TOC DIGITAL ASSIGNMENT -1

17BIT0133

V.Ranadeeshwar

Topic :

Cyclic Cellular Automata -Working principle and a very Detailed survey on its applications.

Introduction:

A **cyclic cellular automaton** is a kind of cellular automaton rule developed by David Griffeath and studied by several other cellular automaton researchers. In this system, each cell remains unchanged until some neighbouring cell has a modular value exactly one unit larger than that of the cell itself, at which point it copies its neighbour's value. One-dimensional cyclic cellular automata can be interpreted as systems of interacting particles, while cyclic cellular automata in higher dimensions exhibit complex spiraling behaviour.

The one-dimensional cyclic cellular automaton has been extensively studied by Robert Fisch, a student of Griffeath.[1] Starting from a random configuration with *n* = 3 or *n* = 4, this type of rule can produce a pattern which, when presented as a time-space diagram, shows growing triangles of values competing for larger regions of the grid.

The boundaries between these regions can be viewed as moving particles which collide and interact with each other. In the three-state cyclic cellular automaton, the boundary between regions with values *i* and *i* + 1 (mod *n*) can be viewed as a particle that moves either leftwards or rightwards depending on the ordering of the regions; when a leftward-moving particle collides with a rightward-moving one, they annihilate each other, leaving two fewer particles in the system. This type of ballistic annihilation process occurs in several other cellular automata and related systems, including Rule 184, a cellular automaton used to model traffic flow.[2]

In the *n* = 4 automaton, the same two types of particles and the same annihilation reaction occur. Additionally, a boundary between regions with values *i* and *i* + 2 (mod *n*) can be viewed as a third type of particle, that remains stationary. A collision between a moving and a stationary particle results in a single moving particle moving in the opposite direction.

However, for *n* ≥ 5, random initial configurations tend to stabilize quickly rather than forming any non-trivial long-range dynamics. Griffeath has nicknamed this dichotomy between the long-range particle dynamics of the *n* = 3 and *n* = 4 automata on the one hand, and the static behavior of the *n* ≥ 5 automata on the other hand, "Bob's dilemma", after Bob Fisch.

Rules:

As with any cellular automaton, the cyclic cellular automaton consists of a regular grid of cells in one or more dimensions. The cells can take on any of n-1 states, ranging from to n. The first generation starts out with random states in each of the cells. In each subsequent generation, if a cell has a neighboring cell whose value is the successor of the cell's value, the cell is "consumed" and takes on the succeeding value. More general forms of this type of rule also include a *threshold* parameter, and only allow a cell to be consumed when the number of neighbors with the successor value exceeds this threshold.

The Nature of Cellular Automata and a Simple Example :

Cellular automata are simple mathematical idealizations of natural systems. They consist of a lattice of discrete identical sites, each site taking on a finite set of, say, integer values. The values of the sites evolve in discrete time steps according to deterministic rules that specify the value of each site in terms of the values of neighboring sites. Cellular automata may thus be considered as discrete idealizations of the partial differential equations often used to describe natural systems. T.heir discrete nature also allows an important analogy with digital computers: cellular automata may be viewed as parallelprocessing computers of simple construction. As a first example of a cellular automaton, consider a line of sites, each with value 0 or I (Fig. I). Take the value of a site at position i on time step t to be a (t). One very simple rule I for the time evolution of these site values is

a(I+I) = a (I) + a(I) mod 2 .

Related works:

Modern urban traffic:

In this we introduce a new cellular automata approach to construct an urban traffic mobility model. Based on the developed model, characteristics of global traffic patterns in urban areas are studied. Our results show that different control mechanisms used at intersections such as cycle duration, green split, and coordination of traffic lights have a significant effect on inter vehicle spacing distribution and traffic dynamics. These findings provide important insights into the network connectivity behaviour of urban traffic, which are essential for designing appropriate

routing protocols for vehicular ad hoc networks in urban scenarios.
Cyclic cellular automata have properties akin to cyclic particle systems, introduced by Bramson and Griffeath, which are continuous-time Markov processes on $\{0, 1, ..., N-1\}^{Zd}$ with a stochastic evolution rather than a deterministic one. In one dimension, some clustering results about cyclic cellular automata are presented; the analogous problems in the context of cyclic particle systems are still unresolved. Also, we discuss the consequences of these clustering results in the context of one-dimensional systems of particles with various interaction mechanisms. Finally, we touch upon the work currently being done on two-dimensional cyclic particle systems and cellular automata.

