**An Initial Survey of Causal Networks of Sequential Substitution Systems**

**Amy Joy Beard, Kenneth E. Caviness (\*) and Breanna Bowden-Green**

Physics and Engineering Department

Southern Adventist University

Collegedale, TN 37315

caviness@southern.edu

(\* corresponding author)

**Abstract**

We survey the various types of causal networks generated by sequential substitution systems, including a method for determining their dimensionality. Networks of exact dimensionality of one and two have been found, as well as exponentially growing networks, while only approximations of three-dimensional networks have been discovered thus far.

1. INTRODUCTION

*Mathematica,* the computational software program created by Stephen Wolfram and developed by Wolfram Research, is well known around the world for its mathematical, graphical and symbolic manipulation tools. What is not as widely known is that *Mathematica* was developed for the purpose of exploring what Stephen Wolfram calls a “new kind of science” (*NKS*) [1]. Historically science has proceeded by the construction of theories to explain experimental data and by the design and execution of experiments to test theoretical models. It is perhaps no surprise that the most successful models involve mathematically tractable cases, explaining the phenomena primarily using continuous functions. The *NKS* approach is to observe the behavior of a large set of “simple programs”, noticing characteristics which resemble phenomena in the natural world, perhaps ultimately providing insights about the real universe. According to Gray [2], Wolfram suggests that time and space are discrete, making it difficult to model phenomena by continuous models, and therefore large classes of problems have been neglected by mainstream science. In contrast, *NKS* allows treatment of both simple and complex behavior, generated by simple rules, such as those of cellular automata. One of the most important insights of *NKS* is the creation of an enumeration of all cellular automata, allowing the sequential generation and visual inspection of all possible “simple programs” of this type.

The present research is the same spirit, based on the short introduction to sequential substitution systems and causal networks in Wolfram’s *A New Kind of Science* [1] and initial programs and visualizations developed by one of us [Caviness] during the 2009 NKS Summer School program in Pisa [3], iterating through a complete enumeration of all rule sets of sequential substitution systems [5]. The causal networks associated with the action of simple programs (such as sequential substitution systems, cellular automata or Turing machines) are of particular interest. Causal networks are defined by Wolfram as “acyclic graph[s] arising from the evolution of a substitution system.” [5] These networks are the models which may contain features mirroring the real world.

The focus of our research is the dimensionality of causal networks produced by sequential substitution systems. In this paper, we review the operation of sequential substitution systems and show how the corresponding causal network is generated. We then discuss the use of the previously mentioned enumeration of all sequential substitution rule sets, and the types of causal networks that have been found. Finally, we will discuss what we mean by dimensionality and what methods we have developed to determine this characteristic for any given causal network. The backbone of all this is the sequential substitution system.

1. SEQUENTIAL SUBSTITUTION SYSTEMS

A sequential substitution system (SSS) is a rule-based process in which a *state string* of arbitrary finite length containing characters from an arbitrary (but finite or countably infinite) alphabet is scanned from left to right, in search of a specific pattern given as the left-hand side of a rule, and if the pattern is found, the indicated substitution (the right-hand side of the rule) is made. If the rule set includes multiple rules, the rules are applied in order. That is, we start by scanning the string from left to right, attempting to apply the first rule. If the pattern for the first rule is matched, then the corresponding pattern is substituted for the original pattern at the first such possible position, producing a new state string. Then the process is repeated, still using the first rule. Only if no match for the first rule exists in the state string is the second rule used, and only if both first and second rules fail is the third applied, and so on [1]. Each new state string represents one substitution that has been applied to the previous one. This sequential process may continue indefinitely or it may stop if none of the substitution rules in the rule set match anywhere in the state string. An example of a SSS is shown in Figure 1.

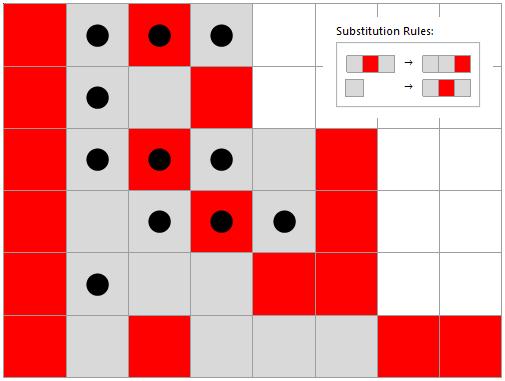


Figure 1 Five iterations of SSS 35,521,928, rule set {“ABA”->”AAB”, “A”->”ABA”} with the initial string “BABA”. The bold black dots represent matches to a specific rule which is then replaced by the substitute on the next line.

We will use this example to describe how a SSS evolves, starting from an initial state string, the initial condition of the system, whose evolution is completely determined by the rule set and the initial state string. The particular SSS shown in Figure 1 has SSS rule set {“ABA”->”AAB”, “A”->”ABA”} (also shown visually with colored squares, where Gray and Red represent “A” and “B”, respectively) and initial state string “BABA” or {Red, Gray, Red, Gray}, as shown in the top line. The first step is to scan the initial string from left to right looking for a match for the left-hand side of first rule (“ABA”). The three black dots on the first row indicate the first (and here, the only) match found. The substitution is then made by replacing the {Gray, Red, Gray} portion with {Gray, Gray, Red}. The process then continues with the string on the second row, but here no match is found for the first rule, so the second rule is applied, and a single black dot indicates the first successful match, the first Gray cell which is in the following line replaced by {Gray, Red, Gray}. This process is continued indefinitely or until there are none of the rules in the set can be applied anywhere. In this particular case we see that each application of the second rule causes the state string to grow, opening up the possibility of increased complexity of the system.

It is worth noting that any SSS having a rule set not allowing this potential for state string growth must inevitably die out or begin to repeat. This can be proven by simple considerations: Suppose a rule set has only rules that maintain or shrink the length of the state string. First note that the initial state string has finite length, and whenever a “shrinking rule” is applied, that length decreases, down to a minimum length of zero. Therefore at some point no shrinking rules will match, and either the SSS dies out, or only length-maintaining rules will be applied from that point on, that is, only a subset of the rule set will be active. Now suppose that this latter situation obtains, that beginning with some state string having finite length *n*,only length-maintaining rules match and are applied. If *k* different characters appear in the state string and in the strings of the rule set, then there are only *kn* different strings of length *n* that the system could possibly generate, and therefore either the SSS will die out or within a finite number of steps a repetition will occur: a state string will be generated that has already appeared earlier in the SSS evolution, and from that point on, the system is strictly repetitive. Since we exclude from consideration any SSS that dies out, it follows that any rule set containing only shrinking rules can be immediately ruled out, and further, that any rule set containing at least one shrinking rule and no growing rules can be excluded, since it must at some point begin behaving like the simpler rule set from which the shrinking rules have been removed. This is an example of one way that certain rule sets can be excluded and the iteration through the enumeration of all rule sets accelerated; other ways will be introduced later, in section IV.

1. DERIVATION OF A CAUSAL NETWORK FROM A SEQUENTIAL SUBSTITUTION SYSTEM

The generation of a causal network from a SSS is detailed in Wolfram’s *NKS* book [1], but will be reviewed here. Every step in the “evolution” of a SSS consists of a substitution which creates and/or destroys cells (the individual characters of the state string). Each node of the network represents one such substitution event. The connections represent the “life” of a cell, beginning with the node that creates the cell and ending at the node where the cell is destroyed. Wolfram also describes a tagging system, which allows us to keep track of cell creations and destructions and to create the connections of the causal network. We assign as tags unique numbers naming each cell, beginning with the initial string and numbering the characters from left to right. When a match is found, the matched cells are destroyed and replaced as specified in the rule applied by new cells, each being assigned a new tag, counting up from the largest previous tag. Cells replaced by the same type of cell are *not* considered to be the same and each receives a new tag. Figure 2 reveals the internal tagging used for the same SSS shown in Figure 1.

