m, n respectively the upper and lower addend of c; let $c' = \gamma(c)$, m' and n' respectively the former and latter addend of c', then $m' + n' = m + n$ (examples in Fig. 2).*

Proof. $m_{l-1} = 0$, $n_{l-1} = 0$ by the sufficient length condition, therefore $m_0^{'} = 0$, $n_{l-1}^{'} = 0$ then:

$$m + n = \sum_{i=0}^{l-1} (m_i + n_i)2^i = m_0^{'}2^0 + \sum_{i=0}^{l-2} (m_{i+1}^{'}2^{i+1} + n_i^{'}2^i) + n_{l-1}^{'}2^{l-1}$$
$$= \sum_{i=0}^{l-1} (m_i^{'} + n_i^{'})2^i = m^{'} + n^{'}$$

□

Theorem 2. *Let c be a configuration of ADD with m and n respectively the upper and lower addend of c; let $c' = \gamma(c)$ and m' and n' respectively the upper and lower addend of c', if $m_i = 0$ for $0 \leq i < k < l - 1$ then $m_j^{'} = 0$ for $0 \leq j \leq k$.*

Proof. $m_{l-1} = 0$, $n_{l-1} = 0$ by the sufficient length condition, therefore it is always $m_0^{'} = 0$ and $m_i^{'} = m_{i-1} \wedge n_{i-1} = 0$ for $1 \leq i \leq k$ by applying Eq. (1) of the τ specification of ADD. □

Corollary 1. *Let c be a configuration of ADD, $c' = \gamma(c)$ and $c'' = \gamma^{l-1}(c)$, with m, n, m', n', m'', n'', respectively the upper and lower addend of c, c' and c'', then always $m = 0$.*

Proof. $m_{l-1} = 0$, $n_{l-1} = 0$ by the sufficient length condition, therefore always ${m_0}' = 0$, then $m'' = 0$ by Theorem 2. □

Theorem 3. *Let c be a configuration of ADD with $m = 0$ and n respectively the upper and lower addend of c; let $c' = \gamma(c)$ and m' and n' respectively the upper and lower addend of c', $c' = c$ and c is a stable configuration (see example in Fig. 2).*

Proof. $m_0^{'} = m_{l-1} \wedge n_{l-1} = 0 \wedge n_{l-1} = 0$, $m_i^{'} = m_{i-1} \wedge n_{i-1} = 0 \wedge n_{i-1} = 0$ for $1 \leq i \leq l-1$, by applying Eq. 1 of the τ specification of ADD; by applying Eq. 2 of the τ specification of ADD, ${n_i}' = m_i \oplus n_i = 0 \oplus n_i = n_i$ for $0 \leq i \leq l - 1$. □

Corollary 2. *Let c be a generic configuration of ADD of length l with m and n respectively the upper and lower addend of c, $c' = \gamma^{l-1}(c)$, with m', n', respectively the upper and lower addend of c', then it is always $m' = 0$ and $n' = m + n$ after $l - 1$ steps (see Fig. 2).*

Proof. $m' = 0$ from Corollary 2, $m + n = m' + n'$ from Theorem 1, therefore $n' = m + n$. □

The addition is performed by ADD in $l - 1$ steps in the worst case, therefore the time cost is $\mathcal{O}(l)$. An example of ADD evolution with length $l = 8$ is presented in Fig. 2, where the stable configuration is obtained after 4 steps.

ADD parallelism speeds up addition in irregular way, it depends on how short is the longest sequence of consecutive carries 1 in the conventional arithmetic operation of addition.

An extension of ADD for integers according to the two complement representation could be developed in several ways; the most intuitive way is breaking the ring between cells 0 and $l-1$ (cell $l-1$ assumes a positional weight of -2^{l-1}) and considering that the state of the -1 neighbor of cell 0 (now such a neighbor no longer exists) is always acquired as $\frac{0}{0}$. The operability holds for integers in the interval $[-2^{l-1},\ 2^{l-1}]$.