Modelling excitable cells using cycle-linear hybrid automata:

Cycle-linear hybrid automata (CLHAs), a new model of excitable cells that efficiently and accurately captures action-potential morphology and other typical excitable-cell characteristics such as refractoriness and restitution, is introduced. Hybrid automata combine discrete transition graphs with continuous dynamics and emerge in a natural way during the (piecewise) approximation process of any nonlinear system. CLHAs are a new form of hybrid automata that exhibit linear behaviour on a per-cycle basis but whose overall behaviour is appropriately nonlinear. To motivate the need for this modelling formalism, first it is shown how to recast two recently proposed models of excitable cells as hybrid automata: the piecewise-linear model of Biktashev and the nonlinear model of Fenton–Karma. Both of these models were designed to efficiently approximate excitable-cell behaviour. We then show that the CLHA closely mimics the behaviour of several classical highly nonlinear models of excitable cells, thereby retaining the simplicity of Biktashev's model without sacrificing the expressiveness of Fenton–Karma. CLHAs are not restricted to excitable cells; they can be used to capture the behaviour of a wide class of dynamic systems that exhibit some level of periodicity plus adaptation.

Seed encoding with LFSRs and cellular automata:

Reseeding is used to improve fault coverage of pseudo-random testing. The seed corresponds to the initial state of the PRPG before filling the scan chain. In this paper, we present a technique for encoding a given seed by the number of clock cycles that the PRPG needs to run to reach it. This encoding requires many fewer bits than the bits of the seed itself. The cost is the time to reach the intended seed. We reduce this cost using the degrees of freedom (due to don't cares in test patterns) in solving the equations for the seeds. We show results for implementing our technique completely in on-chip hardware and for applying it from a tester. Simulations show that with low hardware overhead, the technique provides 100% single-stuck fault coverage. Also, when compared with conventional reseeding from an external tester or on-chip ROM, the technique reduces seed storage by up to 85%. We show how to apply the technique for both LFSRs and CA.

Another model is :
This presents a technique for building complex and adaptive meshes for urban and architectural design. The combination of a self-organizing map and cellular automata algorithms stands as a method for generating meshes otherwise static. This intends to be an auxiliary tool for the architect or the urban planner, improving control over large amounts of spatial

information. The traditional grid employed as design aid is improved to become more general and flexibility The problem description would be :
Meshes introduce two issues; these are the limitations we want to solve with this proposal: Fixed topology: basic design meshes have fixed topology, as they aim for simplicity and ease of use for the designer. In contrast with the usability, the information this meshes use to be built from is inherently dynamical in its topology, as one consequence of complexity in real-life originated data. In other words, spatial relationships are not kept over time in urban reality. So our tool should be able to assume that feature and develop the framework to process such variations. Shape constrains: refers to the land division methods and geometrical structures that conform the actual plan. Historical and cultural diversity show us several different ways to accomplish this goal. However, any taken decision at this aspect would limit the method's generality. This suggests that building a mesh where basic units are not limited by enclosures, thus conforming shapes, but turning its basic units into something shapeless would be desirable. This is quite obvious but not a common design approach, as it implies a higher level of abstraction and a bigger leap towards materialization of a design idea. The objective is to build a new mesh that improves on the basis of these two limitations for the specific task of helping

the urban planner. After that we should be able to use it in order to evaluate its first results.