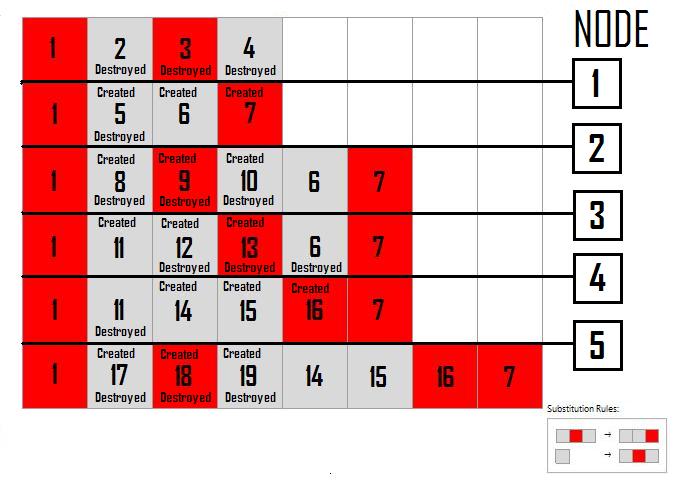


Figure 2 A representation of the tagging system, node occurrences and the birth and death of individual cells which represent connections for 5 iterations of SSS 35,521,928, rule set {“ABA”->“AAB”, “A”->“ABA”} with the initial string “BABA”.

The numbers in the center of each cell are the internally used tags. When the first match is made by the first rule, the tags 2, 3, and 4 are destroyed and replaced by the tags 5, 6, and 7. Nodes represent substitution events. In event 1, three cells are destroyed and three cells are created. An arrow will be drawn in the network from node 1 to any node representing an event that destroys any cell which was created in node 1. For example, cell 5 was created by event 1 and destroyed by event 2, while cell 6 was created in event 1 and destroyed in event 4. In Figure 3, we see that there is a connection from node 1 to node 2 and from node 1 to node 4 and that each connection is labeled by the tag of the cell which was created and destroyed by the events concerned. In this SSS there are at most three direct connections possible from one node to other nodes, since a maximum of three cells are created by any event, as can be seen from the lengths of the strings in the rule set. For example, event 1 only created three cells. Cell 7 was created in by event 1 but is not destroyed in the five iterations of the network shown in the figure. It is possible that a cell could be created by a relatively early node and destroyed hundreds of steps later, but the nature of the SSS may prevent the destruction of certain cells from ever occurring. Figure 3 shows the causal network of the first five iterations of the same SSS shown in Figure 2, and Figure 4 continues the network 29 and then 234 steps, no longer showing the cell tags or arrows on the graph edges.

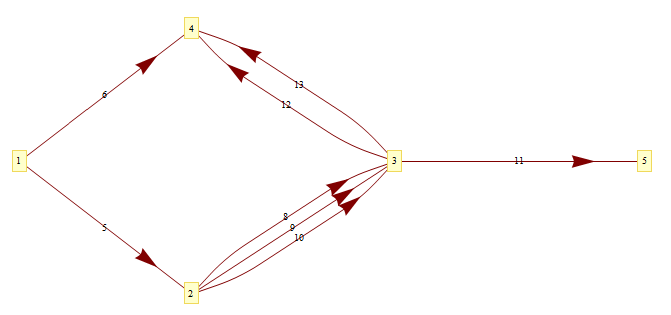


Figure 3 The network generated from five iterations of SSS 35,521,928, rule set {“ABA”->“AAB”, “A”->“ABA”} with the initial string “BABA” where each connection is labeled according to the tagging system of cells created and destroyed.

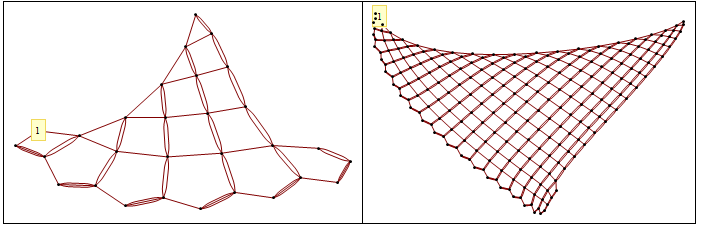


Figure 4 The same network (SSS 35,521,928, rule set {“ABA”->“AAB”, “A”->“ABA”}, initial string “BABA”)), with only the first node identified, for 29 and for 234 iterations.

Given a specified rule set and initial state this process uniquely determines a causal network, but it should be noted that the network created is not unique to the SSS. There are many instances of two different sequential substitution systems creating the same network, as will be discussed in Section IV. But unless it repeats in a regular way, even the same causal network can be made to appear somewhat different by lopping off some finite number of nodes, starting the network at a later node. This can often be done so as to simplify the pattern of the network, particularly the edges: the interior structure is generally unchanged, but perhaps the beginning of the network is regularized, or some initial links prior to the establishment of the “real pattern” of the network are removed. In the case above, the network is simplified by starting at node 3 (dropping the first quadrilateral and beginning with the first pentagon of the network), equivalent to using the third line of the SSS, “BABAAB”, as the initial string. A little experimentation, shows that for any sufficiently complicated initial string of the letters and A and B gives the same pattern, and we find that “ABAA” is the initial state that gives the simplest SSS (with minimum unused cells) while still generating the full regular network.

In addition, the causal networks have the property of progressing forward in time, with connections always starting at an earlier (lower-numbered) event and ending at a later event. The nature of causal networks guarantees that a cell can never be destroyed before it is created. Thus we always know the direction of a connection and will often leave out the directional arrows in our figures. It should also be noted that any cells in the visualization of the SSS which are never destroyed don’t affect the network, so the causal network omitting irrelevant information, abstracting out fundamental features and behaviors of the SSS. We now turn to the analysis of the causal networks generated by sequential substitution systems.

1. GENERATING ALL DIFFERENT CAUSAL NETWORKS OF SEQUENTIAL SUBSTITUTION SYSTEMS

The number of possible sets of rule sets for sequential substitution systems is infinite. We can increase the number of cell types (the size of the alphabet), the number of cells used in the rule set (basically the length of the strings), and the number of rules in the rule set. Fortunately, this set is countable, enumerable. In order to provide a systematic method of analyzing the sequential substitution systems without leaving any out, an enumeration of these rule sets was needed. The enumeration used in the present work was created by one of us [Caviness] and detailed in a previous paper [5], to which the interested reader is directed. Without going into detail, it should be noted that in this enumeration, all rule sets of “weight” *n* appear before the rule set of weight *n*+1, where the weight of a character is given by “A”->1, “B”->2, “C”->3, …, and rule set weight is defined as the sum of the weights of all the characters in the strings in the rule set. Any rule set containing a “”->”” rule is discarded, so the first rule sets to appear in the enumeration are the two which have weight 1: {“”‑>“A”} and {“A”->“”}, and the nine rule sets of weight 2: {“”->“A”, “”->“A”}, {“”->“A”, “A”->“”}, {“A”->“”, “”->“A”}, {“A”->“”, “A”‑>“”} , {“A”->“A”}, {“”->“AA”} , {“AA”->“”} , {“”->“B”} , {“B”->“”}, where those grayed out can be eliminated from consideration by methods discussed later in this section. It suffices here to repeat that smaller weight rule sets appear in the enumeration before larger weight rule sets, and to note that the empty string is considered here to be a valid match or replacement string, corresponding to an insertion or deletion, respectively. This extension beyond the allowed SSS rules described by Wolfram [1] we justify on the basis of the existence of similar already present rules: there are substitutions that create a different number of cells than they create, effectively growing or shrinking the state string. Allowing “nothing-to-something” or “something-to-nothing”, pure creation or annihilation events, simply includes simpler examples of already included cases. The generalization is also consistent with the effort to include all possible meaningful SSS rule sets, to “not leave anything out”.

