Universality of Evolved Cellular Automata in-Materio

STEFANO NICHELE[1,2]*, SIGVE SEBASTIAN FARSTAD[1], GUNNAR TUFTE[1]

[1] *Department of Computer and Information Science, Norwegian University of Science and Technology, Norway*
[2] *Department of Computer Science, Oslo and Akershus University College of Applied Sciences, Norway*

Received xxxxx; In final form xxxxx

Evolution-in-Materio (EIM) is a method of using artificial evolution to exploit physical properties of materials for computation. It has previously been successfully used to evolve a multitude of different computational devices implemented in physical materials. One of the biggest problems in exploiting materials is finding a good computational abstraction to carry computation on top of the underlying physical process. This paper presents elementary cellular automata (CA) as a possible abstraction and presents successfully evolved CA transition functions in single-walled carbon nanotube (SWCNT) and polymer composite materials. Such implementation allows reasoning about the computational capabilities of materials and draw analogies with cellular automata complexity and computation at the Edge of Chaos. This work is done within the European Project NASCENCE.

Key words: Evolution-in-Materio, Cellular Automata, Edge of Chaos, Single-Walled Carbon Nanotubes, Complexity, Computational Materials.

* email: stefano.nichele@hioa.no

1 INTRODUCTION

The natural evolutionary process, in stark contrast to the traditional human top-down building-block-oriented engineering approach, is not one of abstraction, componentization and careful manipulation based on an understanding of underlying mechanics. Rather, it is a "blind-yet-guided" meta-heuristic approach able to exploit properties to create "designs" without needing to understand them or the underlying mechanics that power them. Evolution-in-Materio (EIM) [22] is an attempt to mimic this natural evolution process in order to exploit new, interesting and not-fully-understood materials for different use cases, most notably for computational purposes.

In this paper, the idea of exploiting a single-walled carbon nanotube and polymer composite mesh material as stable medium for computation of cellular automata (CA) transition functions is explored. An Evolution-in-Materio platform called Mecobo [18], a product of the EU-funded research project NASCENCE [3], is used to facilitate evolution experiments. Results in this paper show the successful evolution of elementary CA in SWCNTs. CA provide a new physical abstraction for computation in unconventional materials. The way computation is performed at the physical level is based on local interactions amongst neighboring molecules without a central controller. It may be possible to abstract such a process as a "cellular-automaton-in-materio" and exploit materials to perform computation as a CA, e.g. evolve cellular automata transition tables of different complexity, or even embody a cellular automaton within the material, e.g. exploiting the underlying intrinsic CA-in-materio to execute on a given external input. Evolution of CA-in-materio is placed in the context of computation at the Edge of Chaos [16], and different evolved transition functions are analyzed within the complexity framework provided by Langton's Lambda parameter [16] and Wolfram CA classes [34]. All 256 Elementary Cellular Automata are investigated. The hypothesis is that different cellular automata should be easier or more difficult to evolve based on how computationally complex they are [2], and that the ease of evolution may reveal details about the computational capacity of the material-under-study.

The article is laid out as follows: Section II provides background information and Section III describes the experimental setup. In Section IV the results for the evolution of selected CA transition functions are presented and Section V gives results for the evolution of all elementary CA in-materio. Discussion and analysis are presented in Section VI, together with directions for future work.

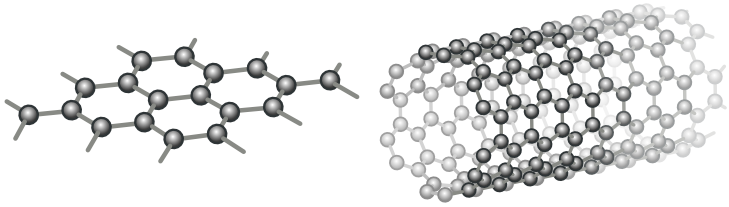
2 BACKGROUND

2.1 Evolution-in-Materio

Evolution-in-Materio is the exploitation of emergent computational behavior in physical materials through artificial evolution. The idea is that some physical processes inherent in different materials may be interpreted as useful computation. EIM attempts to harness this computational power using artificial evolution in order to discover favorable material configurations that may be exploited. One of the earliest attempts at manipulating a physical material for computation was conducted by cybernetician Gordon Pask in 1958. He attempted to create a physical signal processing device based on configurations of grown dendritic iron wires in a ferrous sulphate solution [29]. Modern Evolution-in-Materio was started in 1996 when Adrian Thompson demonstrated that unconstrained evolution in a physically implemented logical system was able to exploit physical properties outside of the logical domain to improve fitness, and hence perform computation [32]. In his experiments, Thompson tried to use artificial evolution to configure a field-programmable gate array (FPGA) to perform tone discrimination. Upon inspection of solutions, it became clear that the computation was not entirely performed in the discrete logical circuit. Instead, the computation relied on physical properties of the FPGA chip itself, outside of the discrete logical domain. This result shows that using evolution to design physically-implemented computers allows for a much larger design space than what is possible with traditional constructive engineering. This, in turn, opens the door to creating more efficient computational devices that to a greater extent exploit natural physical behaviors of the underlying computational substrate.

2.2 Current Materials

One challenge in the field of Evolution-in-Materio is discovering which materials are suitable for use as a computational substrate. A good material should preferably exhibit a number of properties that both enable computation and configuration so that evolution can be reliably performed. These properties include having a complex, practically (read: electronically) configurable semi-conducting structure in such a way that the material responds near-instantly and consistently to different inputs, as well as being robust to changes in the external environment such as lighting conditions, temperature and electromagnetic fields [1, 31]. It also helps if the material is easily available. Some interesting candidate substrates that are currently under research are single-walled carbon nanotubes, liquid crystals matrices, silicon FPGA chips, and slime mould [13].



(a) Graphene **(b)** Carbon nanotube

FIGURE 1: An atom-scale visualization of a single-walled carbon nanotube. Each sphere represents a carbon atom, and the cylinders between them represent atomic bindings. The sheet to the left is graphene, and the cylinder to the right is graphene wrapped to form a carbon nanotube. Illustrations adapted with permission from original work by Jozef Sivek, distributed under a CC BY-SA 4.0 license.

2.3 Liquid Crystal Matrices

Liquid crystals are matter that can exist in a mesomorphic state, carrying properties similar to both conventional liquids and solid crystal. For instance, the matter may behave like a liquid in terms of flow, all the while having molecules that are arranged in a distinct crystal-like fashion. Liquid crystals are relatively stable materials, but change their characteristics when subjected to electric fields. This makes them interesting subjects for Evolution-in-Materio, as it means that they can be configured to assume certain behaviors electronically. Matrices of liquid crystals are today already mass-produced on a gigantic scale, because they are a core technology used in modern computer displays. This makes the material cheap and readily available. Liquid Crystal Matrices have successfully been used as a substrate for computation in several experiments [10, 11, 12, 14].

2.4 Single-Walled Carbon Nanotubes

Single-walled carbon nanotubes (SWCNT) are cylindrical carbon allotropes with unusual physical properties. They exhibit extraordinary electrical properties, which is interesting from an Evolution-in-Materio standpoint. Single-walled carbon nanotubes are engineered by wrapping a one-atom-thick sheet of graphene into a tube. The ”angle” at which the nanotube is wrapped affects the electrical properties of the nanotube - some SWCNTs’ electrical conductivity show metallic conducting behavior, whilst others show different levels of semiconducting behavior. An illustration of a SWCNT can be seen in FIGURE 1. One way of using SWCNTs for computation is by arbitrar-

ily arranging many SWCNTs in a random network and treating it as a single computational device [30]. The advantage of this is that it enables SWCNT mesh device production in large scale at the wafer level [9]. Single-walled carbon nanotube mesh devices have been successfully used as a substrate for computation in several experiments [5, 15, 24, 25, 26]. The material-under-study in the experiments presented in this article is a random single-walled carbon nanotube and polymer mesh material.

2.5 Evolution-in-Materio Platform

The EU project NASCENCE [1, 3], or NAnoSCale Engineering for Novel Computation using Evolution, aims at "modeling, understanding and exploiting the behavior of evolving nanosystems (e.g. networks of nanoparticles, carbon nanotubes) with the long-term goal to build information processing devices exploiting these architectures without reproducing individual components". One of the products that have emerged from the NASCENCE project is the Mecobo platform. It is a hardware and software platform for Evolution-in-Materio developed by Lykkebø et al. [18]. It is designed to interface with a large variety of materials, allowing Evolution-in-Materio fitness evaluation directly on a physical substrate. The Mecobo platform is used for the experiments herein, with a similar approach as in [17, 27, 28]. For a schematic overview of EIM plantform see FIGURE 2.

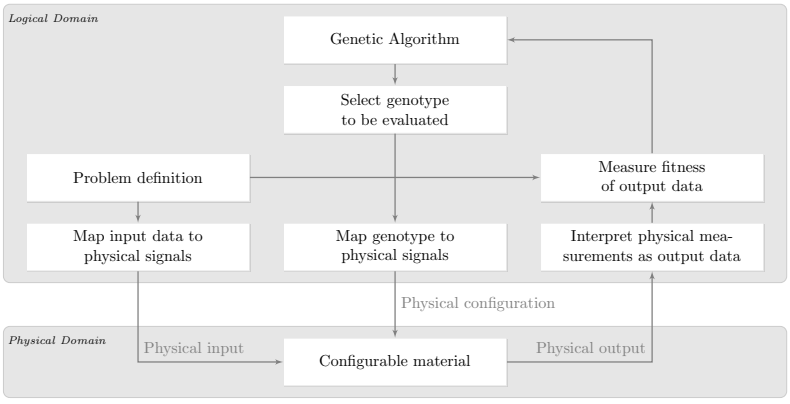


FIGURE 2: An overview of Evolution-in-Materio – artificial evolution is simulated in the logical domain, typically on a traditional computer, and fitness is evaluated by performing computations in the physical domain.

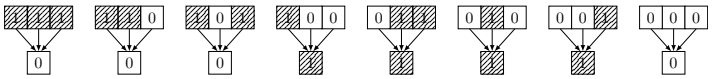


FIGURE 3: An Elementary Cellular Automata state transition table laid out to illustrate the Wolfram Code numbering scheme. The top row of groups of three and three cells are the possible neighborhood states for a cell at time step $t-1$, and the cell beneath each group is the resulting state for that same cell at time step t. Reading the bottom row as a binary number ($00011110_2 = 30$) reveals the name of the automaton: Rule 30.