Positional weight	→	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
m (25)		0	0	0	1	1	0	0	1
n (15)		0	0	0	0	1	1	1	1
	t=0								
m (18)		0	0	0	1	0	0	1	0
n (22)		0	0	0	1	0	1	1	0
	t=1								
m (36)		0	0	1	0	0	1	0	0
n (4)		0	0	0	0	0	1	0	0
	t=2								
m (8)		0	0	0	0	1	0	0	0
n (32)		0	0	1	0	0	0	0	0
	t=3								
m (0)		0	0	0	0	0	0	0	0
n (40)		0	0	1	0	1	0	0	0
	t=4								
m (0)		0	0	0	0	0	0	0	0
n (40)		0	0	1	0	1	0	0	0
	t=5								

Fig. 2. Evolution example of ADD with length 8 for 5 steps (t). Configurations are highlighted, the upper sequence of bit is the former addend m, the lower one is the latter addend n, their values in base 10 are on the left in brackets; the cell positional weight is specified on top.

3 CA for Addition in a Redundant Binary Representation

3.1 The Redundant Binary Representation RBN for $\mathbb{N}$

The proposed redundant binary representation RBN is very similar to those presented in [5,6]; it differs from CBN because the same positional weight is assigned to a couple of consecutive bits, this involves that there are more sequences of bits for the same value (except 0).

Definition 3. *RBN associates to a sequence of* $2l$ *bits:* $b_{2l-1}, b_{2l-2}, \ldots, b_0$*, the value:*

$$b = \sum_{i=0}^{2l-1} b_i 2^{\lfloor \frac{i}{2} \rfloor}$$

Examples:

- 1001 in RBN gives $1 \cdot 2^{\lfloor 3/2 \rfloor} + 0 \cdot 2^{\lfloor 2/2 \rfloor} + 0 \cdot 2^{\lfloor 1/2 \rfloor} + 1 \cdot 2^{\lfloor 0/2 \rfloor} = 3$
- 110 in RBN gives $1 \cdot 2^{\lfloor 2/2 \rfloor} + 1 \cdot 2^{\lfloor 1/2 \rfloor} + 0 \cdot 2^{\lfloor 0/2 \rfloor} = 3$

Definition 4. *A string of bits representing in RBN a natural number* n *is called canonical form* $\alpha(\beta)$ *of* n *if each even (odd) bit is 0.*

By the previous definition, if even (odd) 0 digits are removed from a canonical form $\alpha(\beta)$ of RBN, a binary string is obtained with the same value in CBN; if we put the 0 digit at the right (at the left) of each digit of a binary string representing a numerical value in CBN, a canonical form $\alpha(\beta)$ is obtained with the same value in RBN, an example is here given for $n = 13$:

$$\begin{array}{rcccl} 10100010 & \leftarrow & 1101 & \rightarrow & 01010001 \\ \text{RBN canonical form } \alpha & \leftarrow & \text{CBN} & \rightarrow & \text{RBN canonical form } \beta \end{array}$$

The passage from a canonical form $\alpha(\beta)$ of RBN to CBN and vice versa may be considered costless in the prospective of CA, as specified afterwards.

From now on, the length of strings of bits in RBN will be always taken even without loss of generality, the canonical form α is abbreviated in cfα.

3.2 CA ADDr Definition and Properties

A CA ADDr for addition of two natural numbers m and n in RBN is here defined as a 1-dimension CA of l cells with ring topology and $l > max(\lceil \log_2 m \rceil), (\lceil \log_2 n \rceil)$

Definition 5. *ADDr* $= (Z_l, X, S, \tau)$ *where:*

- $Z_l = \langle l-1, l-2, \cdots, 1, 0 \rangle$ *is the finite cellular space of length* l *with ring topology and reverse numeration of cells by formalization convenience;*
- $X = \langle 0, -1 \rangle$ *is the neighborhood: the cell itself and the 'right' one;*

