Chapter 12
The Universal Constructor

INTRODUCTION

At the beginning of the 1950s, John von Neumann (1966) asked himself whether it is possible to design a machine with the ability to create exact copies of itself which would themselves have the ability to produce new copies. Such a machine would have reproductive capabilities comparable to those we find in biological organisms. In this setting, von Neumann’s goal was to design a Universal Constructor capable of reading the instructions for, and assembling, any machine the designer might seek to build. If the instructions specified a Universal Constructor, the machine would have the ability to build a copy of itself. In other words, it would be able to reproduce. If we wanted the copies of the machine to share this ability, all we would have to do would be to copy the instructions and incorporate them in the new machines.

Von Neumann’s model showed that in principle it is possible to create a self-replicating artificial machine. In the early stages of his work, he proposed a mechanical design. However, the design was not convincing and the mechanical details were so complicated they obscured the replication process. After discussions with Stanley Ulam, the great physicist, von Neumann proposed a model based on Cellular Automata. The model - derived from Turing’s universal computer and including several feedback mechanisms - was still very complex. However, Langton (1984)
was to show that CAs can generate self-replicating structures that are not universal constructors. This proved that the presence of a Universal Constructor is not a necessary condition for self-replication.

From this point on, the real problem was how to build systems that could do more than self-replicate. In 1995, Tempesti (1995) proposed a loop, similar to Langton’s, enhanced to allow the incorporation of an executable program that would be duplicated and incorporated in every new copy. The program is conserved inside the loop where it is connected to the self-replicating segment of the code. Tempesti considered CAs with 10 states and a neighborhood of 9 cells. His work was followed by studies from Perrier, Sipper and Zahnd (1996), who implemented a loop with the ability to implement any possible program, so long as it was written in a very simple language. A further contribution came from Mange et al. et al. (1996), who implemented a loop with the ability to act as a Universal Constructor. Julian & Chua (2002) demonstrated special self-replication capabilities in CAs based on 1-dimensional, additive rules. Wolfram (2002) made a similar proposal, disregarding earlier comments by Langton [1984] who had considered this kind of structure as trivial. In 2002, Bilotta et al. demonstrated the existence of CA’s with self-replicators that join together to form complex structures. The way in which they join is far from arbitrary - each combination obeys specific conditions. Their observed behavior brings to mind, on the one hand, the typical behaviors of additive rules, on the other, the properties of a Universal Constructor. This suggests that the class of self-replicators is far larger than previously thought. In the previous chapter, we also showed how it is possible to increase the complexity of initially simple self-replicators. This suggests that the division between trivial and non-trivial replicators is arbitrary, both being based on the same underlying processes. In this chapter, we return briefly to our discussion of additive rules, showing how we can extend our treatment to other rules (such as ECA90), which can act as templates for complex self-replicators. During the discussion we will see how self-replicating structures can couple to produce more complex systems which are themselves self-replicators. We will see, furthermore that this ability is a property, not just of the rule presented in (Bilotta et al., 2002), but of many rules producing self-replicating structures. These observations bring to mind a form of additivity.

**BACKGROUND**

**Additivity**

Certain CAs display *additivity*. One of these is ECA 90, a CA whose fundamental role we have already discussed in earlier chapters. Given two initial configurations $P_1(0)$ and $P_2(0)$, a CA is said to be additive if developing a superimposition of the
Feature Selection for Bankruptcy Prediction: A Multi-Objective Optimization Approach