2.6 Cellular Automata

Cellular automata are abstract discrete n-dimensional dynamical systems that evolve over time. They consist of a graph of locally-connected nodes that each take on one of k discrete states in time step t. The state of a node n in the graph at time step t is given by the states of n's neighboring nodes at time step $t-1$. Each cellular automaton has a rule table describing how to transition from time step to time step.

In the simplest case, a cellular automaton takes on the form of a one-dimensional array of binary cells where each cell has three neighbors: the cell immediately to the left, the cell immediately to the right, and itself, as illustrated in FIGURE 4. These specific cellular automata are called Elementary Cellular Automata [33]. There are 256 possible rule table permutations for the Elementary Cellular Automata. Of these 256 automata, 88 are fundamentally inequivalent [34, p.57].

The Elementary Cellular Automata are given a numbering scheme known as the Wolfram Code in [33] that is rooted in binary number representation. Each of the $2^3 = 8$ possible neighborhood states for a given Elementary Cellular Automaton E are represented as each their binary number and ordered numerically. The resulting states for the next time step given from each of these neighborhood states are then taken in order as bits of a new binary

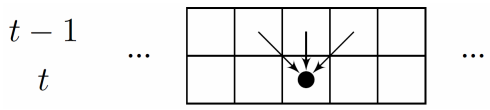
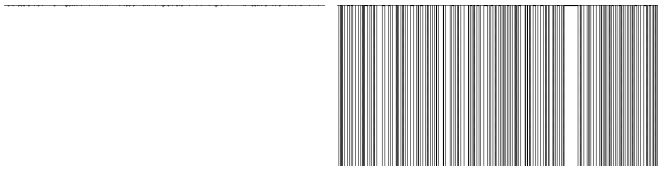
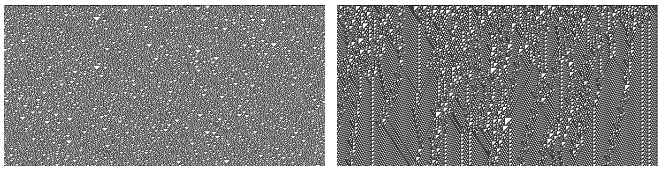


FIGURE 4: An example state-time representation of an Elementary Cellular Automaton. The state of a cell is decided by the state of its two neighboring cells and itself in the previous time step.



(a) Rule 32, a Class I cellular automaton.

(b) Rule 108, a Class II cellular automaton.



(c) Rule 30, a Class III cellular automaton.

(d) Rule 110, a Class IV cellular automaton.

FIGURE 5: Example Elementary Cellular Automata with random initial states.

number r. This number r is the number identifying E. As an example, Elementary Cellular Automaton Rule 30's rule table and corresponding Wolfram Code number is illustrated in FIGURE 3.

2.7 Wolfram CA Classes

Wolfram classified cellular automata into four different classes based on the behaviors they seem to exhibit [34]. However, the class definitions are not strict in the mathematical sense.

Class I are CA that tend to a stable homogeneous state. Randomness in the initial state tends to disappear as time progresses. Rule 32, which is shown in FIGURE 5a, is an example of a Class I cellular automaton.

Class II are CA that yield a sequence of simple stable or periodic structures. Randomness in the initial state is somewhat retained in periodic structures. Changes made to the initial state tend to only have a local impact on the behavior of the cellular automata over time. Rule 108, which is shown in FIGURE 5b, is an example of a Class II cellular automaton.

Class III are CA that exhibit chaotic aperiodic behavior. Changes made to the initial state tend to have a global impact on the behavior of the cellular automata over time. Rule 30, which is shown in FIGURE 5c, is an example of a Class III cellular automaton.

Class IV are CA that yield complicated localized structures, some propa-

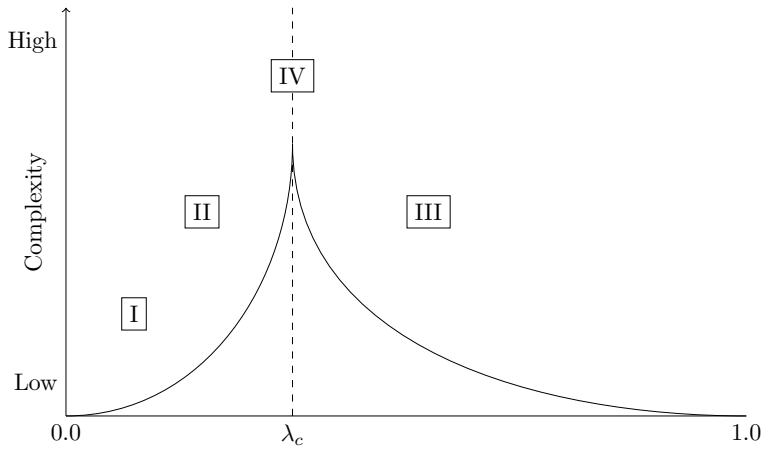


FIGURE 6: Location of Wolfram Classes in λ space, recreated from [16, Fig. 16].

gating. Changes made to the initial state tend to have a global impact on the behavior of the cellular automata over time. Wolfram postulates that most Class IV cellular automata are capable of general computation [34]. Examples of Class IV cellular automata include Rule 110 (shown in FIGURE 5d) and Conway's Game of Life [34].

2.8 λ-Parametrization and the Edge of Chaos

A different approach to cellular automata classification is through λ-parametrization [16]. The λ-parameter is a measure of what percentage of transitions in a cellular automaton's rule table go to an arbitrarily selected quiescent state. It is defined as

$$\lambda = \frac{K^N - n}{K^N} \tag{1}$$

given by (1), where K is is the number of different states a cell can have, N is the neighborhood size, and n is the number of state transitions that go the the quiescent state. For example, in Rule 30 of the Elementary Cellular Automata, taking state 0 as the quiescent state, 6 of 8 transitions go to the quiescent state. Since ECA are binary cellular automata with a neighborhood size of 3, this means that

$$\lambda_{\text{Rule 30}} = \frac{2^3 - 6}{2^3} = 0.25 \tag{2}$$

of Rule 30 is as given by (2).

The idea is that cellular automata with similar λ-values tend to exhibit similar behaviors. This means that, as an example, Rule 129 of the elementary cellular automata with its $\lambda_{\text{Rule 129}} = 0.25$ should behave more similarly to Rule 30 than a Rule with a different λ, like Rule 85, which has a λ of $\lambda_{\text{Rule 85}} = 0.5$.

Langton examined a set of different 1-dimensional 4-state cellular automata with a neighborhood size of 5, and gave a qualitative classification on the behavior of the cellular automata in relation to their λ-values [16]. Langton observed that low λ-values tended to give cellular automata with a high amount of order, reaching a steady (potentially periodic) state quickly, and high λ-values tended to give cellular automata exhibiting chaotic behavior. Langton argues that the cellular automata most suitable for computation will be found at the boundary between these two extremes in behavior, at the "Edge-of-Chaos". In Fig. 16 of [16], recreated here as FIGURE 6, Langton illustrates the relationship between the computational complexity in cellular automata with regards to the λ-parameter and the possible locations in λ space of cellular automata of different Wolfram Classes. The λ-parametrization's significance in identifying computationally interesting cellular automata at the "Edge-of-Chaos" was later disputed by Mitchell et al., who suggested that the original findings are not properly reproducible [23].

2.9 Cellular Automata as a Physical Abstraction

One of the advantages of computation in-materio is that it offers the possibility of performing computation "directly" in the material, as opposed to in some abstracted computational model implemented in a material. The latter will necessarily discard a large part of the computational power of the substrate as a consequence of the abstraction. It seems, then, that exploiting direct in-materio computation is favorable in terms of computation power. However, direct computation in-materio can be quite difficult, especially if universality and scalability is desired. There is an apparent trade-off between efficient usage of the computational complexity in the substrate and ease of programmability for practically useful results which seems to be related to the intuition that computational potential is lost in the abstraction from substrate to theoretical model. The larger the disaffinity between the abstract model and the physical processes in the material becomes, the larger the inefficiency in translation will grow. Thus, finding an abstract computational model that closely matches the physical properties of a material might minimize the computational gap between the physical and the abstract [31].

Cellular automata is a promising abstract computational model for this purpose. Just like they seem to be in physical materials, the computational processes in cellular automata are massively parallel in a distributed and localized fashion. This is a completely different paradigm than the centralized model found in conventional computing.

3 EVOLUTION-IN-MATERIO EXPERIMENTAL SETUP

In this section, the used Evolution-in-Materio experimental setup is presented. The goal is to evolve Elementary Cellular Automata transition functions in the material, to test the central hypothesis that computing with cellular automata in a single-walled carbon nanotube and polymer composite material is an interesting avenue for further research. The computational capacity of the material is investigated by doing an evolutionary sweep of all the 256 Elementary Cellular Automata in the material, enabling existing knowledge about elementary CA complexity to be used in gauging the computational capacity of the material under study. In [8] a preliminary study of CA-in-materio evolution is presented and in [4] an overview of the EIM setup is described.

The experiments herein model the material as a mapping function capable of mapping a binary input to a binary output. The material is assumed to be capable of performing some computational function on a set of inputs coded as voltage patterns applied to one or more material electrodes, allowing the output to be read as a voltage pattern from one or more electrodes.

The flow of an experiment is as follows: first, an output calibration over the material is performed for each problem specification. This is done by sweeping through all possible inputs, or at least a somewhat uniformly distributed randomly selected subset of inputs, in order to determine the distribution of possible readable outputs, as to be able to make a meaningful interpretation of the output. This is necessary because the range of voltages readable on output electrodes to be interpreted as computational output can vary to a great extent based on the way input is coded.

Once an experiment has been calibrated, a Genetic Algorithm is run to search for configurations that exhibit the desired computational behavior.