- $S = \{\begin{smallmatrix}00\\00\end{smallmatrix}, \begin{smallmatrix}00\\01\end{smallmatrix}, \begin{smallmatrix}01\\00\end{smallmatrix}, \begin{smallmatrix}01\\01\end{smallmatrix}, \begin{smallmatrix}00\\10\end{smallmatrix}, \begin{smallmatrix}00\\11\end{smallmatrix}, \begin{smallmatrix}01\\10\end{smallmatrix}, \begin{smallmatrix}01\\11\end{smallmatrix}, \begin{smallmatrix}10\\00\end{smallmatrix}, \begin{smallmatrix}10\\01\end{smallmatrix}, \begin{smallmatrix}11\\00\end{smallmatrix}, \begin{smallmatrix}11\\01\end{smallmatrix}, \begin{smallmatrix}10\\10\end{smallmatrix}, \begin{smallmatrix}10\\11\end{smallmatrix}, \begin{smallmatrix}11\\10\end{smallmatrix}, \begin{smallmatrix}11\\11\end{smallmatrix}\}$ *is the set of states, ($S = S_1 \times S_2$, with $S_1 = S_2 = \{00, 01, 10, 11\}$ the 4 couples of bits); the former (upper) couple of bits in the cell i are respectively the $(2i+1)^{th}$ and the $2i^{th}$ bit of the former (upper) addend m and are specified as m_{2i+1}, m_{2i}, the latter (lower) couple of bits in the cell i are respectively the $(2i+1)^{th}$ and the $2i^{th}$ bit of the latter addend n and are specified as n_{2i+1}, n_{2i}, all with positional weight 2^i (see Fig. 3).*
- $\tau : S^2 \rightarrow S$ *is the transition function so defined from the following equations, where two configurations c' and c'' are considered such that $c'' = \gamma(c')$:*

1. $m''_{2i} = 0,\ 0 \leq i < l;$
2. $m''_{2i+1} = m'_{2i},\ 0 \leq i < l;$
3. $n''_{2i} = (m'_{2i-1} \wedge n_{2i-1}') \vee (m'_{2i-1} \wedge n'_{2i-2}) \vee (n'_{2i-1} \wedge n'_{2i-2}),\ 0 < i < l;$
   $n''_0 = (m'_{2l-1} \wedge n'_{2l-1}) \vee (m'_{2l-1} \wedge n'_{2l-2}) \vee (n'_{2l-1} \wedge n'_{2l-2})$
4. $n''_{2i+1} = m'_{2i+1} \oplus n'_{2i+1} \oplus n'_{2i},\ 0 \leq i < l;$

m' and n' are respectively the former and latter addend of a configuration c', m'' and n'' are respectively the former and latter addend of c'' where n''_{2i} is the carry bit with positional weight 2^i of $m'_{2i-1} + n'_{2i-1} + n'_{2i-2}$; n''_{2i-1} is the 'lesser' bit with positional weight 2^i of $m'_{2i+1} + n'_{2i} + n'_{2i+1}$ (see Fig. 4).

The ring topology of ADDr (therefore a finite number of cells) involves that additions are performed properly, only if there is no overflow, i.e., significant length of numbers doesn't overcome $l-1$ bits, because the last cell is the neighbor to the first one; l may be large at will, so a sufficient length of cells may be always assumed (sufficient length condition).

Theorem 4. *Let c' be a generic configuration of ADD with length l and m', n' respectively the upper and lower addend of c'; let $c'' = \gamma(c')$ and m'' and n'' respectively the upper and lower addend of c'', then $m' + n' = m'' + n''$ (see Fig. 4).*

Proof. $m'_{2l-1} = 0$, $n'_{2l-1} = 0$, $m'_{2l-2} = 0$, $n'_{2l-21} = 0$ by the sufficient length condition. Therefore $m''_0 = 0$, $n''_{2l-1} = 0$, $m''_{2i} = 0$ for $0 \leq i < l$ by Eq. 1 defining τ:

m (37)	→	00	00	00	01	11	10	00	01
n (25)	→	00	00	00	00	11	01	11	10
Positional weight	→	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Fig. 3. An example of ADDr configuration c (highlighted) with $l = 8$; the upper sequence of bits is the former addend m, the lower sequence is the latter addend n; the positional weight of each cell is specified below, values of m and n in base 10 are on the left in brackets.

$$
\begin{aligned}
m' + n' &= \sum_{i=0}^{l-1} (m'_{2i+1} + m'_{2i} + n'_{2i+1} + n'_{2i}) 2^i \\
&= \sum_{i=0}^{l-2} (m'_{2i+1} + n'_{2i+1} + n'_{2i}) 2^i + (m'_{2l-1} + n'_{2l-1} + n'_{2l-2}) 2^{l-1} + \sum_{i=0}^{l-1} m'_{2i} 2^i \\
&= \sum_{i=0}^{l-2} (n''_{2i} 2^{i+1} + n''_{2i+1} 2^i) + (n''_0 2^0 + n''_{2l-1} 2^{l-1}) + \sum_{i=0}^{l-1} m''_{2i+1} 2^i + \sum_{i=0}^{l-1} m''_{2i} 2^i \\
&= \sum_{i=0}^{l-1} (m''_{2i+1} + m''_{2i} + n''_{2i+1} + n''_{2i}) 2^i = m'' + n''
\end{aligned}
$$

