International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:8, 2008

Mimicking morphogenesis for robust behaviour of cellular architectures

David Jones, Richard McWilliam, Alan Purvis

***Abstract*—**Morphogenesis is the process that underpins the self-organised development and regeneration of biological systems. The ability to mimick morphogenesis in artificial systems has great potential for many engineering applications, including production of biological tissue, design of robust electronic systems and the co-ordination of parallel computing. Previous attempts to mimick these complex dynamics within artificial systems have relied upon the use of evolutionary algorithms that have limited their size and complexity. This paper will present some insight into the underlying dynamics of morphogenesis, then show how to, without the assistance of evolutionary algorithms, design cellular architectures that converge to complex patterns.

***Keywords*—**Morphogenesis, Regeneration, Robustness, Convergence, Cellular automata

I. Introduction

Morphogenesis provides biological systems with a robust framework for the differentiation of undeveloped or partially developed cells. Remarkable examples of biological systems that utilize morphogenesis include:

1) The human liver: This organ is capable of withstanding and repairing damage to up to two-thirds of its constituent cells.
2) The Salamander: If bissected from its tail, the tail will often grow back.
3) Ascidians (marine filter feeders), whose blood cells alone have been reported to give rise to a fully functional organism[1].

Morphogens are soluble proteins that diffuse about source cells within developing tissue. These chemical messages co-ordinate the differentiation of cells, determining what type of cell belongs where in the tissue. Alan Turing, in his seminal paper "The Chemical basis of Morphogenesis" [3] showed that systems of multiple chemicals that diffuse and interact can create some of the irregular patterns found in biological systems.

Wolpert[4] compared morphogenesis to the formation of a simple pattern representing the French flag. This pattern has served as the benchmark for those seeking to mimic morphogenesis. Miller *et al* presented an evolved solution to Wolpert's French Flag on a cellular architecture[2], that proved to be capable of repairing damage incurred to up to 25% of its cells. That Miller *et. al*'s work was constrained to simple patterns and limited reliability is because the relationship between the rules obeyed by each cell and the resulting pattern is unknown. Researchers in this field have used a supervised evolutionary algorithm to evolve local rules against a cost function that is defined as the difference between the desired pattern and the pattern that results from the execution of the system.

This paper will first show that cellular automata can be constrained to form robust patterns that perpetually repair themselves in the event of corruption, then present a mechanism for designing cellular automata to form arbitrary patterns that does not require the use of an evolutionary algorithm.

d.h.jones2@dur.ac.uk, r.p.mcwilliam@dur.ac.uk, alan.purvis@dur.ac.uk
University of Durham, School of Engineering , South Road, Durham, DH1 2PQ

II. A cellular automata model of morphogenesis

A two-dimensional cellular automata of identical cells can be used to mimick morphogenesis. Each cell of the automaton receives inputs from its four immediate neighbours, those to the north, south, east and west of itself. These inputs are "morphogens" that, via a set of rules determine both the state of the cell and the form of the output morphogen from the cell. The set of rules obeyed by each cell is identical for each cell. The biological analogue of the output of an automata cell would be the proteins it forms respective of its location within the body. Within the model, the organ will be a pattern of colours and each cell output will be a colour. Each cell will communicate an integer state with its immediate neighbours.

At each discrete time-step every cell computes its next state. Let us index each cell with the tuple (i, j), then describe the state of each cell at time t with an integer, $c_{i,j,t}$ and the pattern of the entire array as a matrix, $\mathbf{C}_t$ (see figure 1a).

III. A framework for the design of robust cellular automata

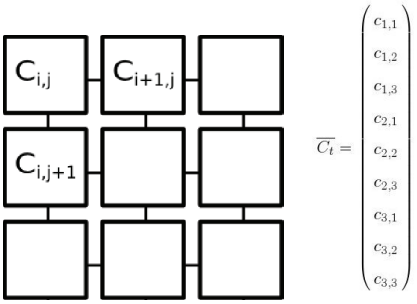
If $\mathbf{C}_0$ is the initial pattern of C_t, $f(\mathbf{C}_0)$ is its subsequent pattern after one time step, and $f(f(\mathbf{C}_0))$ or $f^2(\mathbf{C}_0)$ is its pattern at $t = 2$; where the function $f()$ describes the transition function. The matrix $\mathbf{C}_t$ is first transcribed into a row-major vector, $\overline{C_t}$ (figure 1b) in order for $f()$ to be a linear function of matrix algebra.

