Algorithmically Coarsening Computationally Irreducible Complex Systems

Ian J. Klasky
Bowdoin College,
Brunswick, ME 04011

Joshua A. Grochow
Santa Fe Institute,
Santa Fe, NM 87501
(Dated: November 25, 2016)

In this study, we developed an algorithm to determine coarsenings for cellular automata (CA) rules. The algorithmic coarsening of systems that exhibit complex behavior, such as CA, can be used to drastically reduce the computing power and time required to model these systems. These computationally simpler coarse-grained models could be used to accurately determine macroscopic properties of the fine-grained system.

One way to learn macroscopic properties of complex systems is to make a model of that system, and run a simulation using that model. The data from this simulation would give details of the systems microscopic and macroscopic behavior. However, if a model is too complicated, or the simulation requires too much time to run this strategy becomes computationally unfeasible. If we were able to algorithmically determine how to simplify, or coarsen our initial fine-grained model to preserve the macroscopic properties of the system, we would be able to learn about the complex system where we wouldn't have been able to before.

In this study we develop an algorithm to coarse-grain systems of one-dimensional cellular automata (CA). CA are discrete complex systems that have been used to model fluid flow, forest fires, and biological processes. Being able to algorithmically coarse-grain high-dimensional CA with cells of many values could allow us to coarse-grain models that accurately represent real-world systems. The first step in this direction is to derive an algorithm that can coarse-grain one-dimensional CA with cells that can contain one of two states; 0 or 1.

In the paper "Computational Irreducibility and the Predictability of Complex Physical Systems"[1], the authors developed a framework for coarse-graining CA rules into other CA rules. Figure 1 shows a fine-grained CA rule evolved over 60 time steps and using 120 initial condition cells. Figure 2 shows a coarse-grained CA rule evolved over only 30 time steps and using 60 initial condition cells. We say that rule B is a coarsening of size 2 of rule A because it cuts down both the required number of time steps and initial conditions by a factor of 2 to represent similar macroscopic properties. We can also say one cell in coarse-grain rule B represents 2 adjacent cells from rule A. Each cell in rule B is thus determined by some function applied to the 2 cells in rule A that it was mapped from. We call this function $P(x)$, and its general form for a coarsening of size 2 is:

$$P(x) = x_0 + x_1 a + x_2 b + x_3 ab, \tag{1}$$

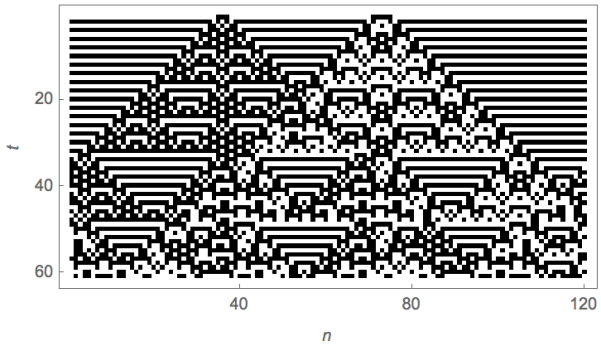

Figure 1. Rule 105 is shown here with 120 cells being evolved at each time step, over 60 time steps. Time is represented as running from the top of the image to the bottom.

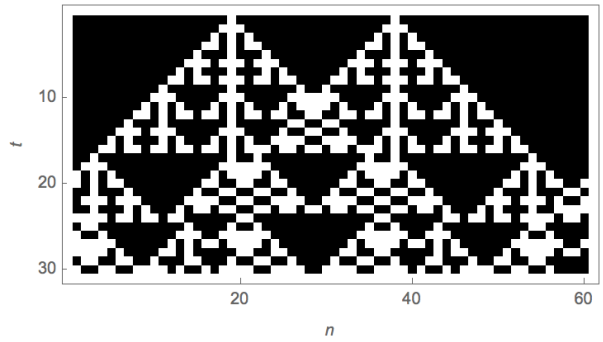

Figure 2. Rule 150 is shown here with 60 cells being evolved at each time step, over 30 time steps.

where a and b are the values of the 2 cells that are being mapped into rule B. The family of functions P(x) for coarsenings of size 2 are characterized by their values of x_0, x_1, x_2, x_3; each of which can take either the value 0 or 1. For any rule B to be considered a viable coarse-grain model of any fine-grain rule A, there must exist some function P that can map from A to B.

Figure 3 represents a fine-grained rule A being coarse-grained by a function $P(x)$ into a rule B. This diagram

shows every two consecutive pairs of initial condition cells in the fine-grained CA being mapped to one cell in the coarse-grain initial conditions. The lower part of the diagram represents the evolution of the given initial conditions in the coarse-grain rule A after 2 time steps. We can see that this results in 2 cells. These 2 cells can then be mapped to the single cell that results from the evolution of the coarse-grain rule B after only one time step. We can view the coarse-graining function $P(x)$ as a horizontal coarsening of the fine-grained rule. A coarse-graining of 2 requires every 2 generations after the initial condition to be mapped to each consecutive generation in the coarse-grain rule B. This vertically coarsens rule B by the same factor as its horizontal coarsening.

