Jin Huai Xuan, Kent Feng

Professor Alexey Nikolaev

CSCI 49201

5 May 2019

Cellular Automation Using OCaml

The concept of automated cellular behavior is interesting because of how simple rules could be applied to any model to develop unique and characteristic structures with each subsequent application. Cellular automatons were studied as early as the 1950s as a possible model for biological systems such as gas behavior, forest fires, and even life itself. There are many components to creating a cellular automaton and we utilized much of what we learned in class to accomplish our goals in this project. The initial scope of our project was to create a cellular automaton using OCaml that could replicate John Conway's game of life as a starting point and then move onto replicating several other cellular automaton formats if time permits. Our proof of concept was that since cellular automatons are simply models of simple mathematical systems and logical based rules, it would be just as easy to create cellular automatons in OCaml as in any other high level programming languages.

A cellular automaton program requires many components such as an original seed, a set of families of rules to follow, and methods of output and representation. We were confident that we would be able to implement most components of the program by ourselves without borrowing code from others although we did require the assistance of external libraries such as map and imagelib. The seed was simple enough to make since it was only a text file that we provided as the input file. We used the map library to allow us to map out our cellular automata on a two dimensional plane which serves as our method of representation. We then used

imagelib library to allow us to print the dimensional planes onto PNG files effectively serving as our method of output. Overall, we used many OCaml programming techniques such as currying, recursions, iterators, file streams, modules, and much more. The first version of the program was an one-dimensional cellular automaton in the S/B family format. This version was very primitive meaning that there was no custom input on execution and the rules were strictly 23/3 in the S/B format, also known as the rules for Conway's game of life. Our intentions were to make sure that we understood the mechanics of a cellular automaton in the first dimension before we move up to the second dimension. Our second dimensional cellular automaton was much easier once we had studied the automaton in the first dimension. On our first few iterations of the 2D cellular automata, we implemented the feature of custom user input for the number of generations the program outputs. Then on the next few iterations of our program, we modularized the program separating and began organizing the program into a main file followed by formats contained in their own modules.

Consequently, we also implemented a feature for custom user input for the choice of format and the rule for each format. Each format had its own set of generation functions and rule parsers as required. On our latest iteration of the project, our cellular automaton program now includes a variety of user input options and supports two families of rules, the S/B format and the Margulos format. We have optimized the experience of using the program to allow the best of convenience for our users. In addition we have included some fun features to bring color and variety to the program.

Cellular automation is a concept worthy of discovery for all programmers in the area of physical and biological systems because of its seemingly infinite possible ways of representation and flexibility. This kind of computer simulation is a great way for beginner and advanced programmers to explore the field of simulation without much commitment. It is simple

to implement but in no way are the results simple. They can produce some surprising results and may lead to ground breaking discoveries. There are many more families of cellular generation and thus there are many more rules that are possible to generate unique patterns and results. Our program is far from being a complete and finished cellular automaton simulator but it is the perfect beginning of one.