Each SSS is initiated with a sufficiently complicated string which is guaranteed to have at least a few matches. In iterating through the enumeration, one notices that the majority of possible sequential substitution systems simply “die” out. That is, the nature of the rule sets is such that eventually the rules are no longer be applicable, no matter what initial state is chosen. Although all possible rule sets appear somewhere in the enumeration, twenty-five percent of the rule sets generated by the algorithm are duplicates and are immediately discarded, as described in [5]. Others can be eliminated from consideration without actually following the evolution of the SSS and creating the associated causal network. In addition to rule sets producing exact SSS duplicates (generally because some rules never were applied), we have found many rule sets that produce the same causal network even though the SSS visualizations are different. Example duplicate network rule sets are given for each rule set we use in this paper in Table 1.

Table 1 Every causal network can be produced by an infinite number of different rule sets. The rule sets used in the paper are given with the corresponding enumeration index along with an example of an alternative rule set producing the same network.

|  |  |  |
| --- | --- | --- |
| Index | SSS Rule Set | Alternate SSS Rule Set |
| 35,521,928 | {“ABA”->“AAB”, “A”->“ABA” } | { “BAB”->“BBA”, “B”->“BAB” } |
| 3 | { “A”->””,””->“A” } | { “A”->“B”, “”->“BA” } |
| 6 | { “A”->“A” } | { “A”->“BA”,””->“A” } |
| 137,679 | { “AB”->“BA”, “A”->””, ””->“AA” } | { “AB”->“BA”, “A”->””, ””->“AAA” } |
| 137,703 | { “AB”->“BA” , ””->“BA” } | { “BA”->“AB” , “”->“AB” } |
| 33,157 | { “AAB”->“BA”, “”->“A” } | { “AAA”->“AAB”, ””->“A” } |
| 135,400,946,080 | { “AAA”->“AB”, “BB”->“BA”, “B”->“BAA” } | { “BBB”->“BA”, “AA”->“AB”,“A”->“ABB” } |
| 530,557 | { “AAB”->“ABAA”, “”->“A” } | { “AAB”->“ABAA”, “”->“AA” } |
| 43,052,276,128 | { “BAB”->“ABA”, “A”->“B”, “B”->“BAA” } | { “ABA”->“BAB”, “B”->“A”, “A”->“ABB” } |
| 2,123,767 | { “AAB”->“BAAA”, “”->“AA” } | { “AAB”->“BAAA”, ””->“AAA” } |
| 2,203,521 | { “AB”->“BAA”, “”->“AAB”} | { “BA”->“ABB”, “”->“BBA” } |

Reference was made earlier to the fact that “shrinking” rule sets could be safely excluded, since the SSS either will die out or will be a duplicate of that of a simpler rule set (that obtained when the shrinking rules have been removed). Another type of case that can be safely excluded is that a “non-solo identity rule”. An identity rule is a rule in which a match is replaced by itself, such as “AB”->“AB”. If the rule is ever applied, then it will be applied indefinitely, and from that point on the SSS behavior is identical to that of the solo rule, and the causal network will be a chain. On the other hand, if the identity rule is never used, the SSS is identical to that of a previously treated case, that of the simpler rule set formed by removing the identity rule. Any either case the behavior is the same as that of a simpler rule set, already seen earlier in the enumeration, and so we may safely skip a rule set containing an identity rule, unless it is the only rule in the rule set. Similar considerations permit the exclusion of rule sets containing “substring rules”, such as “AB”->”ABC”. Here even the solo rule set can be excluded, since the causal network is identical to that of a simpler, previously seen, solo identity rule, in this case, “AB”->“AB”.

Another instance for which we skip a rule set is when the rules themselves conflict with each other, having the same left-hand sides. For instance, if the rule set contains the rules “A”->“B” and “A”‑>“C”, then the second rule will never be applied since they look for the same match. Similarly, if the left-hand side of one rule is a substring of the left-hand side of a later rule, the second will never be invoked, e.g., {“A”->”B”, “AB”->”B”}.

Others methods used to make our search for unique interesting causal networks more efficient include eliminating rule sets in which the permuting or renaming the characters results in a previously seen case, e.g., {“B”‑>“B”} generates a causal network indistinguishable from that of {“A”‑>“A”}. The greatest acceleration is obtained when a long sequence of enumerated cases can be eliminated and jumped over, because they all share the same problem.

1. TYPES OF CAUSAL NETWORKS AND DIMENSIONALITY

Using the SSS rule set enumeration, we began systematically iterating through and analyzing each causal network beginning with index 1 and continuing through increasing index numbers. Many causal networks, even large blocks of sequential cases, were skipped in the enumeration due to reasons analyzed in the previous section, but this is still an open-ended process, and to date only indices < 1016 and rule set weight 17 have been treated. Increasingly complicated networks were observed, and in all cases once a network appears, duplicates appear from that point on, so methods were sought to identify and catalog new types of networks as they were found, such as disconnected repetitive networks (“clumps”), single- or multi-strand chains, chains with attached clumps, two-dimensional sheets, etc.

What we mean by dimensionality will also be explored in this section from an intuitive standpoint. A straight horizontal line has only one direction in which growth or change takes place, it only extends in a single direction. So we view dimensionality as how many directions a network grows. This can be challenging as we normally view causal networks on a two-dimensional plane and only have the capability of displaying it in three dimensions. In addition, a general sense of what we mean by “dimension” may be accomplished by intuitively analyzing the different casual networks that we found. In our analysis, we found four basic types of nontrivial causal networks.

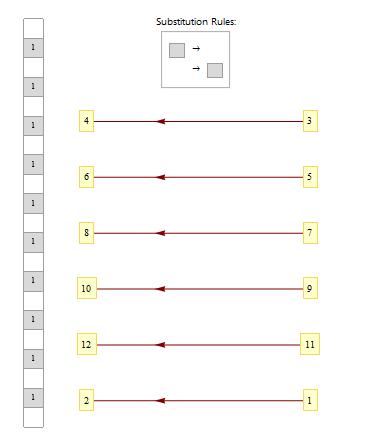


Figure 5 First causal network, SSS 3, rule set {“A”->””,””->“A”} with initial string “”: a detached network

The first type of causal network seen in the enumeration was a detached network shown in Figure 4. This figure shows a periodic detachment which means that a node is not connected in any way with the previous nodes. In this particular example, the detachments occur every two nodes. We define a connected network to be a network that only contains a finite number of detachments and it is well-connected if it contains no detachments. In our definition, we specify a finite number of detachments as there are some networks that may contain a single detachment but afterward seem to grow in a connected way after that detachment. It is difficult in some cases to classify a network as connected or well-connected as we do not always know the end behavior of a causal network. In Figure 4, we can see the cause of the detachments and can prove that the detachments are periodic and thus infinite. We would expect the dimensionality of this network to be zero as it does not seem to grow in any direction.

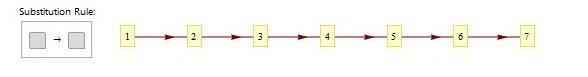


Figure 6 First chain causal network, SSS 6, rule set {“A”->“A”} with initial string “A”.

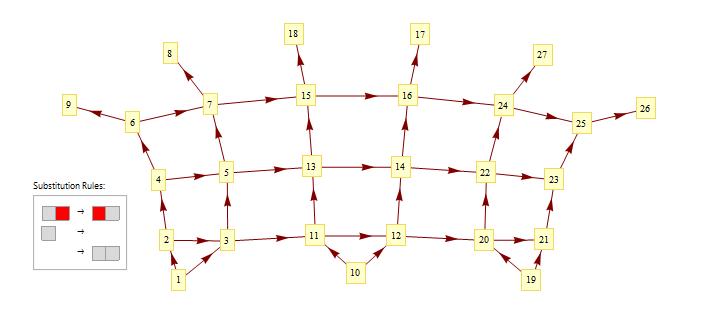
We now turn our attention to well-connected causal networks. The next type of causal network cataloged as we traverse the enumeration was the chain shown in Figure 5. This is number six in the enumeration. This causal network is infinite and grows only in one direction, here shown growing from left to right, and can be considered as one-dimensional. In this example, the nodes are linked together by a single connection. We refer to this as a single-linked chain. If there are two connections, then it is double-linked chain and so on. Anything that has the basic structure of chains with clumps coming off it is also considered a chain. Also, there are bands which are also categorized as one-dimensional. This is when the network grows in one direction but still has more than one node (a finite number) in a perpendicular direction. An example of a one-dimensional band is shown above in Figure 7. This band initially grows in two dimensions but then attains and maintains a constant width of 4.5 in one direction, continuing to grown only one-dimensionally.