Computational stability of a function implemented in-materio is an important concern. Running the same calculation multiple times should result in a correct output each time – having a computational function only return the correct result a fraction of the time is considerably less useful than one which is consistently correct. Therefore, the computational stability of the

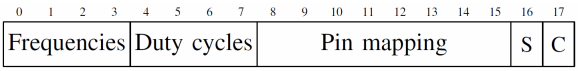


FIGURE 7: The genotype mapping for the cellular automata experiment. S is the static amplitude of a logical low input signal and C is the center threshold offset.

evolved configuration is finally tested by repeatedly performing computation in-materio and verifying the correctness of the output.

3.1 Genetic Algorithm Overview

The genetic algorithm used for the experiments has a population of 40 individuals with generational mixing. Parents are selected by tournament selection where 8 randomly extracted individuals are chosen from the population and, with probability 0.95 the fittest is selected, or else a random individual is chosen. Crossover combines two parents by copying with probability 0.5 each symbol from either the first or second parent. Mutation changes each single gene individually with probability p = 0.1.

The evolvable genotype is represented as a symbol vector of length 18, as shown in FIGURE 7, where the first 4 symbols represent material configuration signals in a form of pulse wave frequencies and the next 4 symbols represent the duty cycles. The next 8 symbols represent the pin mapping. Then follows a symbol representing the static amplitude of logical low coding signal. The last symbol represents an output interpretation threshold offset.

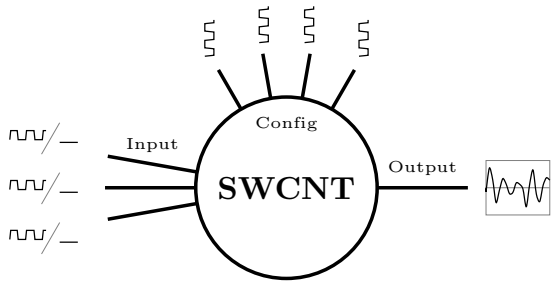


FIGURE 8: Input, output and configuration mapping of the material interface electrodes for Elementary Cellular Automata computation.

Which specific physical electrodes are used for which of the input and output signals is decided by a logical-to-physical mapping of electrodes coded

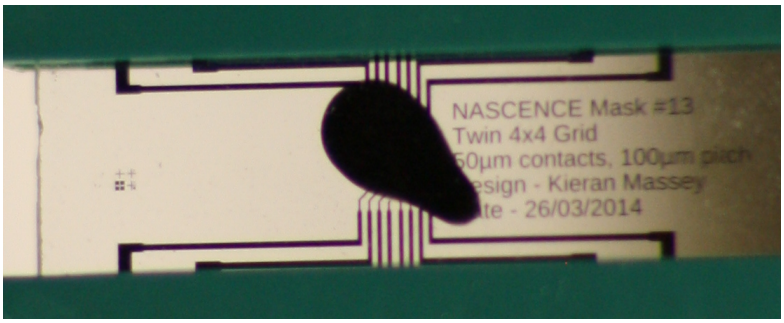


FIGURE 9: A photograph of the SWCNT material on its glass slide.

in the evolvable genotype. A schematic view of the SWCNT/polymer composite material with its logical input and output electrodes can be seen in FIGURE 8.

The fitness is a function of how well it can compute the target cellular automaton's state transition function on all the 8 possible input combinations. The squared scaled confidence value

$$c_n = \frac{(o_n - T)^2}{5^2} \tag{3}$$

for input combination n is then calculated for each input combination as defined in (3), where T is the output threshold value, and o_n is the averaged output value for input combination n. For each set of inputs, the interpreted output is compared to the expected output from the input-to-output mapping function. If the output matches, c_n points are awarded to the total fitness. If the output is undecided, 0 points are awarded to the total fitness. If the output otherwise does not match, $-10c_n$ points are awarded to the total fitness. Then, if all outputs were correct, an additional 8 points are awarded to the total fitness. Finally, the fitness is transposed by an additional 8 points, and then scaled by a factor of 0.0625 in order to obtain values in the range between 0 and 1+, where a score of 1 (or better) signifies a fully correct evolved solution. The fitness values did not directly account for stability, e.g. by performing the same fitness evaluation multiple times and averaging the result.

3.2 Material Overview

The computational substrate material used in the experiments is a random mesh of single-walled carbon nanotubes mixed with poly(butyl methacrylate)

(PBMA) and dissolved in anisole (methoxybenzene). This is laid out in a glass slide with 16 gold electrodes arranged in a 4×4 grid with contacts of 50 µm in diameter and 100 µm pitch between contacts. The material, slide #1 of batch #15, numbered B15S01, was prepared by Kieran Massey at Durham University by the following method: "20 µl of material are dispensed onto the electrode area; This is dried at 85°C for 30 min to leave a *thick film*; The hotplate is turned off and the substrates are allowed to cool slowly over a period of roughly 2 h to room temperature." After this process, the CNTs are not movable. The carbon nanotube concentration in the material is 0.75% by weight. All electrodes show connection resistances on the order of 20 kOhm, but it is reasonable to assume that the nanotube coverage over the electrodes is noticeably uneven, given the nanotube concentration level [21]. FIGURE 9 shows a photograph of the material on its glass slide. The material is produced within the NASCENCE project [1].

4 EVOLVING ELEMENTARY CELLULAR AUTOMATA IN-MATERIO

This experiment attempts to evolve selected Elementary Cellular Automata transition functions in-materio, to test the hypothesis that computing with cellular automata in a single-walled carbon nanotube and polymer composite material is an interesting avenue for further research. Of particular interest are CA in Class III and Class IV of Wolfram's classification, the former because they can be capable of performing complex computations [19], and the latter because they are conjectured to be capable of universal computation [34]. Two stable Elementary Cellular Automata have been evolved: Rule 151 (Class III), and Rule 54 (Class IV), conjectured but not yet proven to be computationally universal [34, p. 697][20]. An evolution of Rule 110 (Class IV), for which a proof of universality of computation was formulated by Cook in [6], was attempted, but no suitable solution was found.

The computational function can be thought of as a mapping from binary inputs to a binary output. The expected input-to-output mappings for Rule 54, Rule 151 and Rule 110, are given in TABLE 1, where I_n is input n, and $O_{\text{Rule } n}$ is the output for Rule n. The values are given in the logical domain.

Seven electrodes were used as input electrodes, three of which represent the state of the three cells in the neighborhood set for an elementary cellular automaton rule transition. Each of these three inputs can either be a logical 0 or a logical 1, coded as a static voltage or a digital pulse applied to the input electrodes, respectively. A logical 0 on an input electrode is realized by applying a selected constant static voltage, the exact value of which is decided

TABLE 1: Input-to-Output-Mapping Functions for the Selected Elementary Cellular Automata

I_2	I_1	I_0	$O_{\text{Rule 54}}$	$O_{\text{Rule 151}}$	$O_{\text{Rule 110}}$
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	0	1	0

on a per-solution basis. A logical 1 on an input electrode is represented by applying a digital square wave with a frequency of 10kHz and a $3.3\,\text{V}$ amplitude, from $0\,\text{V}$ to $3.3\,\text{V}$. The other four input electrodes function as a material configuration, each electrode being applied upon a digital pulse signal chosen from a specific range of frequency and duty cycle combinations (as specified in the evolved genotype).

The computational output expected from an elementary CA neighborhood is a single Boolean value: 0 or 1. A single electrode output electrode was used. The voltage output was sampled at 0.5MHz for 80 ms from the 10^{th} to the 90^{th} ms of computation. The samples were averaged arithmetically and compared to a predetermined threshold value. If the average was higher than the threshold value plus some small empirically determined padding to reduce measurement noise, the output was interpreted as a logical 1. Conversely, if the average was lower or equal to the padded threshold value, the Boolean output was interpreted as a logical 0. The threshold value was obtained experimentally by running a calibration sweep of the material, which consisted of performing 200 computations with randomly generated inputs conforming to the defined input coding. The averaged output was calculated for each computation, and the median of these averages were taken to be the threshold value.

4.1 Results

Computationally stable Rule 54 and Rule 151 transition functions have been found. No suitable solution was found for Rule 110. The evolved solution

configurations for Rule 54 and Rule 151 are shown in FIGURE 10.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Rule 54	160	217	183	55	99	200	160	12	115	180	166	205	30	89	175	119	243	138
Rule 151	149	220	16	174	123	119	75	157	28	51	163	154	138	115	125	36	219	2

FIGURE 10: Genotypes representing successfully evolved Rule 54 and Rule 151 state transition functions.

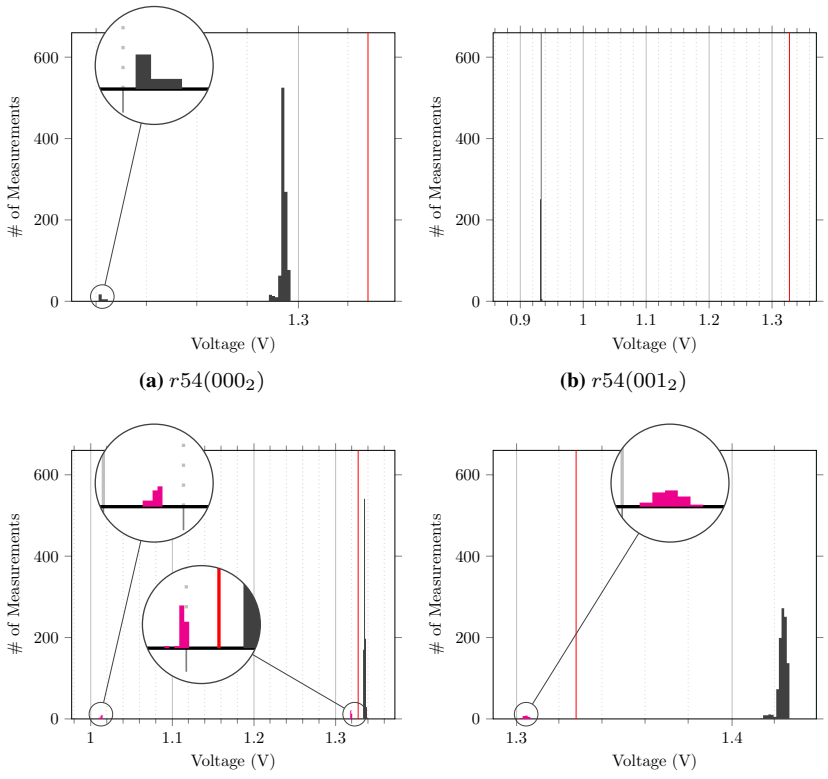
Stability

Both the evolved solutions in FIGURE 10 were tested for correctness by performing repeated computations. Each rule was tested for 10^3 repeated calculations of each of the eight possible inputs, for a total of $8 \cdot 10^3$ tests.