Let us now define a simple transition function from one time step to the next:

$$c_{i,j,t+1} = nc_{i,j-1,t} + wc_{i-1,j,t} + ec_{i+1,j,t} + sc_{i,j+1,t} + xc_{i,j,t} + k \quad (1)$$

Where n, e, s, w and x are coefficents of the state of neighbours of each cell, and of the state of the cell itself.

A transition function for the entire array can be formed from (1) such that $f(C_t) = \mathbf{T}\overline{C_t} + \overline{K}$ where $\overline{K}$ is a constant and the transtion matrix (for a 3 by 3 CA), $\mathbf{T}$, takes the form:



(a) (b)

Fig. 1. Index of CA elements, and a row-major vector equivalent

$$\mathbf{T} = \begin{pmatrix} x & e & 0 & s & 0 & 0 & 0 & 0 & 0 \\ w & x & e & 0 & s & 0 & 0 & 0 & 0 \\ 0 & w & x & 0 & 0 & s & 0 & 0 & 0 \\ n & 0 & 0 & x & e & 0 & s & 0 & 0 \\ 0 & n & 0 & w & x & e & 0 & s & 0 \\ 0 & 0 & n & 0 & w & x & 0 & 0 & s \\ 0 & 0 & 0 & n & 0 & 0 & x & e & 0 \\ 0 & 0 & 0 & 0 & n & 0 & w & x & e \\ 0 & 0 & 0 & 0 & 0 & n & 0 & w & x \end{pmatrix} \quad (2)$$

The spacing of the coefficients n, e, s, w and x within $\mathbf{T}$ depend on the size of the CA.

By the repeated application of $f()$, the transition from $\mathbf{C}_0$ to $\mathbf{C}_t$ (where $t > 1$) becomes a non-linear function:

$$\begin{aligned} f^2(\overline{C_0}) &= \mathbf{T}(\mathbf{T}\overline{C_0} + \overline{K}) + \overline{K} \\ f^3(\overline{C_0}) &= \mathbf{T}(\mathbf{T}(\mathbf{T}\overline{C_0} + \overline{K}) + \overline{K}) + \overline{K} \\ f^3(\overline{C_0}) &= \mathbf{T}^3\overline{C_0} + \mathbf{T}^2\overline{K} + \mathbf{T}\overline{K} + \overline{K} \end{aligned}$$

This can be expanded to form:

$$f^t(\overline{C_0}) = \mathbf{T}^t\overline{C_0} + \mathbf{T}^{t-1}\overline{K} + \mathbf{K}^{t-2}\overline{K} + ... + \mathbf{T}\overline{K} + \overline{K}$$

Using the geometric series equation this can be simplified to form:

$$f^t(\overline{C_0}) = \mathbf{T}^t\overline{C_0} + (\frac{\mathbf{I} - \mathbf{T}^{t-1}}{\mathbf{I} - \mathbf{T}})\overline{K}$$

Equation (III) determines the pattern formed after t iterations of the transition function (1) have been applied to every cell synchronously.

Given a sufficiently large t, in order for the dynamic non-linear system to converge, the final pattern, $\overline{C_t}$, must be independent of the initial pattern, $\overline{C_0}$. Thus no matter what the starting pattern (where $t = 0$ refers to the initial pattern or any pattern that might be the result of system corruption), the pattern of cell states will always return to the same stable pattern.