Figure 7 Causal network for SSS 137679, rule set {“AB”->”BA”,”A”->””,””->”AA”} with initial string “BBB”, a band.

We also find causal networks in the enumeration that grow in two directions, such as our original example, shown in Figure 4. Upon analysis, we find more rows are being added to the figure as the iterations increase and that each row is increasing in length. This is exactly what we would intuitively call a two-dimensional causal network.

We usually view two-dimensional images of these causal networks, and indeed the method of construction guarantees that a two-dimensional layout with no intersecting edges can always be found, perhaps justifying the intuitive expectation that causal networks of sequential substitution systems may be limited to dimension two or less. Is there any sense in which such a network may be considered to have higher dimensionality? Yes, if the growth occurs at a rate greater than that found in two-dimensional space. An example of this kind of behavior is shown in Figure 8 below. The two-dimensional representation of the network shows the nodes packed closer and closer together as we move away from node 1, and the three-dimensional representation shows these extra nodes being forced up or down out of the plane into ruffles – and still becoming very densely packed, indicating that this the network may be in some sense more than two-dimensional, perhaps even more than three-dimensional. The image of the SSS itself allow us to be confident that this growth will continue without bound, since there is an repeating pattern of increasing length (approximately doubling): whenever the state string consists of red cells followed by gray cells (“BBB…BAA…A”), a new gray cell is inserted at the left and then gradually moved right across the band of red and increasing its width, until it joins a growing sequence of final gray cells on the right, at which point the whole thing repeats. Since each diagonal pass crosses a longer red band, each pass involves more events (nodes in the network), so unbounded growth seems likely.

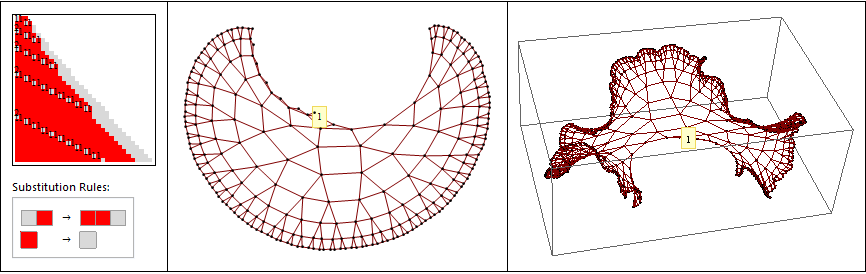


Figure 8 SSS 2 204 134, rule set {"AB" -> "BBA", "B" -> "A"} with initial string “ABB” and its causal network: a “ruffled” network, whose outer boundary we later show to be proportional to 2n.

More cannot be said from a visual inspection of the networks, but [Fig??] shows a variety of examples of the various types mentioned. We now turn to a more rigorous method of specifying the dimensionality of a causal network.

1. Rate of Growth of a SSS as a Method of Determining Dimensionality

Among the many graph-theoretic measures which may be used in the analysis of these causal networks, we considered the rate of network growth as the most likely to provide a measure of its dimensionality and complexity. One promising idea was to compare the average in-degree and average out-degree of the nodes, the number of edges connecting to and from each node. But further consideration shows that the nature of the network is highly dependent on node arrangement, making the degree averages inconclusive for dimensional analysis. Another early attempt at analyzing SSS causal networks focused on the nodes along the two growing sides of causal networks of SSS examples in [1]. We sought patterns in the node numbers of the “inner” and “outer” edges shown in Figure 9, thinking that a method for determining dimension might arise from the inner edge node sequence. Yet unless one continues to evolve the system, it is not immediately apparent which nodes will be part of the permanent inner edge and which will eventually become interior nodes. For example, although it might well be guessed that node 201 is not a part of the inner edge, this is not conclusively demonstrated at the stage of development shown.

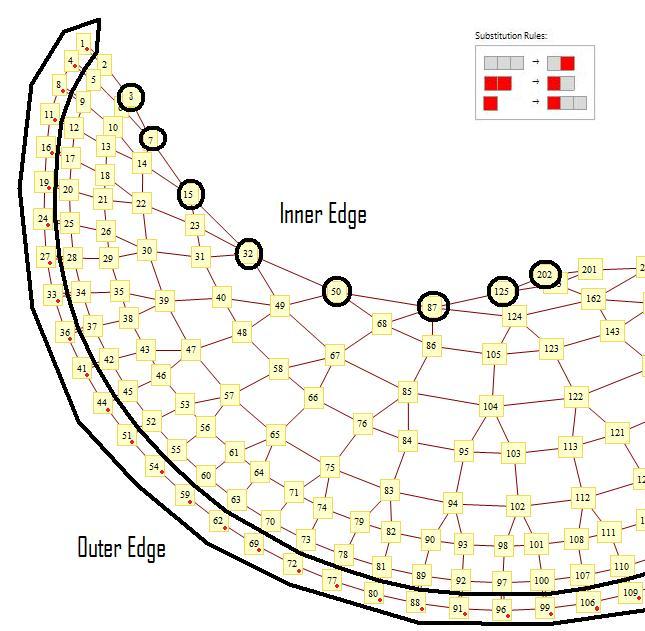


Figure 9 Cropped view of SSS 135,400,946,080, rule set { “AAA”->“AB”, “BB”->“BA”, “B”->“BAA” } with initial string “BABA” over 202 iterations with emphasis on the “inner” and “outer” edges, which grow at different rates.

Once enough edge points had been identified, we used FindSequenceFunction in an attempt to find a pattern that could be used to predict future edge nodes and eventually to derive the dimensionality of the network. The outer and inner edges in this specific example do produce viable formulas *after* the first few nodes in the inner edge and outer edge sequences are dropped, but in practice the whole method was too cumbersome, requiring manual determination of inner edge and outer edge nodes numbers, entailing much time and a high probability of human error. In addition, not every SSS had an easily distinguishable inner or outer edge. A more useable and general way of classifying a sequential substitution system was needed to determine or approximate dimensionality.

Such a method for estimating the dimensionality of a network is described in Chapter 9 of *NKS* [1]. Defining the distance *r* between two nodes of a network as the least number of connections one must traverse in order to move from one to the other, the approximate dimensionality of network can be determined by analyzing the *boundary function* , defined to be the number of nodes that are distance *r* from the origin (node 1 of the network). For example, if there are 3 nodes that are distance 5 away from the origin, we have . The set of all nodes a distance *r* from the origin is in some sense a boundary, corresponding closely to the circumference of a circle of radius *r* or the surface of a sphere of radius *r*, while the set of all nodes which are at most *r* steps from the origin corresponds to the interior – that is, the area of the circle or the volume of the sphere. The circle, a two-dimensional object, has boundary and interior given by and , respectively, while the sphere, a three-dimensional object, has boundary and interior given by and , respectively. That is, the interior of each is proportional to and the boundary to , where *d* is the dimension of the space. In like manner, we take it as indicative that a network has dimensionality *d* if its boundary function has dimension *d*-1, i.e., if . When applied to the various types of SSS causal networks found, this methodology is found to agree with the intuitive assessment of chains and bands being one-dimensional, the various fishnet types being two-dimensional, while the disconnected “clump” causal networks are seen to be in some sense zero-dimensional. The dimension of any network may thus be determined programmatically, by analyzing the boundary function.