The evolved Rule 54 computed the correct value 7917 out of $8 \cdot 10^3$ times, resulting in an estimated failure rate of $1.0375\% \pm 0.2221$ *pp* at a 95% level of confidence. The evolved Rule 151 computed the correct value $8 \cdot 10^3$ out of $8 \cdot 10^3$ times. FIGURE 11 and FIGURE 12 show the levels of each of the measured average outputs for each of the $8 \cdot 10^3$ calculations grouped by input, for the evolved Rule 54 and the evolved Rule 151, respectively.

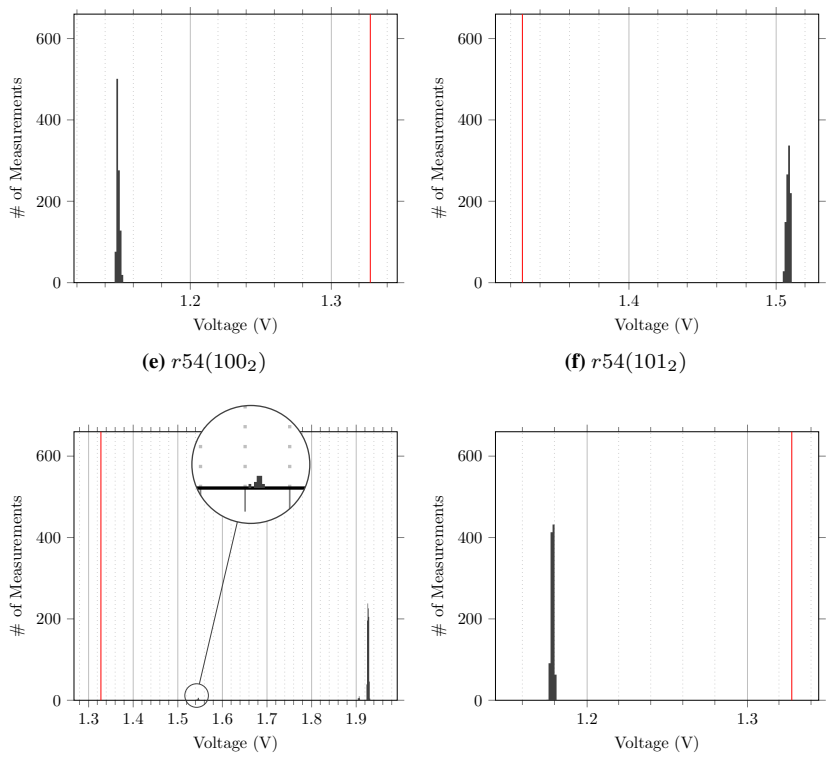
Example Computations

Here, a set of recorded Rule 54 and Rule 151 computations are presented. The material configurations are encoded in the genotypes in FIGURE 10. FIGURE 13 and FIGURE 14 show the states of the (logically mapped) input and output electrodes over the course of the execution of the computation of the Rule 54 and Rule 151 transition functions for the neighborhood state 011_2. FIGURE 15 presents samples recorded from eight Rule 54 calculations and FIGURE 16 presents samples recorded from eight Rule 151 executions. The entire recording for each calculation is in reality $4 \cdot 10^4$ samples long, and for practical reasons only the first 500 samples are plotted here. The horizontal black lines represent the measured total average of the entire recording for each calculation. The magenta threshold together with a small gray padding band represents the threshold value above or under which a recorded average is taken to be a logical 1 or 0, respectively.



(a) $r54(000_2)$ **(b)** $r54(001_2)$

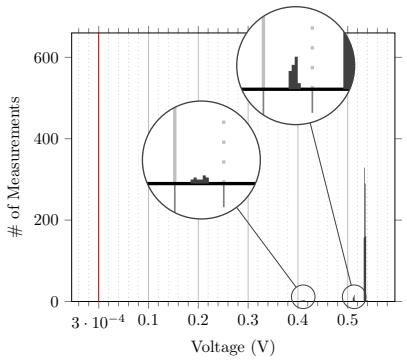
(c) $r54(010_2)$ **(d)** $r54(011_2)$



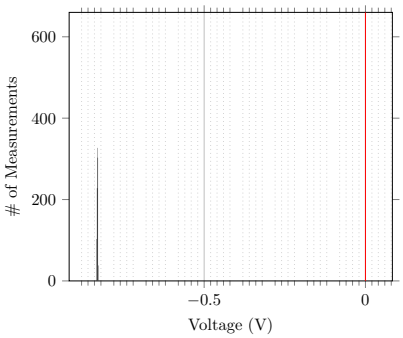
(e) $r54(100_2)$ **(f)** $r54(101_2)$

(g) $r54(110_2)$ **(h)** $r54(111_2)$

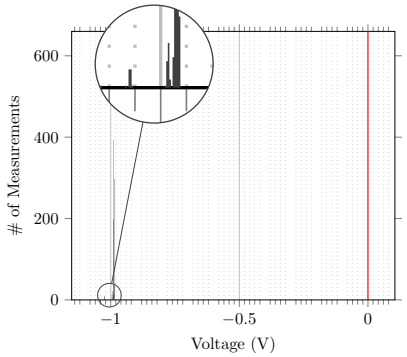
FIGURE 11: Histograms representing the measured average outputs over $8 \cdot 10^3$ calculations using the evolved Rule 54 configuration, grouped by input. The output threshold used for mapping the measured output into the binary logical domain is plotted as a red vertical line in each histogram. Measurements that result in a wrong computational answer are in magenta. Interesting details are magnified.



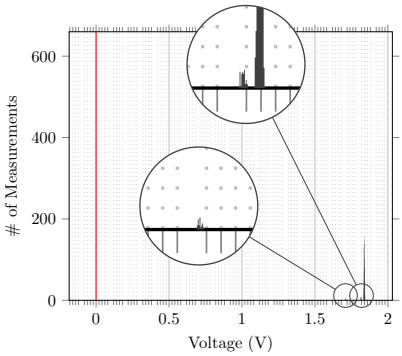
(a) $r151(000_2)$



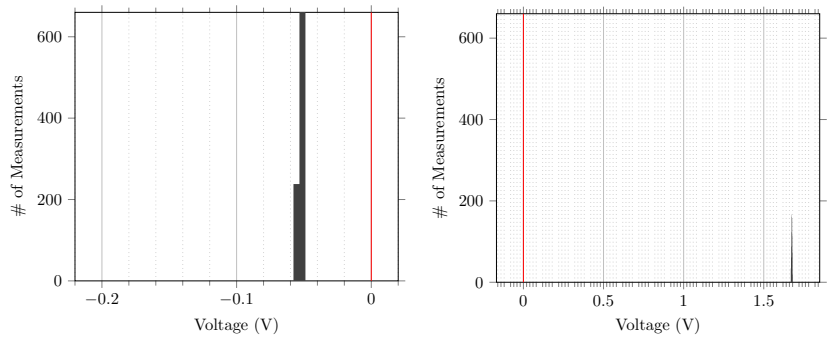
(b) $r151(001_2)$



(c) $r151(010_2)$

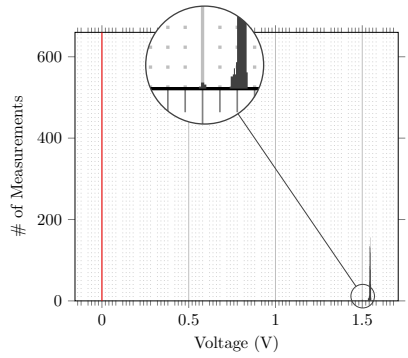


(d) $r151(011_2)$

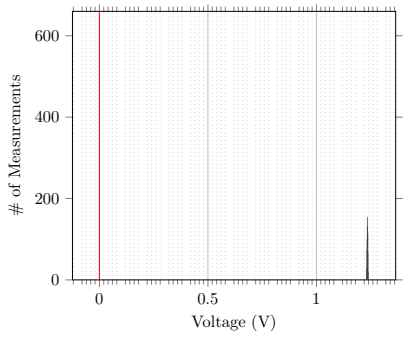


(e) $r151(100_2)$

(f) $r151(101_2)$



(g) $r151(110_2)$



(h) $r151(111_2)$

FIGURE 12: Histograms representing the measured average outputs over $8 \cdot 10^3$ calculations using the evolved Rule 151 configuration, grouped by input. The output threshold used for mapping the measured output into the binary logical domain is plotted as a red vertical line in each histogram. Interesting details are magnified.

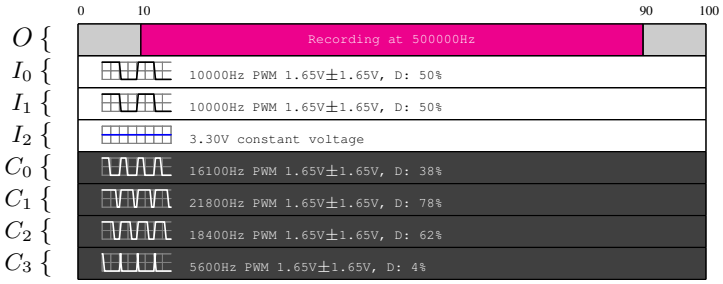


FIGURE 13: A timing overview of the activity on each logical electrode over time during a single computation of r54(011_2) on the evolved Rule 54 device. Time progresses along the x-axis, labeled in milliseconds at points of interest. O is the output pin, I_n are the input pins, and C_n are the configuration pins. The small waveform illustrations reflect the duty cycle, but not the frequency. D is the duty cycle.