To satisfy this constraint $\mathbf{T}^t$, the coefficient of $\overline{C_0}$, must equal zero. For this to be so, referring to the coefficients of the states of the cells above, below, left and right and of the cell itself respectively, the following three constraints must hold:

1) Either n or s must equal zero
2) Either e or w must equal zero
3) x must equal zero.

This tells us that each cell must determine its next input according to the state of one neighbouring cell per axis.

By expanding this analysis to an alternative sum-of-products modulo-two transition function it can be shown that this conclusion also holds true for combinatorial transition functions.

IV. Designing robust cellular architectures to form specific patterns

Let us consider the requirements on the implementation of robust cellular architectures. In a two-dimensional automata each cell with have at most two inputs, one nearest-neighbour from each axis. Each cell will obey the same transition function. We require that this architecture should be able to converge upon any specified pattern.

If a combinatorial transition function (a function formed using simple logic operators) is used, certain patterns will exist that cannot be formed. This is because the same two-input combination cannot be mapped to many different outputs. Thus if the pattern we desire the automata to converge to requires two cells to have the same inputs, but have different outputs, a more elaborate solution is required.

Instead we could use two combinatorial functions implemented using look-up tables (LUT). The first function, $g()$, determines the next-state of the cell based on the current state of two of its neighbours, one to its left or right, the other above or below the cell. The second function, $h()$, will use the current state of the cell to determine its output in a many-to-one mapping. Figure 2 shows one possible implementation of the transition function defined by the LUTs.

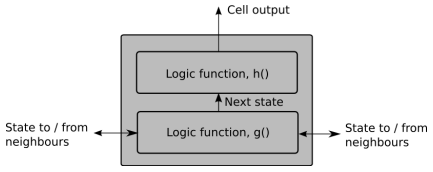


Fig. 2. An implementation of the cell transition function using LUTs

To determine the entries to be stored in each LUT we need a design algorithm. At each time-step, every cell determines its next state. Having reached the desired final state, the cell transition rules of the automata must ensure that the next-state of the automata is the same as the current state. Therefore the necessary LUT entries can be derived from the final state by determining the two-inputs of each cell and its next-state output whilst in its correct state.

A further complication exists because the output of each cell is not necessarily the same as its current state. Hence we need some means of assigning a final state to each cell. During execution the pattern emerges from one corner of the automata (henceforth referred to as the origin) and progresses towards the opposite corner of C_t (See figures 3 and 4). If we progress through the cells in the same direction, assigning each cell a state that, where possible, has already been assigned to previous cells, it won't be necessary to re-solve portions of the pattern as the algorithm progresses. Listing 1 shows a pseudo-code implementation of this design algorithm.

Listing 1. Design algorithm pseudo-code

```
For each of 4 reflections about the centre of the pattern:
        For each cell in the pattern, in order from top-left to bottom-right:
                Cell state = 0
                While no solution has been found:
                        Determine the inputs to the cell
                        Test the state-to-output mapping against previously
                                determined mappings. If there is a conflict increase
                                 the cell state by 1
                        Test the two-input to cell-state mapping against previously
                                 determined mappings. If there is a conflict
                                increase the state by 1
```

Thus each feature of the desired pattern must have its position in the pattern determined from another feature closer to the origin, or the origin itself. Since each cell can only communicate with its immediate neighbours, each position must be determined by a non-repeating sequence of state values. By progressing through the cells from the corner opposite the origin towards the origin, and attempting to assign each cell with previously used assignments first, these position-determining sequences will use a minimal number of states.

V. Results - a robust Venetian dragon pattern

The design framework we have presented is not limited by the same constraints and is thus capable of forming much more complicated patterns. Thus, rather than form the simple French flag pattern we have chosen to create a robust pattern of the dragon from the Venetian flag.

Using the algorithm describd in section four, a solution to the Venetian dragon pattern on a 75 by 50 array of cells was found using 2457 states and 3627 rules. Note that there are fewer states than would be required to index each cell uniquely. Figures 3 and 4 show the Venetian dragon pattern developing from a null and a corrupt state.

