



Stephen Wolfram
A NEW KIND OF SCIENCE
Open Problems and Projects

INCOMPLETE PRELIMINARY VERSION (as of June 26, 2003)

Stephen Wolfram
A NEW KIND OF SCIENCE
Open Problems and Projects

INCOMPLETE PRELIMINARY VERSION (as of June 26, 2003)

Send mail to contact@wolframscience.com for notification
when the complete version is available.

Copyright © 2003 by Stephen Wolfram, LLC

Key

Technical Level

- 👤 **High school**
- 👤👤 **College**
- 👤👤👤 **Graduate**

Project size

- 🕒 **Quick result**
- 🕒🕒 **Short paper**
- 🕒🕒🕒 **Long paper**
- 🕒🕒🕒🕒 **Monograph**
- 🕒🕒🕒🕒🕒 **Major research program**

2

The Crucial Experiment

■ How Do Simple Programs Behave?

Find other diagrammatic ways to show how cellular automaton rules work.

What is the best way to illustrate—in an animation or a graphic—how a one-dimensional cellular automaton works? In NKS, I show the rules as templates for behavior, then show the result for each successive row in the output.

I have tried making animations in which successive cells on each row are filled in, with templates being highlighted as they are used. I was not particularly keen on the sequential updating that this suggests. And I really wanted to find a static diagrammatic way to indicate all the updates. Just using arrows to show data coming from different cells did not work well—since inevitably the arrows have to cross.

Ideas I have for approaching this include somehow using cells that are drawn as hexagons (so as to make the three inputs look more symmetrical), and starting from block cellular automata.

🔖 | 🕒 | **NKS page:** 24 | **Fields:** graphic design; education

Find short ways to describe common cellular automaton rules in words.

Rule 90 can be explained by saying that "a given cell becomes black if either of its neighbors—but not both—was black on the step before". What is the simplest way to explain rule 30 in words? Or rule 110? The best I came up with for rule 30 is on page 27, and it is three sentences. Try to do better. Probably the table of minimal logical expressions on page 884 will be helpful. It will be interesting to see which elementary rules take the most English text to explain. Can one automate the process of finding short pieces of text to describe rules? What would happen in different languages (compare page 1173)? Note that what is really being done here is simply to describe possible Boolean functions—they do not have to be used as cellular automaton rules.

🔖,🔖,🔖 | 🕒,🕒,🕒 | **NKS pages:** 24, 884 | **Fields:** education; linguistics

Make a gallery of programs for implementing cellular automata.

Even within *Mathematica*, there are many ways to implement cellular automata. Seeing these different ways may both be helpful for education in programming, and may suggest interpretations of cellular automata that will be useful in certain applications.

Beyond *Mathematica*, one can consider implementing cellular automata in all sorts of languages, past and present. A great many practical languages will support mainly C-like implementations. I will be surprised if there are languages that support types of implementations that cannot readily be done in *Mathematica*. But finding such implementations might be useful in suggesting ways to extend the *Mathematica* language.

👍 | 🕒 | **NKS page:** 865 | **Fields:** programming; education

Implement 1D cellular automata on as many kinds of devices as possible.

Cellular automata are—as I emphasize in the book—simple programs. So even given a very low-level or special-purpose device, cellular automata are likely to be programs that can still be implemented on it. I have seen cellular automata implementations on PDAs, cellphones, electronic billboards, diagnostic lights, printers and probably other places I have forgotten. Implementing a one-dimensional cellular automaton may not only be fun, but may also be useful—for example in providing a long-running test that exercises a somewhat random collection of elements in a device.

👍 | 🕒 | **NKS page:** 865 | **Fields:** programming; whimsy

Try to find the best possible implementation of a general cellular automaton.

After writing cellular automaton programs for years, the bitwise optimizations described on page 866 came as a surprise. I am curious what other optimizations may be possible. My guess, however, is that for one-dimensional cellular automata with given k and r but arbitrary rules, there will in fact be a fundamental limit to the optimizations that can be done with any particular formal model of computation. It seems not inconceivable that it could be rigorously proved that there is a certain lower bound on the computational effort required to carry out individual steps in the evolution of all possible cellular automata of a certain class. Note that this is a very different problem from establishing how difficult it is to determine the outcome from many steps of evolution of a particular cellular automaton.