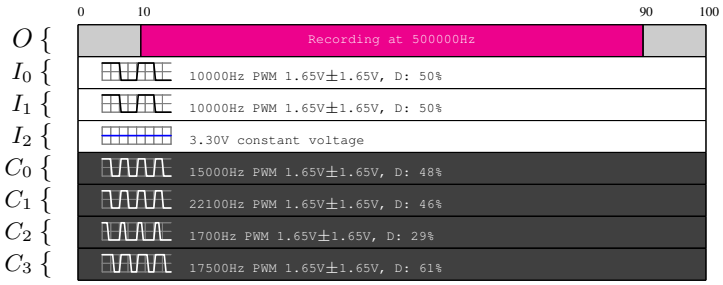


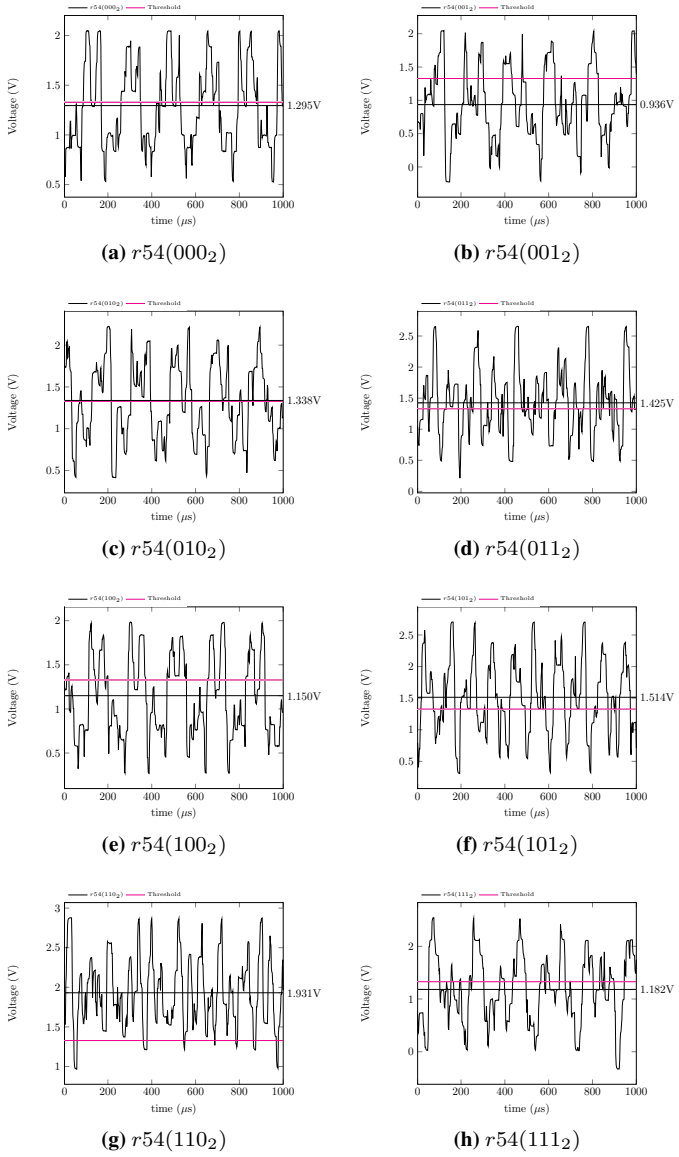
FIGURE 14: A timing overview of the activity on each logical electrode over time during a single computation of r151(011_2) on the evolved Rule 151 device. Consult the FIGURE 13 caption for a legend.

5 ALL ELEMENTARY CELLULAR AUTOMATA IN-MATERIO

This experiment attempts to evolve all the 256 different Elementary Cellular Automata transition functions in-materio, one by one. This is done in an attempt to measure the complexity ceiling of the material by using evolvability of the different rules as a proxy indicator.

The experimental setup is similar to that of the elementary cellular automata transition function experiments detailed in previous section.

The only parameters for the genetic algorithm that were altered are the population size, which is reduced to 20 individuals, and the recording time operating on samples recorded for 8ms, from the 1st to the 9th ms of a total electrode activation time of 10ms. The maximum number of generations has



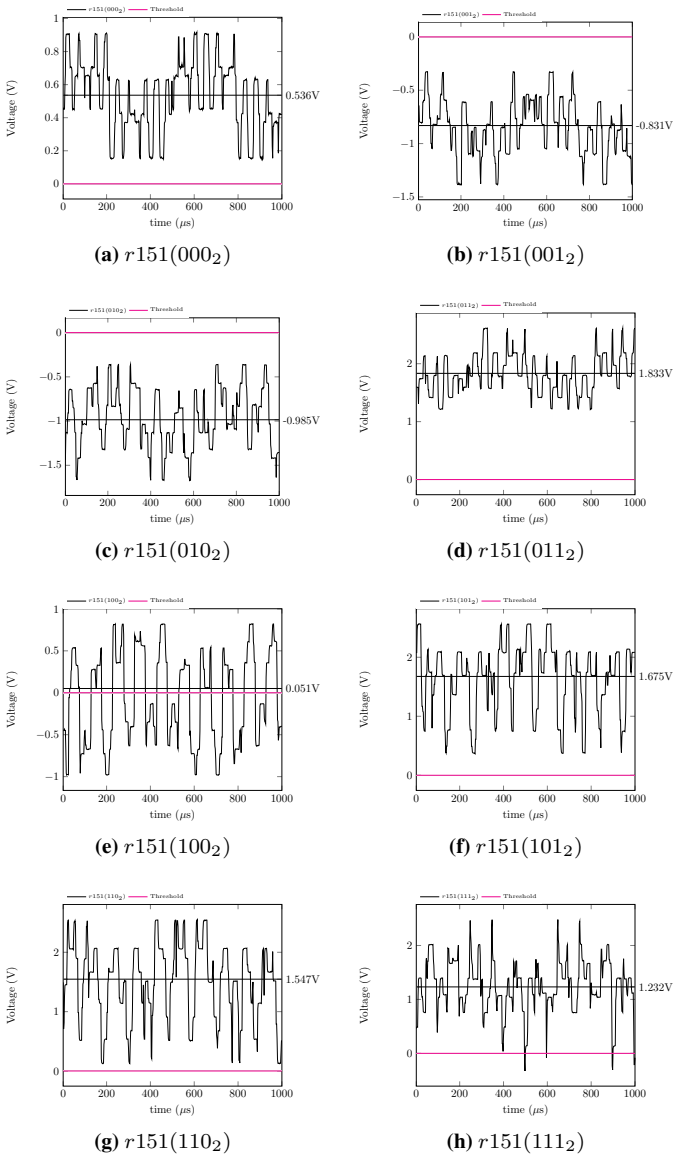
(a) $r54(000_2)$ **(b)** $r54(001_2)$

(c) $r54(010_2)$ **(d)** $r54(011_2)$

(e) $r54(100_2)$ **(f)** $r54(101_2)$

(g) $r54(110_2)$ **(h)** $r54(111_2)$

FIGURE 15: The first 500 raw material output samples recorded during eight Rule 54 computations. The black horizontal lines represent average values for a sample series, and the red horizontal line is the output threshold, which is 1.328V.



(a) $r151(000_2)$ **(b)** $r151(001_2)$

(c) $r151(010_2)$ **(d)** $r151(011_2)$

(e) $r151(100_2)$ **(f)** $r151(101_2)$

(g) $r151(110_2)$ **(h)** $r151(111_2)$

FIGURE 16: The first 500 raw material output samples recorded during eight Rule 151 computations. The black horizontal lines represent average values for a sample series, and the red horizontal line is the output threshold, which is 0.000V.

also been reduced to 100. Such parameters have been tuned with the goal of reducing the amount of time needed to run each experiment, as to carry out experiments in a reasonable amount of time. As such, the overall success rate may be decreased and some of the previously found CA rules may not be found by this specific experiment.

5.1 Results

Of the 256 rule evolution attempts, 42 rules were successfully found. The average number of generations before a successful rule was evolved was 23.57. TABLE 2 illustrates the results, showing one successfully evolved rule in each class. Due to space constrains, the full table for all 256 evolutionary runs is available in [7] and at the following link: `http://www.nichele.eu/files/nichele_ijuc2016_appendix.pdf`.

TABLE 2: Example results for four of the 256 evolutionary runs. The shown rules are 64 (class 1), 77 (class 2), 137 (class 4), and 151 (class 3). Grey rows signify that the rule in that row was successfully evolved. *Binary* shows a binary representation of the transition table. *Gen.* is the number of generations for each run. *Evo.* shows a small fitness graph for each run illustrating the best fitness for each generation along the x-axis.

Rule	Binary	λ	λ'	Class	Gen.	Evo.	Best fitness
Rule_{64}		0.25	0.25	Class I	88		1.0620
Rule_{77}		0.625	0.375	Class II	56		1.0044
Rule_{137}		0.375	0.375	Class IV	100		0.5256
Rule_{151}		0.625	0.375	Class III	21		1.0389

6 ANALYSIS

The single-walled carbon nanotube mesh and polymer composite material used in these experiments has shown itself capable of performing complex non-linearly separable computation tasks. Using the provided results, in the following sessions we reason about the computational properties of the underlying material itself.

6.1 Evolvability and the λ-Parameter

The Lambda parameter is one of many different proposed schemes of classification of cellular automata. Cellular automata with a λ-parameter close to 0 tend toward a frozen, non-changing structure over time, while cellular automata with a λ-parameter close to 1 tend toward completely chaotic behavior; "complex" behavior lies in-between [16]. Assuming that different

materials have different inherent potentials for computation complexity with regards to evolution in-materio, the λ-parameter of different evolved cellular automata in a material can be used as a proxy for measuring the complexity ceiling of that material. Since this metric is a proxy metric, it is limited in scope to the specific methods used for evolution and interpretation of computation.

Care must be taken when using λ-parameter, as it is originally only well-defined for a subset of all cellular automata. A cellular automaton only has a λ-parameter if the non-quiescent state transitions, i.e. state transitions that are not transitions to the quiescent state, are randomly and uniformly distributed over the remaining non-quiescent states [16]. For binary cellular automata there can only be one non-quiescent state, which means that all binary cellular automata strictly speaking have a well-defined λ since non-quiescent state transitions are randomly and uniformly distributed over the only non-quiescent state. However, the lack of choice in non-quiescent states does alter the qualitative behaviors of binary cellular automata at high λ-parameters when compared to the original findings in [16]. High λ-parameter binary cellular automata will tend to frozen structures rather than chaotic behavior. As such, it can be useful to define a binary variant of the λ-parameter, the λ'-parameter, which is similar to the λ-parameter except that the quiescent state is always the most transitioned-to state. This means that the λ'-parameter is effectively a mirroring of the λ-parameter around $\lambda = 0.5$ as the maximum value.