A few examples may serve to illustrate the method. First, consider the case of a causal network that is not well-connected, such as the one shown in Figure 5. This type of network has an infinite number of disconnects, so it is not possible to move past an initial region and reach a node connected to all later nodes. In fact, except for a finite number of nodes close to the origin, all later nodes are inaccessible (denoted by our program as being ∞ steps from the origin), and therefore we may write , for all greater than some finite number of steps *n* (here *n* is at most 1, but for more complicated repeating clumps, *n* is at most the clump size). Treating the “interior” of the network as the integral of the boundary, we arrive at a constant size for the interior, a value not proportional to any power of *r*, or more conveniently, proportional to . It follows that we may consider the dimensionality of such networks to be 0. If the network is not periodically detached but has a finite number of detachments, generating the SSS from a different initial string may very well fix the problem. We have found that while any SSS generally behaves in the same way when started from a sufficiently complicated initial string, changing this initial string may reproduce the same network in a different stage that may make it easier to quantify dimensionality.

For the chain example in Figure 6, : there is exactly 1 node *r* steps away from the origin, for each positive integer *r*. In general, we find that for single or multiple-stranded chains or bands, with or without attached periodic clumps, , where *c* some positive constant, or possibly cycles between several constants, but in any case has no *r*-dependence and we may therefore write for all *d* sufficiently far from the origin. Therefore we consider the boundary or “surface” of such networks to have dimension 0 and the interior to have dimension 1, as expected.

The same method confirms the fishnet example in Figure 4 to be two-dimensional. We first characterize each node of the network by its distance (number of steps) from the origin (node 1). Figure 10 shows the network again, now with the adjusted network starting point mentioned earlier.

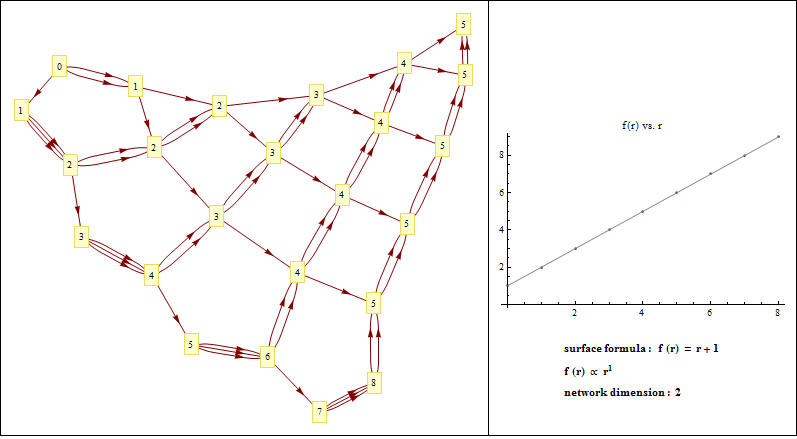


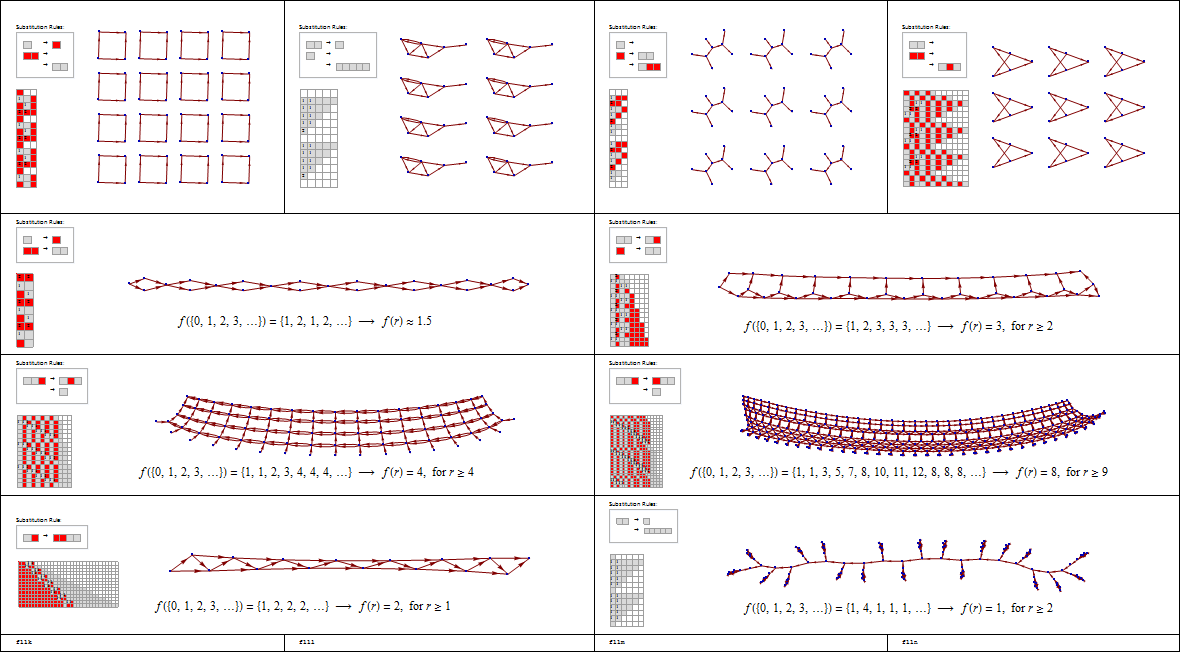
Figure 10 Causal network of SSS 35,521,928, rule set {“ABA”->“AAB”, “A”->“ABA”}, initial string “ABAA”)), 24 steps shown with nodes identified by their distance from the origin. f(r) found proportional to r1, indicating a 2-dimensional network.

Here we find , or for large radii, approximately , and consequently the interior of the network has approximate dimension 2, by this criterion.

As the complexity of the rule set increases, it would not be surprising to find cases of higher dimensionality, and indeed, the first 0-dimensional network occurs at SSS index 3, the first 1-dimensional network occurs at index 6, the first network of dimension higher than 2 occurs at index 33181, and the first 2-dimensional network at index ??. Intuitively one would also expect the dimensionality of Figure 8 and Figure 9 to greater than two, as these networks cannot lie flat in the plane while keeping the spacing between nodes uniform. But actually the boundary function for the network shown in Figure 8 is , for , or for large *r* it is fair to use the approximation, an exponential dependence that implies that the interior of the network grows as (also exponentially), becoming larger than any polynomial in *r*. By our criterion the boundary and the network itself must then be considered to have infinite dimension, or perhaps one might term it a non-constant dimension, increasing without bound as the evolution proceeds.

The actual procedure of determining the number of nodes that are *r* steps away presents practical difficulties. It is important to have generated enough nodes of the network to reach a point for which we adequately trust our data, i.e., we want no new connections made from nodes in the region under consideration to future nodes. An equivalent way of saying this is that any cells created by our subset of events should already have been destroyed or should never be destroyed in the future. Of course, it is impossible to continue the evolution forever to check this. There is a large class of systems in which all cells remain for a long time before eventually being destroyed, so the distance data is reliable only up to the point at which a node is found that has not yet attained its maximum out-degree. But there are other systems that forever insert new cells that will never be destroyed, and in general this is not clear from visual inspection. It is not uncommon to find networks that appear to have one behavior at the beginning, changing at some later point in a fundamental way. A visual inspection at the earlier point may mistakenly identify the network as two-dimensional or exponentially growing, when a longer evolution shows that it degenerates into a simple chain, or even dies out. It is therefore important to treat any patterns found in the causal networks as provisional, until consideration of the underlying SSS assures us that the behavior found will continue indefinitely, as was done in the discussion regarding Figure 8. Increased reliability comes by comparing the current out-degree of each node with its potential out-degree, given by the number of cells created by the event represented, subtracting to find the number of potential remaining connections originating from the node in question. The out-degree data can be trusted completely up to the point where the potential remaining connections is no longer zero, and to a lesser degree as far as the pattern of created but undying cells in the SSS can be used to assure us that no new connections will actually be made.