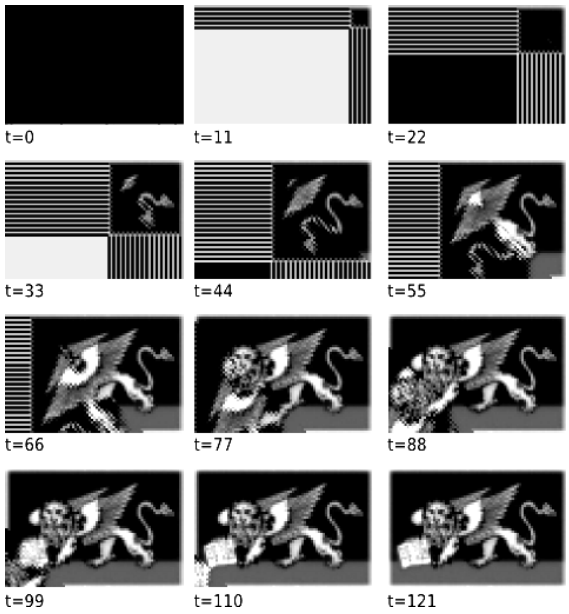


Fig. 3. A Venetian dragon pattern forming from null initial conditions

VI. Conclusions

This report has sought to introduce a mathematical analysis of an artificial cellular system inspired by the biological process of morphogenesis. The results of this analysis include:

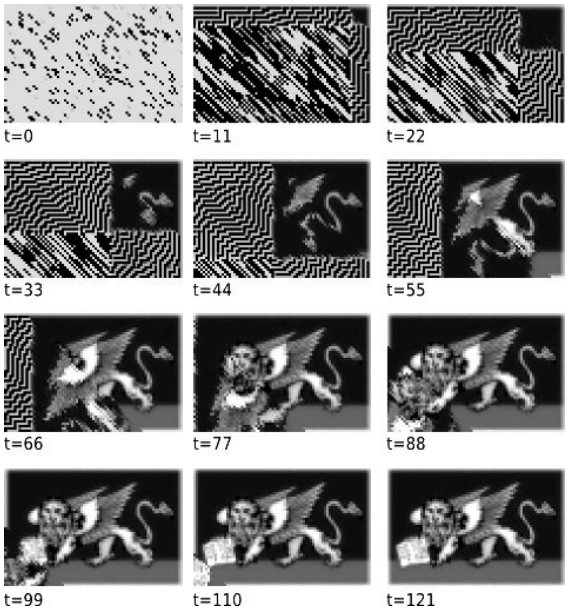


Fig. 4. A Venetian dragon pattern forming from partially corrupted initial conditions

1) That to ensure the final pattern is completely robust, it is necessary that the transition rules stored by each cell are independent of the current cell state. In addition the rules can only depend upon the state of one cell per axis, either the cell to the left or to the right, either the cell above or the cell below.
2) That it is possible to find a mapping from the automata patterns to a set of transition rules using the algorithm proposed in section four.

Morphogenesis has been demonstrated as an effective tool for forming patterns on cellular arrays. Thus remains the question, where else is it applicable? A sufficiently complex self-assembling robot will require a morphogenesis algorithm to co-ordinate the assembly process. The distribution and co-ordination of processes across large computing farms may benefit from this self-organising technique. Systems that cannot afford failure could benefit from an architecture that is intrinsically robust. More generally the approach should be of use to others in the field of bio-mimicry, in particular those working to imitate self-replicating systems.

References

[1] N. J. Berrill and A. Cohen. Regeneration in clavellina lepadiformis. *Journal of experimental biology*, 13, 1936.
[2] J. Miller and W. Banzhaf. Evolving the program for a cell: From french flags to boolean circuits. *On Growh, Form and Computers*, 2003.
[3] A. Turing. The chemical basis of morphogenesis. *Philos, Trans. Roy. Soc.*, Ser. B 237, 37, 1950.
[4] L. Wolpert. Positional information and the spatial pattern of cellular differentiation. *J Theor. Biol.*, 25:1-47, 1969.