Looking at the results from the evolution experiments, some conclusions can be made with regards to the complexity of the material. It seems that Elementary Cellular Automata with extreme λ-parameters, i.e. closer to $\lambda = 0$ and $\lambda = 1$, or closer to $\lambda' = 0$, evolve more easily than Elementary Cellular Automata that have a λ-parameter somewhere in-between. The Elementary Cellular Automata that evolve the least easily in the experimental setup are the ones with the largest λ'-parameters. An overview of evolution difficulty measured as the average number of generations used for evolutionary runs grouped by λ'-parameter can be seen in FIGURE 18. A different overview of evolution difficulty measured as the average fitness of the best individual of the last generation of each evolutionary run grouped by λ'-parameter can be seen in FIGURE 17. Both measures lead to similar considerations: Elementary Cellular Automata with extreme λ'-parameters evolve more easily than Elementary Cellular Automata with λ'-parameters close to 0.5. These findings are in-line with the intuition that more computationally complex cellular automata should be more difficult to evolve, if taken together with the

idea that the most computationally complex cellular automata appear at the Edge-of-Chaos [16], i.e. at the phase transition between ordered and chaotic behavior. Looking at the location of the Edge of Chaos in FIGURE 6, these results support the notion that for Elementary Cellular Automata the Edge of Chaos lies around λ-parameter values of ~0.5.

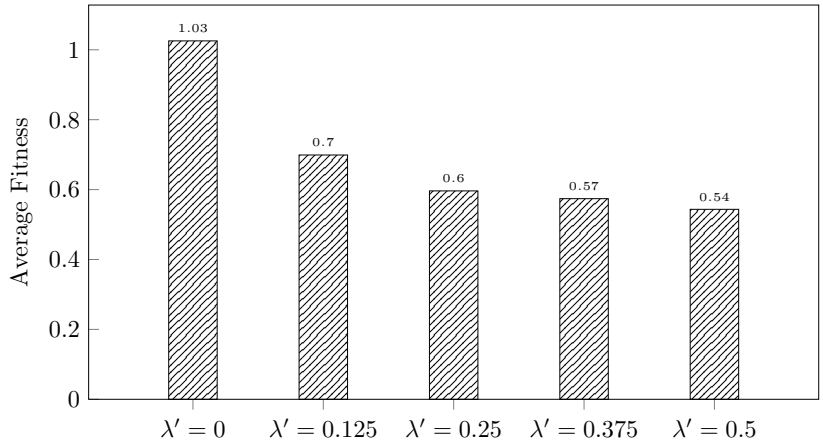


FIGURE 17: The average fitness of the best individual of the last generation of an evolutionary run grouped by λ'-parameter.

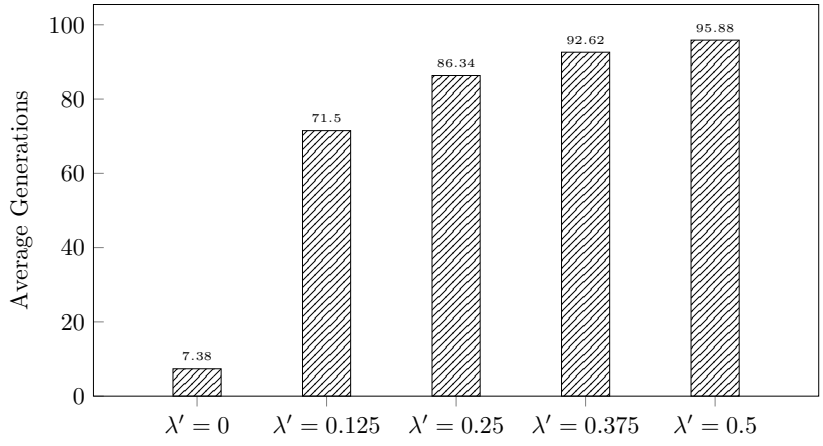


FIGURE 18: The average number of generations simulated per evolutionary run grouped by λ'-parameter.

6.2 Evolvability and Wolfram Classification

Additional insights to the questions around computational complexity in the material might be gleaned from looking at the evolvability of the different Elementary Cellular Automata grouped by Wolfram Classes.

In the Wolfram Classification, the classes are ordered by complexity, so if the hypothesis that less complex Elementary Cellular Automata evolve in-materio more easily than more complex Elementary Cellular Automata, then it is reasonable to expect that evolving a Class I automaton should on average require fewer generations than a Class II automation, a Class II automaton should on average require fewer generations than a Class III automaton, and finally a Class III automaton should require on average fewer generations to evolve than a Class IV automation.

FIGURE 19 shows the distributions of generations simulated in order to evolve an acceptable Elementary Cellular Automata for each of the four classes. The distributions do not include non-successfully evolved Elementary Cellular Automata.

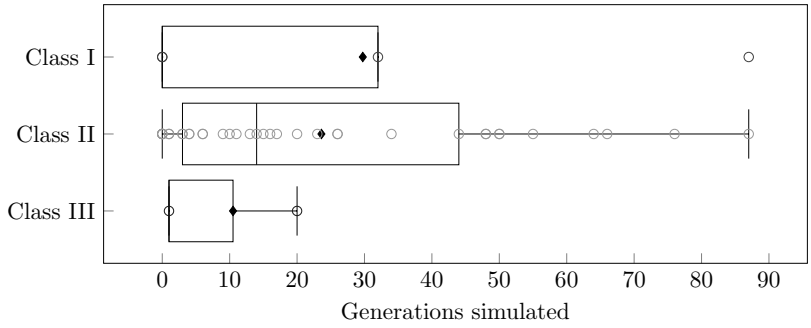


FIGURE 19: Distribution of evolution length as measured by generation count grouped by Wolfram Class. Circles show the number of generations required to evolve a rule. The boxes are Tukey-style boxplots, and show median and quartile values as vertical lines and average values as diamonds, with whiskers showing the smallest value larger than (the lower quartile - 1.5 IQR) and the largest value smaller than (the upper quartile + 1.5 IQR), IQR being the difference between the upper quartile and the lower quartile. Class IV is not present as there were no successful Class IV evolutionary runs.

FIGURE 20 shows the class distribution of the 42 successfully evolved Elementary Cellular Automata compared to the class distribution of all 256 Elementary Cellular Automata.

Of the 256 Elementary Cellular Automata, 25 (~9.8%) are Class I, 192

(~75.8% are Class II), 27 (~10.5%) are Class III, and 12 (~4.7%) are Class IV. If the opposite of what the hypothesis predicts were true, i.e. that Elementary Cellular Automata are on average equally likely to be successfully evolved in-materio regardless of Wolfram Class, a similar distribution of classes should be present in the set of successfully evolved Elementary Cellular Automata. Of the successfully evolved Elementary Cellular Automata, however, ~9.5% are Class I, ~85.7% are Class II, ~4.8% are Class III and 0% are Class IV. This is a very different class distribution than what should be expected if the any Elementary Cellular Automata were equally likely to evolve successfully. Hence, the results from the experiment indicate that there might be a correlation between cellular automata complexity and evolvability in-materio.

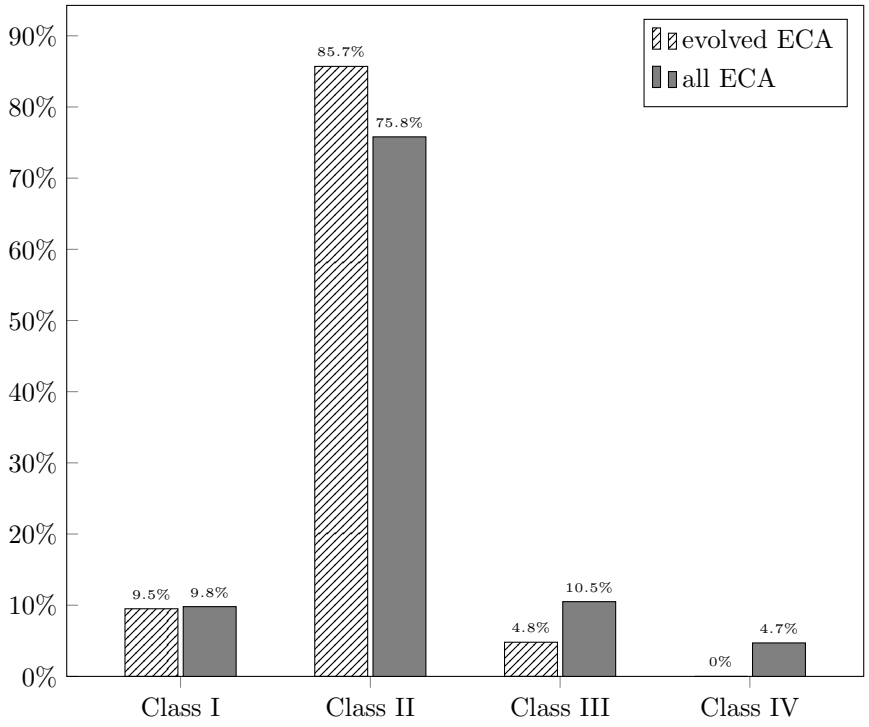


FIGURE 20: Elementary Cellular Automata Grouped by Wolfram Class.