An even more surprising instance of the failure of human observation to adequately identify and sort these causal networks occurs in the network shown in Figure 9, originally taken from p.



For example, Figure displays a portion of the pattern that occurs for the first seventy-six nodes of a particular SSS. This portion would lead you to believe that it continues on forever as a modified chain. However, Figure shows five hundred iterations which reveal a more complicated structure. This example shows why each SSS should be analyzed individually. In examples such as this, it is more efficient to begin the SSS at the string for which the new pattern emerges. In this case we restart the SSS where the chain pattern ends and the ruffle pattern starts at node one hundred and sixteen. Analyzing this SSS from the new starting point, a clear surface formula appears to be . If this equation was true as d approaches infinity, then we could say that the SSS grows exponentially and has an infinite dimension. However, this pattern is broken after 3000 iterations. This happened because a few nodes which were created at the beginning were destroyed after thousands of iterations. This can be predicted by viewing the pattern of the SSS. Information on all possible connections for a particular node is helpful in trusting the reliability of the data. This is why an algorithm to analyze these is difficult as it requires a detailed analysis of each particular SSS.

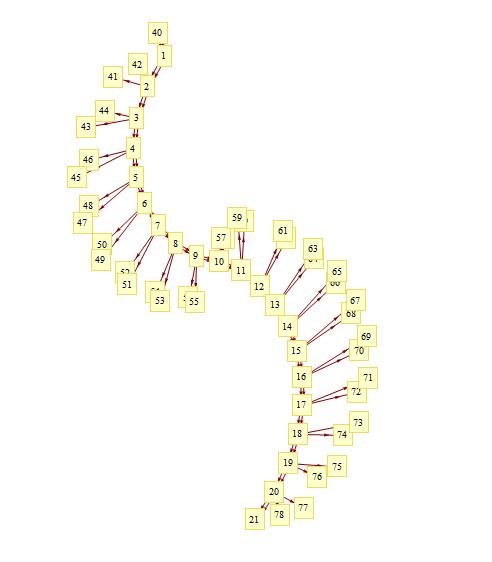


Figure 10 SSS rules set “AAB”->“ABAA” and “”->“A” with initial string “BBABABABABABABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBA”. A portion of this SSS, including 78 iterations

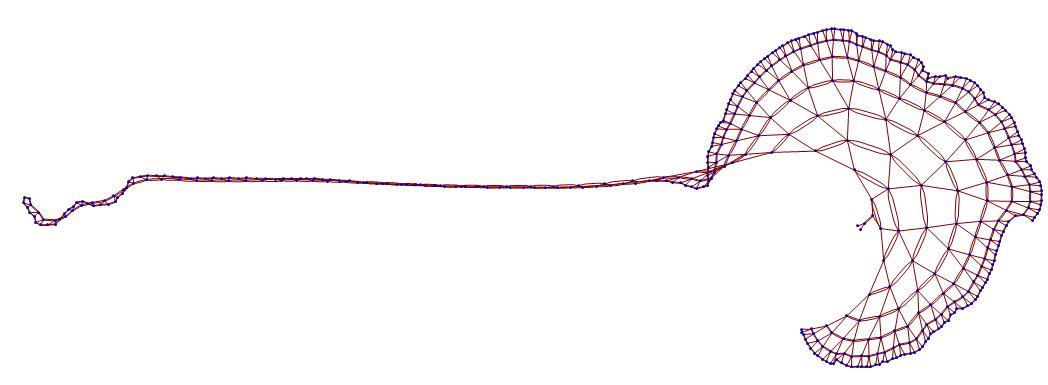


Figure 11 SSS rules set “AAB”->“ABAA” and “”->“A” with initial string “BBABABABABABABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBA”. A portion of this SSS, including 500 iterations.

1. APPROXIMATELY HIGHER THAN TWO DIMENSIONS

Our search for more complex behavior has produced a few examples which appear to have an approximate dimension of more than two. we will show what we mean when we say that we trust data. And introduce a few tools to determine the dimensionality of a given causal network.

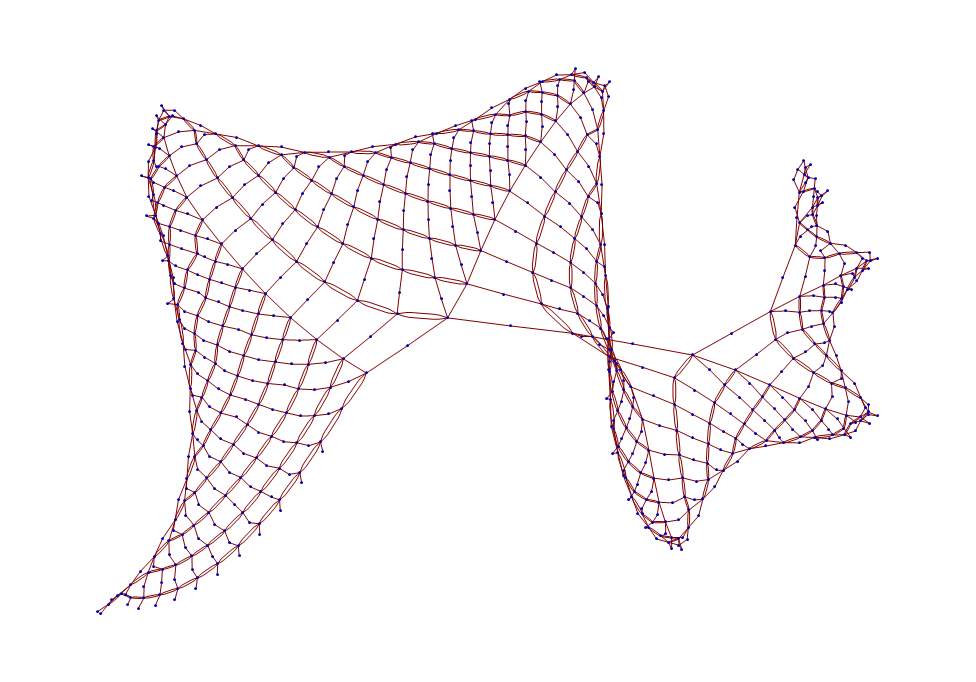
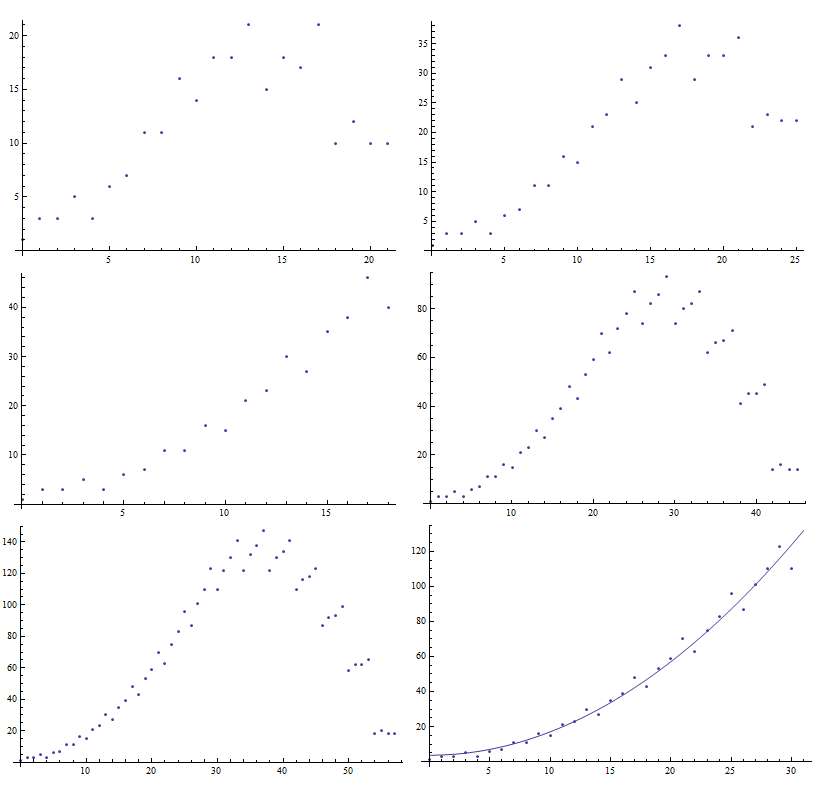


Figure 12 Causal network of SSS rule set {"BAB"->"ABA", "A"->"B", "B"->"BAA"} with initial string "BAB" over 750 iterations.