Proposed technique:

1. Motivations for SOM and CA combined approach
   We detected interesting properties in both models that acting indepently solve different aspects of the grid (see Table 1). Basic SOM algorithms work reconfiguring their neurons' weight in a competitive basis, resulting in a problem-specific distribution while preserving topology of the map. However, CA algorithms work efficiently calculating relationships among cells, with just an initial configuration and no needed input data. On the other hand, classic CA rely on fixed-topology lattices, being highly suitable for massively parallel calculations, although they are not limited to be rectangular and uniform [Abdalla, Setoodeh, Gürdal, 2006]. Self-Organizing Maps Cellular Automata preserves topology of the map relies on topologically static lattice works on input data works from seed and sets of rules problem-specific adaptation (outer generation) cell relationships (inner generation) Table 1. SOM and CA for design. These features suggest the possibility of dividing the design problem into two conceptually complementary parts: ● Outer generation: refers to the ability of the design system to assimilate external information and reconfigure itself

depending on the input. ● Inner generation: concerning the inner properties of the system, not directly depending on the external constrains of the design problem. Although it can introduce external variables to the system, the only potential effect on the global qualities could be emergent, thus unpredictable. 2. Kohonen's self-organizing map SOM are Artificial Neural Networks that carry out their adjustment process through the unsupervised learning paradigm, and the input data are called unlabeled data. As a simplified definition, we can say that, in a topologypreserving map, units located physically next to each other will respond to classes of input vectors that are likewise next to each other. Although it is easy to visualize this in two-dimensional array, it is not so easy in a high-dimensional space. N-dimensional input vectors are projected down on the two-dimensional map in a way that maintains the natural order of the input vectors. This dimensional reduction could allow us to visualize easily relationships among the data that otherwise might go unnoticed [Freeman, Skapura, 1991]. A basic SOM is able to reconfigure its weights distribution depending on the training data. Weight vectors correspond to the classes the algorithm is finding in the training set. If we graph the weights vectors together with the input vectors we get a similarities diagram where the neuron units are shown as representatives of a data class from that input. We can describe the learning process by the equation wi = ⟨(t)(x – wi) U (yi) 4 International Conference «Information Research & Applications» - i.Tech 2007 Where wi is the weight vector of the i unit and x is the input vector. The function U(yi) is zero unless yi > 0 in which case U(yi) = 1 so only active units will learn. The factor α(t) is a function of time to allow its modification as the learning process progresses. As a competitive structure a winning unit is determined for each input vector based on the similarity between the input and the weight vectors. The winning unit is evaluated by || x - wc || = mini { || x – wi || } Finally the updated weights are those of the winning unit plus a defined neighborhood with certain decay.

The conclusion would be

The degree of success this technique could have in the design process is a matter of discussion both in the practical and the theoretical domains. However, different applications have already shown that it can be used as an aid tool with more general ambit than the regular grid. Becoming more than an aid tool, capable of synthesizing part of a design is a promising but ambitious objective. Future work will develop this issues: ● Input data preprocessing ●

Feedback flow as a more powerful and complex mechanism ● Multidimensionality of the input data ● Creation of layered or 3-dimensional meshes

Results:

Merging the two algorithms: results Both programs were implemented with the ability to export their results, so we can use the frozen data externally. In this last section it is described briefly how this information has been introduced into the designer's workflow, the last step of the technique. The most important thing that has to be performed is to process that information into a 3-dimensional geometry inside a design tool. The chosen platform was the open-source 3d modeling software called Blender, running an embedded Python interpreter. The python algorithm is straightforward. These steps are to be followed: 1. Read the SOM mesh data points 2. Read the CA cells status 3. Match the SOM neurons with the CA cells, activating or deactivating the neurons 4. Project the SOM in the 3d space, on the XY plane International Conference «Information Research & Applications» - i.Tech 2007 7 5. Optional use of an interpretative/generative algorithm to create volumes or additional point properties. It is an interesting step, in which relies the final aspect of the urban plan as it introduces the actual phenotype directly. It is important to notice that although it will introduce great differences in the system, the underlying structure will remain dependent on the previous steps. If this step is avoided, the designer will work directly with the imported 2-dimensional mesh.

References:

1)SELF-ORGANIZING MAP AND CELLULAR AUTOMATA COMBINED TECHNIQUE FOR ADVANCED MESH GENERATION IN URBAN AND ARCHITECTURAL DESIGN Álvaro Castro Castilla, Nuria Gómez Blas.

2) CELLULAR AUTOMATA AND APPLICATIONS GAVIN ANDREWS

3) **Seed encoding with LFSRs and cellular automata**

Ahmad A. Al-YamaniStanford University, Stanford, CAEdward J. McCluskeyStanford University, Stanford, CA