□

Theorem 5. *Let c be a generic configuration of ADDr with length l and m, n respectively the upper and lower addend of c; let $c' = \gamma(c)$ and $c'' = \gamma(c')$, m', n' and m'', n'', respectively the upper and lower addend of c' and c'', then m' is a cfα and $m'' = 0$.*

Proof. $m'_{2i} = 0, 0 \leq i < l$ by ADDr definition (Eq. 1), then m' is a cfα (e.g., steps 1 and 2 in Fig. 4); $m''_{2i+1} = m'_{2i} = 0$ by ADDr definition (Eq. 2) and $m''_{2i} = 0$ by ADDr definition (Eq. 1), $0 \leq i < l$; then $m'' = 0$ (e.g., steps 2 and 3 in Fig. 4).□

Corollary 3. *Let c be a generic configuration of ADDr with length l and m, n respectively the upper and lower addend of c; let $c' = \gamma(c)$ and $c'' = \gamma(c')$, m', n' and m'', n'', respectively the upper and lower addend of c' and c'', $m + n = m' + n' = m'' + n'' = n''$.*

Proof. $m + n = m' + n' = m'' + n''$ by Theorem 4, m' is a cfα by Theorem 5, $m'' = 0$ by Theorem 5. □

Therefore an addition in ADDr is exactly performed in two steps, the result is found in the latter addend, that is in RBN representation. If the former addend is in a cfα, such an addition is performed in one step (e.g., step 2 and 3 in Fig. 4).

Theorem 6. *Let c' be a configuration of ADDr of length l with m', n' respectively the upper and lower addend of c; let $c'' = \gamma(c')$, m'', n'', respectively the upper and lower addend of c'', if $m' = 0$, then $m'' = m' = 0$, $n'' = m' + n' = n'$. Furthermore, Eqs. 3 and 4 correspond to Eqs. 1 and 2 of the definition of ADD in Sect. 2.1.*

Proof. The configuration c' with the upper addend $m' = 0$ ($m'_{2i+1} = 0$, $m'_{2i} = 0$) evolves according to the following simplified equations:

1. $m''_{2i} = 0$
2. $m''_{2i+1} = m'_{2i} = 0$
3. $n''_{2i} = (m'_{2i-1} \wedge n'_{2i-1}) \vee (m'_{2i-1} \wedge n'_{2i-2}) \vee (n'_{2i-1} \wedge n'_{2i-2}) = (n'_{2i-1} \wedge n'_{2i-2})$
4. $n''_{2i+1} = m'_{2i+1} \oplus n'_{2i+1} \oplus n'_{2i} = n'_{2i+1} \oplus n'_{2i}$

□

Note that by Theorem 6, Eqs. 1 and 2 ensure that if the upper addend of a configuration in ADDr is 0, the upper addend of the following configurations are 0; furthermore Eqs. 3 and 4 are the same of Eqs. 1 and 2 of ADD. Therefore, if the bits of lower addend of ADDr in even (odd) position match the bits of upper (lower) addend in a configuration of ADD, then the ADDr configurations evolve in a cfα after a maximum steps of $l+1$ (the first two steps obtain that the upper addend is 0, the following ones that the lower addend is a cfα) according to Corollary 3, therefore the following corollary holds:

Corollary 4. *Let c be a generic configuration of ADDr of length l with m, n respectively the upper and lower addend of c; let $c' = \gamma^2(c)$, $c'' = \gamma^{l-1}(c')$, being m', n', respectively the upper and lower addend of c', then $c'' = \gamma^{l+1}(c)$ implies that $n'' = m + n$ and n'' is a cfα.*

Proof. $m' = 0$ by Theorem 6, then $m' = 0$, $n' = m + n$, therefore n'' is a cfα. □

Positional weight	→	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
m (37)		00	00	00	01	11	10	00	01
n (25)		00	00	00	00	11	01	11	10
	t=0								
m (25)		00	00	00	10	10	00	00	10
n (37)		00	00	00	01	11	01	00	10
	t=1								
m (0)		00	00	00	00	00	00	00	00
n (62)		00	00	01	01	10	10	01	00
	t=2								
m (0)		00	00	00	00	00	00	00	00
n (62)		00	00	10	10	10	10	10	00
	t=3								
m (0)		00	00	00	00	00	00	00	00
n (62)		00	00	10	10	10	10	10	00
	t=4								

Fig. 4. Evolution example of ADDr with length $l = 8$ for 4 steps (t). Configurations are highlighted, the upper sequence of bit is the former addend m, the lower one is the latter addend n, their values in base 10 are on the left in brackets; the cell positional weight is specified on top.