Visually this network appears to have dimensionality greater than 2, it seems to rise out of the plane into the third dimension, but perhaps not growing exponentially: an excellent candidate for a 3-dimensional network. We proceed to the analysis of dimensionality by generating the surface function. In this process, the evolution of the causal network is important. We can make an educated guess as to how the network will continue to grow by looking at the progression. This can be visualized in Figure . The plot on the top left represents this progression after two hundred and fifty iterations. So far, no pattern seems to be emerging except for a general trend for the points to increase as one traverses the x axis. On the top right, a pattern becomes a bit more distinct. As we traverse through the iterations, we see that some points toward the origin seem to have stabilized while there



4000 Iterationserations

4000 Iterationserations

2000 Iterationserations

1000 Iterationserations

500 Iterationserations

250 Iterations

Figure 13 Progression of surface data for causal network produced by SSS rule set "BAB"->"ABA", "A"->"B", and "B"->"BAA" with initial string "BAB". The number of iteration is doubled every consecutive figure. (Top left) 250 iterations, (Top right) 500 iterations, (Middle left) 1000 iterations, (Middle right) 2000 iterations, (Bottom Left) 4000 iterations, (Bottom right) 4000 iterations with trusted data fitted to the parabola . Note that the scaling in different on both the x and y axis per figure in the grid.

are some trailing terms which have not settled yet. The expectation is that these trailing terms will stabilize as the number of iterations increase. After four thousand iterations, it appeared that all of the points distance thirty away from the origin or less stabilized. We then took this data and tried to fit it to a function using *Mathematica*’s tools. The best fit produced the parabola . It is also possible to find the best fit of a third degree polynomial; however, the coefficient for the highest term was small enough to be negligible. We rule out higher dimension surface equations by analyzing the highest coefficients. Since this the surface function is of degree two for this particular causal network, it can be concluded that the causal network is approximately three-dimensional. We also show another three-dimensional causal network displayed in **Error! Reference source not found.** and the best fit with the surface data in Figure .

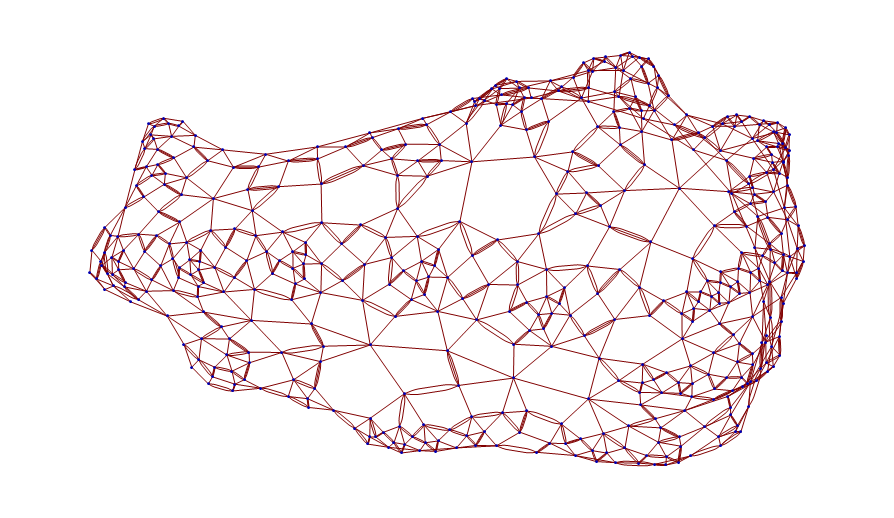


Figure 14 Casual network produced from the SSS rule set “BAA”->“ABBB”, “BABB”->“ABA”, “A”->“AABB” with initial string “BAB” over five hundred iterations.

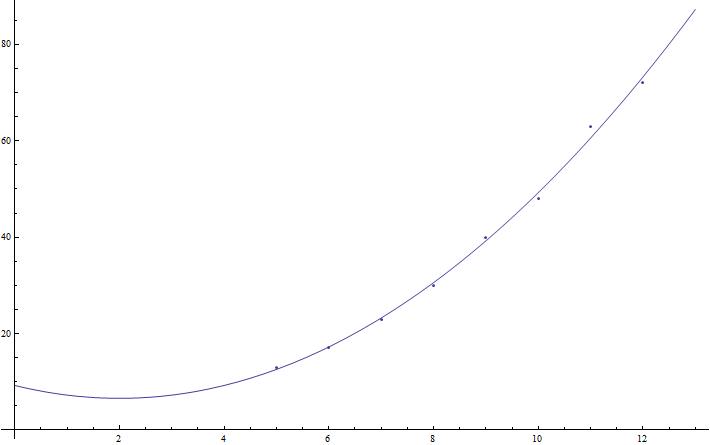


Figure 15 The surface data of the casual network produced from the SSS rule set “BAA”->“ABBB”, “BABB”->“ABA”, “A”->“AABB” with initial string “BAB” over five hundred iterations plotted with . The data was fitted after ignoring the first four terms and securing reliable data.

Our next objective was to determine whether causal networks existed that have an approximate dimension higher than three. While we have not found any causal network with a finite dimension higher than three, we have found two approximately exponential causal networks whose dimension is always increasing. The first is shown in Figure along with its best fit after dropping the first four terms in Figure . The challenge with exponential causal networks is determining whether it is actually exponential or whether it is just a much higher dimension polynomial. This problem is to be expected since an exponential can be written as an infinite series of polynomials [5]. One solution we have found is to plot the surface data with the y axis scaled logarithmically. Linear behavior in the log plot indicates exponential behavior. This is shown in Figure after the first four terms.

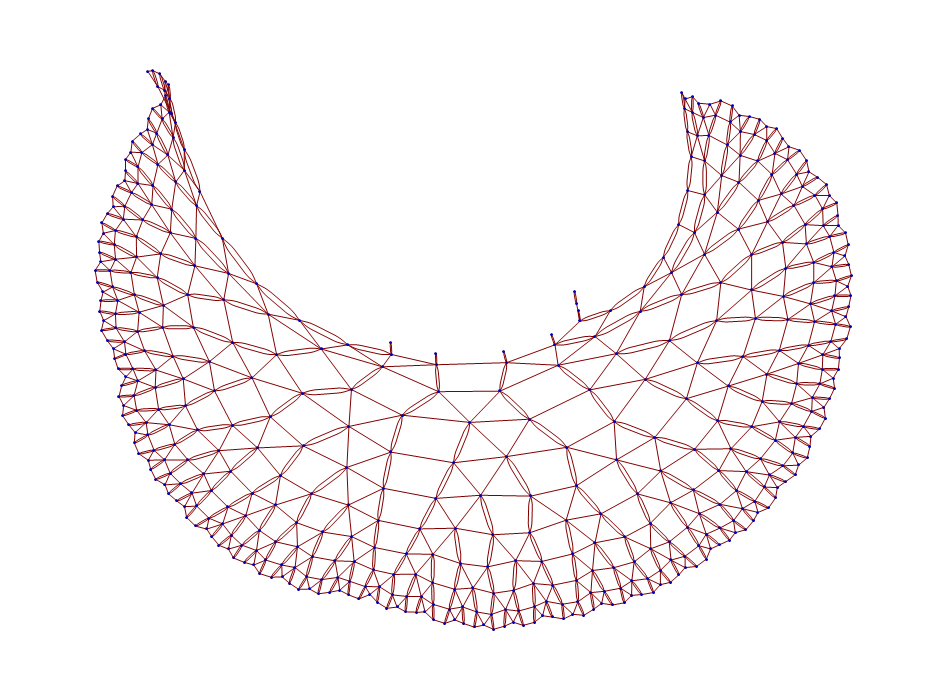


Figure 16 The causal network produced by the SSS rule set “AAB”->“BAAA” and “”->“AA” produced from the string “BBBBBABBABABABAAAAAAAAAAAAAAAAAAAAA” over 500 iterations.