6.3 Evolvability and Set Bits in an Elementary Cellular Automaton Rule

As a contrast to looking at correlations between cellular automata complexity and evolvability in-materio, other potential correlators should be looked at as

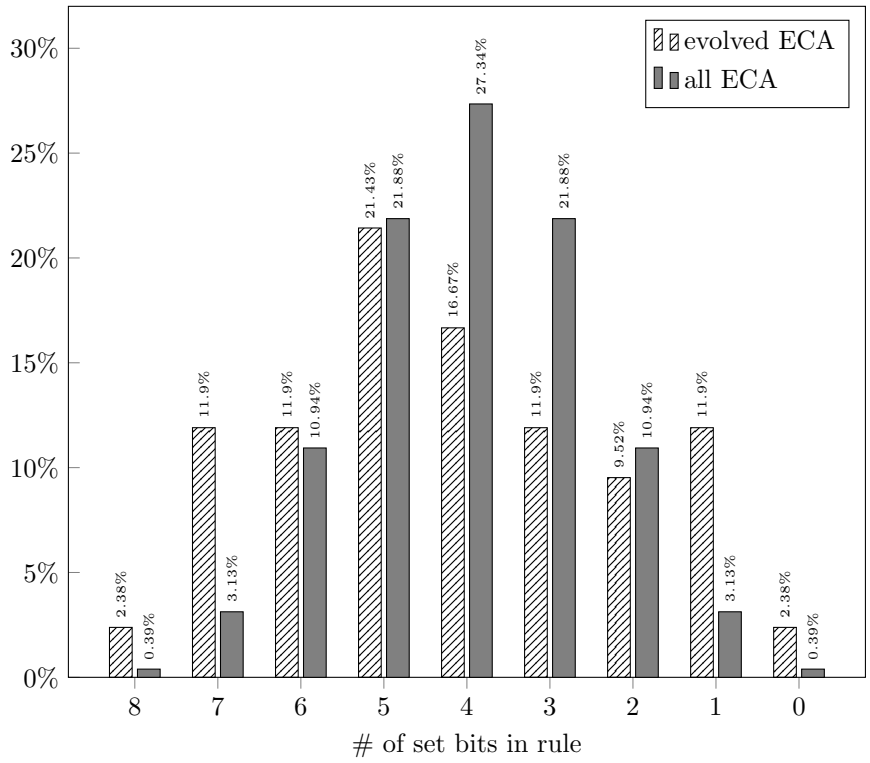


FIGURE 21: Elementary Cellular Automata grouped by set bits in rule.

well. Perhaps the evolvability of an Elementary Cellular Automata is not a function of its complexity, but rather simply the numbers of set bits in the Elementary Cellular Automaton Rule number. It is easy to imagine a simple hypothetical material for which this is the case – any material which favors set bits rather than unset bits as output would qualify. An extreme example is a hypothetical material which always outputs set bits regardless of input. Looking at the distribution of the amount of set bits in the 256 Elementary Cellular Automata compared to distribution of the amount of set bits in the 42 successfully evolved Elementary Cellular Automata, solutions with 6-8 bits set are over-represented, and solutions with 2-5 bits set are under-represented. Solutions with 0 or 1 set bits are over-represented again. A comparison of the two distributions can be seen in FIGURE 21.

Counting the number of set bits in any binary cellular automaton, and certainly therefore in the Elementary Cellular Automata, is analogous to calcu-

lating the λ-parameter of a rule.

6.4 Sensitivity Analysis

Only a single evolutionary run has been executed for each of the 256 Elementary Cellular Automata in the experiment detailed in Section 5. When taken individually, a single run for each of the 256 different rules is not enough to be able to draw meaningful conclusions about a single Elementary Cellular Automaton in-materio, statistically speaking. As an example, consider the evolution of Elementary Cellular Automata Rule_{54} in Section 4 and the experiment in Section 5. The latter was not able to evolve a suitable Rule_{54}, yet a suitable solution for that same rule was clearly found in the previous experiment with longer recording and evolutionary time. Care must be taken when analyzing the results from a statistical point-of-view. Looking at the experiment as a whole, and treating the different runs as repeat experiments over different classifications, is the better approach to extracting a meaningful interpretation.

For the purpose of reducing the worst-case evolution time for a single evolutionary run, each run was capped at 100 generations. That is, if a solution was not found after 100 generations, the run would be considered unsuccessful. This generation cap has probably pruned away a few would-be-successful evolutions had the generation cap been higher, e.g. capped at 1000 generations. Although more Elementary Cellular Automata were successfully evolved after few generations rather than many, it seems reasonable to assume that more generations per evolutionary run would ultimately yield more successful evolutions.

Again for the purpose of reducing the evolution time for a single evolutionary run, the fitness evaluation was changed to a $10\,\text{ms}$-based computation rather than a $100\,\text{ms}$-based computation. This change could also impact evolvability of a rule in the All-ECA experiment when compared to the Single-ECA experiment. The order of magnitude between maximum input oscillation frequency ($50\,\text{kHz}$) and the sampling frequency of the output electrode ($500\,\text{kHz}$) together with the material model, assuming a stabilization time of microseconds (or less), makes an impact on evolvability caused by change in execution time unlikely. Further, considering the fact that the $100\,\text{ms}$ value was chosen rather arbitrarily in the initial experiment, it seems at least intuitively unlikely that a change of execution time from $100\,\text{ms}$ to $10\,\text{ms}$ should greatly impact evolvability of a fit solution. Still, the impact of this change remains an open question.

Further, still motivated by time constraints, the population size in the ex-

periments was reduced from 40 to 20 individuals in each of the adult and child pools. A change in this direction generally increases the number of generations that must be simulated before an acceptable solution is found, and a too small population increases the risk of the evolutionary algorithm getting stuck at local maxima in the fitness space. Still, the decreased population size is still well within the limits of what has been shown to work for evolution in-materio in random single-walled carbon nanotube and polymer meshes. In a number of experiments, desired computation is successfully evolved in-materio using an evolutionary algorithm population size of 5 [5, 24, 25, 26], which is considerably less than the 20(+20) population size used in the latter experiments in this paper.

Performing a single fitness evaluation of a solution candidate in the material still takes on the order of 10 s to compute because of various unavoidable overheads. With the enormous number of fitness evaluations required, performing the experiments has taken several months of around-the-clock in-materio computation. Increasing the number of repeat runs of each rule evolution and perhaps also increasing the generation cap for each evolutionary run would improve results in terms of statistical significance. This was outside the scope of the paper.

6.5 Material Sample

The same material sample was used for all the experiments. Random single-walled carbon nanotube (SWCNT) and polymer mesh devices, as the name suggests, are randomly constructed. Because of this, different material samples may exhibit vastly different computational behavior. There are many different variables such as nanotube concentration, electrode layout, production methods and more that may improve or decrease the material's computational properties.

An abstract computational device evolved on one material sample cannot be used on a different material sample directly. This limits the commercial potential of SWCNT devices when used for evolution-in-materio, since while they can be efficiently mass-produced [9], each individual physical device needs to have a unique configuration evolved to be used.

6.6 Stability of Results

The stability of the results is greater than that of previous work [15, 18], and the evolved solutions seem stable enough to be called "stable" solutions in the context of Evolution-in-Materio. Looking at FIGURE 11 and FIGURE 12, there is sometimes a small separate clustered group of measurements far away

from the median which severely reduce the stability of the otherwise very tight clustering of measurements around the median. This seems strange, and may be caused by some complex intrinsic process within the material itself, but it may also be caused by some experimental error in process, equipment, software or similar. If the latter is the case, the true computational stability of the solutions may very well be much higher than the experiments herein.

6.7 Environmental Dependence

The experiments model the material as an ideal device that only reacts to electrical signals on the electrodes. In reality, the computational properties and process vary based on other external effects such as changes in temperature, light, and other environment variables. No special care was taken to maintain a stable environment – the experiments were run on a desk in a shared computer hardware laboratory in close proximity to noisy computers, a soldering station and multiple different types of lamps and light fixtures, as you might commonly expect to find in a computer hardware laboratory.

The relative stability of the results despite lack of a strictly controlled environment suggest that the material is reasonably invariant to the changing environmental effects of an indoor environment. This is also what one might expect when looking at the material from a material sciences perspective. The demonstrated environmental invariance in the computational substrate corroborates the attractiveness of single-walled carbon nanotube and polymer composite meshes as a computational substrate.

6.8 Speed of Computation

Currently, performing a computation in-materio takes on the order of $10\,\text{ms}$ to complete. This is because the input/output encoding is specified somewhat arbitrarily to last for that length of time. $10\,\text{ms}$ is quite slow compared to even consumer-grade conventional computers, which are easily capable of upwards of hundreds of millions of operations over the same time period. Computation speeds may be improved upon in further work.

6.9 Where Does Computation Take Place?

Does the computation actually take place in-materio? When performing evolution-guided search for computation in a material, the entire input domain and output range of the computational function is known, and a signal encoding and decoding process is performed off-material. This can make it hard to pinpoint exactly where the computation takes place. Certainly it is possible to construct a fitness evaluator and input/output encoding that is so complex that it can find computation in anything – even random noise. In such a case, the

computation is in reality happening outside of the material. How can the origin of computation be measured? It can be helpful to replace the material with different hypothetical materials and imagine what would happen if the same computations were performed using the switched hypothetical materials, but still using the same input and output coding schemes. Considering the following three hypothetical alternative materials, some insight might be gained into the computational complexity of the SWCNT material: 1. a computationally "dead" material that always outputs the same static signal(s); 2. a material that produces "true random noise" on its output(s) regardless of the input; and 3. a material that linearly combines its input(s) and passes it on to its output(s). Does the computation that allegedly happens in the real material also happen when the material is replaced with one these hypothetical materials? For one, the real material certainly out-performs the "dead" material – all of the implemented functions show a range that requires the output voltage to be above or below some static non-changing threshold level depending on the input. Since the expected output depends on the input, and the static threshold crucially does *not* change based on the input, it demonstrably performs more computation than the "dead" material.

Now, in the case of the random material, it is possible that the random output happens to measure on the right side of the threshold level for different inputs by pure chance. However, it will probably not do so very often, statistically speaking. The evolved devices presented in this thesis are all reasonably stable in their output, or at least much more stable than what one can expect from a "true" random material. This hints at an understanding where at least some of the computation happens in the material itself.

In the case of the linearly combining material, linear computation is possible in-materio almost by definition, but computations that are not linearly separable should not be implementable. Several of the evolved Elementary Cellular Automata, however, are not linearly separable functions. Thus, the SWCNT material seems to exhibit computational promise beyond linearly separable functions.