An addend in CBN can be translated in RBN as a cfα costless, just adding in parallel 0's at right of each bit, vice versa an addend in cfα of RBN can be translated in CBN costless, just eliminating in parallel the even 0's, an addition in ADD takes $l-1$ (l is the number of cells of ADD and ADDr) steps, while an addition in ADDr takes one step if the first addend is in cfα, but it takes $l-1$ steps if the result of a such addition has to be obtained in cfα. So working in ADDr is convenient only if ADDR is fed by $p > 2l$ upper addends, if the calculation involves a sequence of additions of natural numbers.

An extension of ADDr for the integers according to the two's complement representation could be developed; an intuitive way is breaking the ring between cells 0 and $l-1$ (cell $l-1$ assumes a positional weight of -2^{l-1}) and considering that the state of the -1 neighbor of cell 0 is always acquired as $\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}$.

4 Conclusions and Comments

The exemplary case of the addition operation on $\mathbb{N}$ within two CA ADD and ADDr with two different representations is here treated in order to investigate how CA can efficiently exploit their intrinsic parallelism. Natural numbers were considered in order that CA properties could emerge more clearly, even if a possible extension to $\mathbb{Z}$ (therefore to the elementary arithmetic) could be straightforward, but lengthy. A further investigation will be devoted to this problem.

The addition of two natural numbers can be surely operated by a CA in parallel way, but the carry problem in the usual numerical representation could make the parallel calculation a caricature of the sequential calculation, but, if we adopt an appropriate redundant numerical representation, then all the power of the parallelism discloses.

However to give an explicandum for a criterion of cost-effectiveness is not easy because situation is further complicated if a particular numeral representation is mandatory for the solution of a problem: e.g., sensors of automatic mobile systems could receive information only in a particular representation and utilize the elaborated solutions in that same representation. If a single operation (a single addition of two natural numbers in this case) is considered, there is no convenience in using a faster CA, because costs of translation from RBN to CBN annul any advantage, but not if a long sequence of consecutive additions is necessary.

This often elusive question has to move from the analysis of single operations over the whole of the operations, necessary for the problem solution. A notion of complexity that accounts for the relations operation/representation should possibly be investigated.

References

1. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)
2. Mitchell, M.: Computation in cellular automata: a selected review. In: Schuster, H.G., Gramms, T. (eds.) Nonconventional Computation, pp. 95–140. VCH Verlagsgesellschaft, Weinheim (1996)
3. Sheth, B., Nag, P., Hellwarth, R.W.: Binary addition on cellular automata. Complex Syst. **5**(5), 479–486 (1991)
4. Toffoli, T., Margolus, N.: Cellular Automata Machines: A New Environment for Modeling. MIT Press, Cambridge (1987)
5. Di Gregorio, S.: Una rappresentazione non univoca degli interi: basi per una nuova aritmetica degli elaboratori elettronici (in Italian). Quaderno del Progetto Nazionale Teoria degli Algoritmi, M.P.I., Università della Calabria (1984)
6. Clementi, A., De Biase, G.A., Massini, A.: Fast parallel arithmetic on cellular automata. Complex Syst. **8**(6), 435–442 (1994)
7. Clementi, A., De Biase, G.A., Massini, A.: Pipelined addition, accumulation and multiplication of binary numbers on cellular automata. Università della Sapienza, Roma (1995)
8. Choudhury, P.P., Sahoo, S., Chakraborty, M.: Implementation of basic arithmetic operations using cellular automaton. In: IEEE International Conference on Information Technology, ICIT 2008, pp. 79–80 (2008)
9. Vourkas I., Sirakoulis G.: FPGA based cellular automata for environmental modeling. In: 19th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2012, pp. 93–96 (2012)
10. Di Gregorio, S., Trautteur, G.: On reversibility in cellular automata. J. Comput. Syst. Sci. **11**, 382–391 (1975)