Evolution of Asynchronous Cellular Automata: Finding the Good Compromise

Mathieu S. Capcarrere
Logic Systems Laboratory
School of Computer and Communication Sciences
Swiss Federal Institute of Technology, Lausanne
CH-1015 Lausanne, Switzerland
E.mail: mathieu.capcarrere@epfl.ch

ABSTRACT

One of the prominent features of the Cellular Automata (CA) model is its synchronous mode of operation, meaning that all cells are updated simultaneously. But this feature is far from being realistic from a biological point of view as well as from a computational point of view. Past research has mainly concentrated on studying Asynchronous CAs in themselves, trying to determine what behaviors were an "artifact" of the global clock. In this paper, I propose to evolve Asynchronous CAs that compute successfully one of the well-studied task for regular CAs: The synchronization task. As I will show evolved solutions are both unexpected and best for certain criteria than a perfect solution.

The model used is fully asynchronous. Each cell has the same probability p_f of not updating its state at each step.

THE ADVANTAGES OF REDUNDANT CELLULAR AUTOMATA

The extremely weak capabilities of binary CAs to cope with even limited asynchrony called for the use of redundant CA to deal with full asynchrony. I call redundant CA, a CA which uses more states in the asynchronous mode than is necessary in the synchronous mode to solve the same task. The idea behind redundancy is that the information in a CA configuration is not only the current state and the topology, but also the *timing*.

A Simple and Perfect Time-Stamping Method

If we are looking for a method to perfectly correct asynchrony, then all information should be maintained. That is to say, all the 3-tuples (c, i, t), where c is the state of cell i at time t, of the synchronous case should be reconstructible in the asynchronous case. A time-stamp added to each cell so that the cell may know if it is ahead of one of its neighbors does the trick. The minimum value of the time-stamp, not to confuse between being ahead or being late, is 3. If each cell stores both its current and last state, the new CA is both able to know if it can update and how it should update. Thus we can design a $3 * q^2$ state CA that simulates perfectly, whatever p_f, a q-state CA.

CO-EVOLUTION OF SYNCHRONIZING CELLULAR AUTOMATA

The evolution of binary CAs does not produce very good results on real asynchrony. In the previous section, a simple method to deal perfectly with full asynchrony was designed using $3 * q^2$ states. However if we consider a task like the synchronization task, we have a perfect example of a lossy task, i.e., a task where there is no need to maintain absolutely the full information present in the synchronous case to solve the problem in the asynchronous case. The question is then to find the good compromise between the number of states needed, i.e., between the q states necessary in the synchronous case, and the $3 * q^2$ we know to be sufficient to simulate perfectly the synchronous CA in the asynchronous mode. I thus proposed here to try to evolve 4-state CAs, following the cellular programming approach developed by Sipper.

Globally the evolutionary runs are very successful, and if we consider a fitness of 0.98, for $p_f < .01$, as equivalent to a fitness of 1.0 in the synchronous case[1], the success rate is equivalent to the evolution of binary CAs in the synchronous case.

CONCLUDING REMARKS

CA asynchrony was often studied in itself in the past literature and it was often concluded that the global behavior from a CA, the emergent behavior, was an artifact of the global clock. This conclusion was not wrong in itself but rather the wrong standpoint on a reality. Time is part of the visual information contained in a CA. Now if we tackle the asynchrony problem with this idea of restoring all the information, then as we saw, we can easily design a totally asynchronous CA that simulates exactly, with no loss of information, any synchronous CA. However this presents two main problems. First, the required number of states is quite higher the original number of states. Second, it is visually different from the original CA. The visual efficiency of the original CA is lost. Evolution may then be used to limit both these problems. As presented, the cellular programming algorithm was very successful at finding 4-state solutions that were both economic and still visually efficient. Actually, it's all a question of the possible compromise between the information loss and the necessity to maintain that information.[2].

[1]The faults introduce necessarily some cells in the wrong state.

[2]Details on this work may be found in Mathieu S. Capcarrere. *Cellular Automata and Other Cellular Systems: Design & Evolution.* Phd Thesis No 2541, Swiss Federal Institute of Technology, Lausanne, 2002.