7 CONCLUSION

This paper has explored the idea of using Evolution-in-Materio to exploit a single-walled carbon nanotube and polymer composite random mesh material for abstracted computation using cellular automata. The goals were to investigate the capacity for computational complexity in the material-under-study, and to reason about the complexity ceiling for a computational substrate for

Evolution-in-Materio in a general sense. The experiments presented show that reasonably stable Class I, II, III and IV Elementary Cellular Automata can be successfully evolved in-materio. The degree of complexity of different abstract computational devices is correlated with the probability of a successful evolution of a physical implementation of those abstract devices. As such, the degree of success in evolution of Elementary Cellular Automata of different Wolfram Classes and with differing λ-parameters in-materio has been used as a proxy to measure the complexity ceiling for the material-under-study, and supports Langton's notions of complexity at the Edge-of-Chaos [16].

7.1 Further Work

Evolution-in-materio is a time-consuming approach to designing computational devices. The analysis and conclusions made in this paper could be strengthened significantly by increasing the number of evolutionary runs made to increase sample sizes. In order to do this efficiently, new implementation schemes could be devised that allow for shorter computation times in-materio for quicker fitness evaluation. This also synergizes well with lifting the efficiency of an evolved device out of the realm of proof-of-concepts and into the realm of situationally useful computational devices.

The current input/output encoding scheme for signals in and out of the material are not chainable without intermediate conversion. This limits the usefulness in creating larger composite computation devices through traditional componentized design. One possible avenue for further work is to search for an input-output compatible chainable representation that allows for feeding output from one device as input to the next.

Energy efficiency is one of the areas in which Evolution-in-Materio could show promise. No work has been done in this thesis to measure energy efficiency of the presented devices. One possible avenue for further work is to measure and compare the relative energy efficiency of different signal encoding schemes coupled with different abstract computational devices implemented in the material-under-study.

No special effort has been made to measure the environmental dependence of the material. One possible avenue for further work is to examine how environmental factors such as temperature, light and others affect computation in the material.

One possible avenue for further work is a hybrid conventional/materio system where Cellular Automata, e.g. the computation-universal Rule 110 [6], is implemented in-materio and used to calculate state transitions for a cellu-

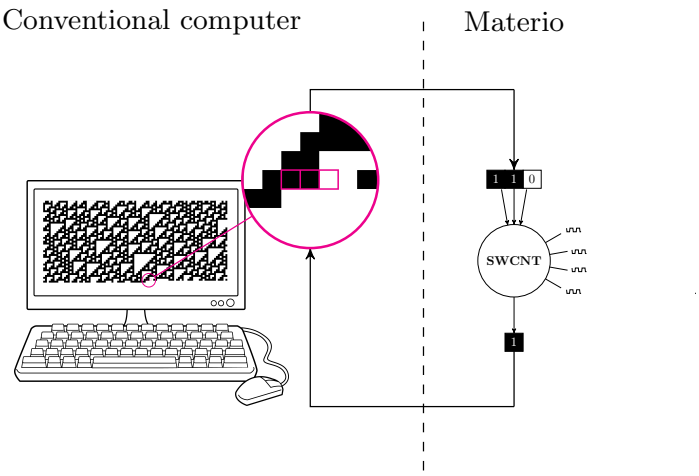


FIGURE 22: A setup for a practical simulation of Rule 110 in a hybrid conventional/materio device. The transitions are computed in-materio, and the state is stored on a conventional computer.

lar automaton simulation where the state of the simulation is kept track of on a traditional computer for practicality, e.g., a buffer. An illustration of a possible setup for such a simulation can be seen in FIGURE 22⋆ .

Ultimately, Evolution-in-Materio in the long term promises new possibilities for physical computational devices arising from specialized exploitation of substrates beyond what is possible with traditional design approaches. It is even possible to envision computing systems where material configurations are evolved "on-the-fly" to be used for a short period of period of time before it is discarded as the requirements of the environment changes, by way of analogy much like a just-in-time compiler from the world of programming language interpreters works.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP/2007-2013) under grant agreement 317662.

⋆ Desktop computer drawing adapted with permission from sweetclipart.com.

REFERENCES

[1] NASCENCE Project Website, http://nascence.no. [Accessed 28-May-2015].

[2] Andy Adamatzky and Larry Bull. (2009). Are complex systems hard to evolve? *Complexity*, 14(6):15–20.

[3] H. Broersma, F. Gomez, J. F. Miller, M Petty, and G. Tufte. (2012). Nascence project: Nanoscale engineering for novel computation using evolution. *International Journal of Unconventional Computing*, 8(4):313–317.

[4] Hajo Broersma, Julian F Miller, and Stefano Nichele. (2017). Computational matter: Evolving computational functions in nanoscale materials. In *Advances in Unconventional Computing*, pages 397–428. Springer International Publishing.

[5] KesterDean Clegg, JulianFrancis Miller, Kieran Massey, and Mike Petty. (2014). Travelling Salesman Problem Solved 'in materio' by Evolved Carbon Nanotube Device. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 692–701. Springer International Publishing.

[6] Matthew Cook. (2004). Universality in Elementary Cellular Automata. *Complex Systems*, 15(1):1–40.

[7] Sigve Sebastian Farstad. (2015). Evolving cellular automata in materio. Master's thesis, Norwegian University of Science and Technology.

[8] Sigve Sebastian Farstad, Stefano Nichele, and Gunnar Tufte. (2016). Towards standalone in-materio devices: Stable logic gates and elementary cellular automata in carbon nanotubes material. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 5246–5254. IEEE.

[9] J.C. Gabriel, K. Bradley, and P. Collins, (May 10 2006). Dispersed growth of nanotubes on a substrate. EP Patent App. EP20,030,808,389.

[10] S. Harding and J. Miller. (June 2004). Evolution in materio: initial experiments with liquid crystal. In *Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on*, pages 298–305.

[11] S. Harding and J.F. Miller. (June 2004). Evolution in materio: a tone discriminator in liquid crystal. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1800–1807 Vol.2.

[12] S. Harding and J.F. Miller. (June 2005). Evolution in materio: a real-time robot controller in liquid crystal. In *Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on*, pages 229–238.

[13] Simon Harding, Jan Koutnik, Klaus Greff, Jurgen Schmidhuber, and Andy Adamatzky. (2016). Discovering boolean gates in slime mould. *arXiv preprint arXiv:1607.02168*.

[14] Simon Harding and Julian F. Miller. (2005). Evolution in materio: Evolving logic gates in liquid crystal. In *In Proceedings of the workshop on unconventional computing at ECAL 2005 VIIIth European*, page 12.

[15] A. Kotsialos, M.K. Massey, F. Qaiser, D.A. Zeze, C. Pearson, and M.C. Petty. (September 2014). Logic gate and circuit training on randomly dispersed carbon nanotubes. *International journal of unconventional computing.*, 10(5-6):473–497.

[16] C. Langton. (June 1990). Computation at the Edge of Chaos Phase Transitions and Emergent Computation. *Physica D*, 42:12–37.

[17] Odd Rune Lykkebø, Stefano Nichele, and Gunnar Tufte. (2015). An investigation of square waves for evolution in carbon nanotubes material. In *13th European Conference on Artificial Life*. Springer.

[18] OddRune Lykkebø, Simon Harding, Gunnar Tufte, and JulianF. Miller. (2014). Mecobo: A Hardware and Software Platform for In Materio Evolution. In Oscar H. Ibarra, Lila Kari, and Steffen Kopecki, editors, *Unconventional Computation and Natural Computation*, volume 8553 of *Lecture Notes in Computer Science*, pages 267–279. Springer International Publishing.

[19] G. J. Martinez, J. C. Seck-Tuoh-Mora, and H. Zenil. (2012). Wolfram's Classification and Computation in Cellular Automata Classes III and IV. *ArXiv e-prints*.

[20] Genaro Juárez Martínez, Andrew Adamatzky, and Harold V McIntosh. (2006). Phenomenology of glider collisions in cellular automaton rule 54 and associated logical gates. *Chaos, Solitons & Fractals*, 28(1):100–111.

[21] Kieran Massey, (2014). Material for NTNU 3. private communication.

[22] Julian F Miller and Keith Downing. (2002). Evolution in materio: Looking beyond the silicon box. In *Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on*, pages 167–176. IEEE.

[23] Melanie Mitchell, Peter T. Hraber, and James P. Crutchfield. (1993). Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations. *Complex Systems*, 7:89–130.

[24] M. Mohid, J.F. Miller, S.L. Harding, G. Tufte, O.R. Lykkebo, M.K. Massey, and M.C. Petty. (Dec 2014). Evolution-in-materio: A frequency classifier using materials. In *Evolvable Systems (ICES), 2014 IEEE International Conference on*, pages 46–53.

[25] M. Mohid, J.F. Miller, S.L. Harding, G. Tufte, O.R. Lykkebo, M.K. Massey, and M.C. Petty. (Sept 2014). Evolution-in-materio: Solving function optimization problems using materials. In *Computational Intelligence (UKCI), 2014 14th UK Workshop on*, pages 1–8.

[26] Maktuba Mohid, JulianFrancis Miller, SimonL. Harding, Gunnar Tufte, OddRune Lykkebø, MarkK. Massey, and MichaelC. Petty. (2014). Evolution-In-Materio: Solving Machine Learning Classification Problems Using Materials. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 721–730. Springer International Publishing.

[27] S. Nichele, D. Laketic, O. R. Lykkebø, and G. Tufte. (2015). Is there chaos in blobs of carbon nanotubes used to perform computation? In *7th International Conference on Future Comp. Tech. and Applications*. XPS Press.

[28] Stefano Nichele, Odd Rune Lykkebø, and Gunnar Tufte. (2015). An investigation of underlying physical properties exploited by evolution in nanotubes materials. In *Proceedings of 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*. IEEE.

[29] G. Pask. (1958). Physical analogues to the growth of a concept. *Mechanisation of Thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory.*, pages 879—922.

[30] E. S. Snow, J. P. Novak, P. M. Campbell, and D. Park. (2003). Random networks of carbon nanotubes as an electronic material. *Applied Physics Letters*, 82(13):2145–2147.

[31] Susan Stepney. (2008). The Neglected Pillar of Material Computation. *Physica d-Nonlinear phenomena*, 237(9):1157–1164.

[32] A. Thompson. (1997). An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics. In Tetsuya Higuchi, Masaya Iwata, and L. Weixin, editors, *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, pages 390–405, Berlin. Springer-Verlag.

[33] Stephen Wolfram. (Jul 1983). Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55:601–644.

[34] Stephen Wolfram. (2002). *A New Kind of Science*. Wolfram Media.