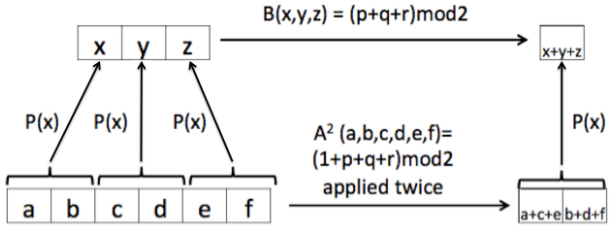

Figure 3. Running the fine-grained rule N times and then applying the coarsening function P(x) yields the same result as applying the coarsening rule 3 times and then running the coarse-grained rule only once. The latter approach is much more computationally efficient as the size of N increases.

In the same paper, the authors checked if there could be a function $P(x)$ that would allow a rule B be to be a coarse-grain version of a rule A by searching through every possible function $P(x)$ that could be mapped between the two. They needed to find a $P(x)$ that would satisfy the equation

$$B(P^N(S_A^{3N})) = P(A^N(S_A^{3N})), \tag{2}$$

where A is the fine-grain CA, B is the coarse-grain CA, P is the coarsening function, N is the coarsening size, and S_A^{3N} represents the set of all possible initial conditions for which the equation needs to hold. The right hand side of this equation represents first running the fine-grained CA N-times and then coarsening. The left hand side represents first coarsening and then running the coarse-grain rule once. This brute force algorithm runs in the time complexity order $O(2^{2^N})$, where N is the size of the coarsening that is being verified. Developing an efficient algorithm to find if coarsenings exist between any two CA would allow us to search for coarsenings in more coarsening sizes.

In order to algorithmically determine if there exists a coarsening function P between two CA A and B, we start with the general equation (2). We then substitute in the specific coarsening size, the rule A, and the rule B that we want to determine if there exists a coarsening to. Using our previous example this gives us

$$x_0+x_1(a+c+e)+x_2(b+d+f)+x_3(a+c+e)(b+d+f) =$$

$$x_0+x_1a+x_2b+x_3ab+x_1c+x_2d+x_3cd+x_1e+x_2f+x_3ef. \tag{3}$$

We now have 1 equation with 4 variables. It's important to note that for every term on the left hand side of the equation that contains a specific set of initial conditions we need a term on the right hand side that contains the exact same set of initial conditions. If we have a term that contains a group of initial conditions that isn't included on the opposite side, there could be no way for the two sides to be equal, since no number of different terms would be able to make up the difference of not having that specific combination. This would mean that there could exist no coarsening function $P(x)$ that would allow there to be a coarsening from A to B. This indicates that there must be the same number of each term of initial conditions on the left side as there are on the right side of the equation for any $P(x)$ to exist. In our example equation, there are 16 terms that contain combinations of initial conditions on the left hand side, and 10 terms on the right. This gives us 16 equations, 4 of which are independent. We can now solve for the 4 variables x_0, x_1, x_2, x_3 that define $P(x)$.

We find that for this particular fine-grain rule A, coarse-grain rule B, and coarsening size N, x_0 is free, x_1 is free, x_2 is free, and $x_3 = 0$. If there exists a contradiction within the variables no coarsening from A to B can exist. If all variables (besides x_0) are forced to equal 0, only trivial coarsenings exist. These trivial coarsenings send all combinations of input conditions to 0 all the time, or 1 all the time, and therefore are useless in coarsening non-trivial CA. If the variables can take on other values, we can say non-trivial coarsenings exist. In our example we have three variables that can take on a non-zero value. This indicates that our particular CA rule B is a viable coarsening of our fine-grain rule A.

This algorithm functions the same way for larger coarsening sizes. It uses N, rule A, and rule B, to turn each resulting equation into a 2^N by 2^N matrix to solve for each of the 2^N variables in $P(x)$. Since square matrices can be solved in $O(n^3)$, where n is both the number of rows and columns in the matrix, we can say that our algorithm runs in 2^{3N} for our example rule B of degree 1, which is a large increase in efficiency from the brute force algorithm's 2^{2^N} running time.

We were also able to find relationships between certain CA rules that we could use to prove there could be no coarsening between them for any coarsening size N. The relationship we found that ruled out the greatest number of coarsenings was that no fine-grain CA that sent the values 000 to 0 could be coarse grained to any rule that sent 000 to 1 and 111 to 0. Figure 3 illustrates this relationship. We can see that no matter the coarsening

size N, after N iterations the fine-grained CA will result in all zeros from the initial condition zero. We also see that all three coarse-grain initial conditions will all be 0 or all be 1. Coarse-grain rule B would send all 0's to 1 and all 1's to zero. Therefore, there can exist no coarsening function $P(x)$ that sends all 0's to both 0 and 1. This relationship rules out an eigth of all possible coarsening pairs. There exist other similar relationships that rule out a fewer number of potential coarsenings across coarsening sizes N. If many or all of these relationships could be unified in an algorithm, the search time for possible coarsenings would be cut down drastically.

The algorithm we present decreases the time required to search for coarsenings between CA. This allows us to determine whether a CA rule can be coarsened to another rule in higher coarsening sizes than were previously possible. Future work on this topic could include using the relationships between CA to increase search efficiency. Algorithms that coarse-grained CA with cells that could contain multiple states would also be interesting. The more efficient these algorithms become the more practical it will be to coarsen high dimensional CA with many values per cell. This will increase the potential for algorithmic coarse-graining to be used in modeling real-world complex system.

[1] N. Israeli and N. Goldenfeld, Phys. Rev. Lett. 92, 074105 (2004).