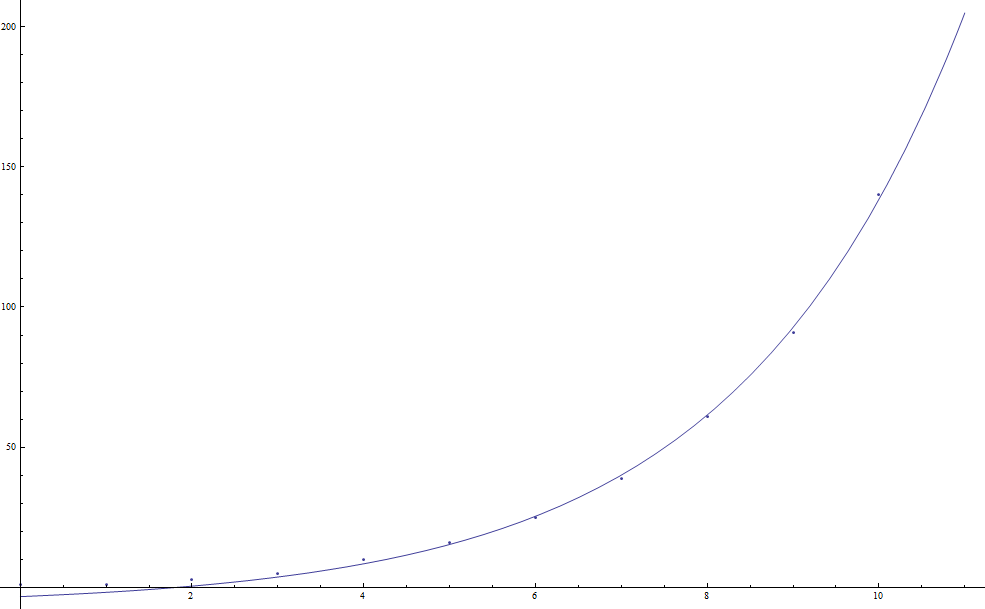


Figure 17 Plot of the surface data for the causal network produced from the SSS rule set “AAB”->“BAAA” and “”->“AA” produced from the string “BBBBBABBABABABAAAAAAAAAAAAAAAAAAAAA” fitted to using data after the first four terms.

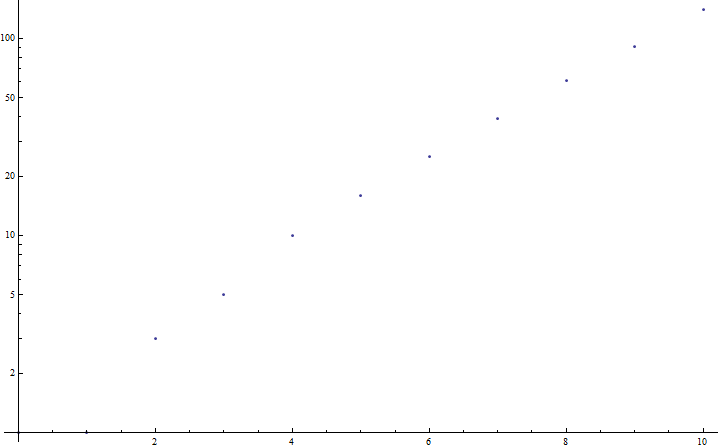


Figure 18 Plot of the surface data for the causal network produced from the SSS rule set “AAB”->“BAAA” and “”->“AA” produced from the string “BBBBBABBABABABAAAAAAAAAAAAAAAAAAAAA” using a logarithmic scale for the y axis.

1. CONCLUSIONS

Causal networks derived from sequential substitution systems have many characteristics to explore. Knowledge of how the causal network is created from a sequential substitution system, an enumeration of all possible combinations of rule sets, and methods to reduce the number of duplicate sequential substitution systems analyzed are all essential in analyzing and determining the dimensionality of a causal network. The dimensionality is determined by creating a function which outputs the number of nodes a certain distance away with the input being the given distance. These formulas may be determined by analyzing each SSS individually to determine the number of iterations needed to create a set of trusted data. Then finally we analyze these functions and determine the dimension of them using normal mathematical procedure of analyzing the highest exponent or pronouncing infinite by finding an exponential. We have been unable to find a causal network generated from an SSS whose dimension exceeds two. However, we have several examples whose dimension is approximately exponential or greater than two. There is still much research to be done in modifying the program, developing new tools to continue analyzing causal networks in the order of the enumeration, and equating causal network to physical phenomena.

Part of our goal is to modify the program in order to skip less interesting causal networks in order to identify the unusual causal networks faster. Methods of how to skip everything that has a dimension of two or less is in progress. This can be hard as not every SSS grows in exactly the same way to produce the simple behavior. Our efforts are focused on analyzing the sequential substitution systems in order to discard uninteresting networks. In addition, there is work to be done in creating an efficient way of skipping more duplicate causal networks without slowing the program down. We want to create more tools that are geared toward analyzing the dimensionality of the causal networks. This may be an ongoing process as more ideas of what would make it easier to analyze the networks.

One modification to the program that we are working on is to generate the information of how many possible connections a given node has as compared with how many already exist. This will tell us how prone the network is to spontaneously changing its pattern. The surface formula is reliable unless a node is randomly generated near the origin. This is another way of measuring how much we trust our data. In addition, we are always searching for more methods of analyzing sophisticated causal networks as we continue with the enumeration. We have encountered a very small number of the different sequential substitution systems that are possible. We will also continue traversing through the enumeration in search of complex causal networks in order to analyze them. The whole essence of this project is to discover causal networks that can be used to explain phenomena in the physical world. We will continue to catalog interesting causal networks in order that someday someone may use them. Actual analyzing of the networks which resemble the real world will forever be the study of anyone involved in a new kind of science.

1. ACKNOWLEDGEMENTS

The authors acknowledge the generous support of the Academic Research Committee of Southern Adventist University for funding this project, and thank Kelsey Dobbs for selection and preliminary analysis of the data, and Chris Hansen, Chair, Physics & Engineering Department, Southern Adventist, for many helpful suggestions.

REFERENCES

1. S. Wolfram, *A New Kind of Science* (Wolfram Media, Champaign, IL, 2002) pp. 1-22, 82-94, 433-503.
2. L. Gray, “A Mathematician Looks at Wolfram’s New Kind of Science”, Notices of the American Mathematical Society **50** (2), 200-211 (2003).
3. K.E. Caviness, "Causal Networks of Sequential Substitution Systems: Views of highly non-local systems", Poster Session and Lecture at Area della Ricerca CNR-Pisa, 2009.
4. Wolfram Research Inc., 100 Trade Center Dr., Champaign, IL 61820-7237,

http://www.wolframalpha.com.

1. K.E. Caviness, “Indexing Strings and Rulesets: An Exploration Leading to an Enumeration”, The Mathematica Journal **13** (2011).
2. J. Stewart, *Calculus: Concepts & Contexts, Third Edition* (Bob Pirtle, 2006) pp 610.