👍,👍👍👍 | 🕒 | **NKS page:** 866 | **Fields:** programming; computational complexity theory

Experiment with audio representations of cellular automata.

Are there ways to map cellular automaton evolutions to sounds so that aspects of the evolution not clear in a visual representation now become clear? Based on the discussion on page 587 I am not especially hopeful. But it is always conceivable that there will be some particular form of regularity that is not evident in the visual domain that would be perceivable in the audio domain. And certainly there could be some interesting sounds produced. I would like one as cellphone ring, for example.

👍 | 🕒 | **NKS page:** 869L | **Fields:** art; psychology

Is there a simple ordering of states that makes certain cellular automaton mappings smooth?

With cells taken to correspond to digits in integers, only trivial cellular automata like the shift map seem to correspond to smooth mappings. Other cellular automata lead to Cantor-set-like mappings. The question is whether there might be some scheme—like a Gray code ordering—that would make an interesting set of cellular automata correspond to smooth mappings. All sorts of analysis would become possible if such a scheme could be found. It is fairly clear that there could be no general such scheme. But there might well be a scheme that would work for more interesting rules than the shift map.

👁️ | 🕒 | **NKS page:** 869R | **Fields:** discrete math; dynamical systems theory

Investigate patterns from combinations of binomial coefficients and polynomials modulo k.

$\text{Mod}[\text{Binomial}[x, y], 2]$ yields the nested rule 90 pattern. What kinds of patterns can be produced by combining Binomial and Power? When do they have simple interpretations in terms of the digit sequences x and y ? (Compare page 611.) Knowing how complex the patterns can be is helpful in assessing whether Binomial and Power might form the basis for a universal system. (Compare page 1160.)

👁️ | 🕒 | **NKS pages:** 870L, 611 | **Fields:** discrete math; number theory; computer experiment

What kinds of visually distinct nested patterns can be produced by rule 90?

I expected that with a small seed rule 90 would always eventually produce patterns that were visually just like the Sierpinski pattern of page 26, even on non-white backgrounds. So I was surprised to find the example on page 870L. I now wonder what other kinds of visually distinct nested patterns can be produced by rule 90.

👁️ | 🕒 | **NKS page:** 870L | **Fields:** discrete mathematics; computer experiment

Catalog patterns produced by combinations of integer functions modulo 2.

Multinomial coefficients and Stirling numbers both lead to simple nested patterns modulo 2. Investigate whether there are other forms of regularity seen in other integer functions—say from *Mathematica*—when reduced modulo 2.

👁️ | 🕒 | **NKS page:** 870R | **Fields:** number theory; computer experiment

Investigate curves made by bitwise functions.

Functions like $\text{BitAnd}[n, 2n]$ yield patterns that look complicated, though are effectively nested. Investigate such patterns, for example proving results about their asymptotic average properties, fractal dimensions, or Fourier (or wavelet) spectra. Consider for example $\text{BitAnd}[n, a n]$, and understand its dependence on a . What kind of nesting, if any, does $\text{BitAnd}[n, n^2]$ show? Investigate also curves made by nested bitwise functions. $\text{BitXor}[\text{BitXor}[n, 2n], 3n]$ makes a rather attractive curve. Are there combinations of bitwise functions—perhaps together with polynomials—that lead asymptotically to smooth curves? Consider also what happens with functions in which the "bitwise" operations are done in bases other than 2.

👁️ | 🕒 | **NKS page:** 871L | **Fields:** computer experiment; number theory; functional analysis | **ID:** 871-bitwise | **See Also:** 906-bitwise

Study the periodicities on diagonals of the rule 30 pattern.

On the left-hand side of the rule 30 pattern there is gradual period doubling; on the right-hand side there is rapid period doubling. More data should be obtained on both. Is there some theory or estimate that can be found of the rate of period doubling?

🔒 | 🌐 | NKS page: 871 | Fields: computer experiment; discrete mathematics

Study properties of the boundary of the region of repetitive behavior in the rule 30 pattern.

The boundary between repetitive and random behavior in the rule 30 pattern appears to follow a random walk with drift. The drift is about 0.252 cells per step—close to but not exactly $1/4$. It would be worthwhile to get much more data on the form of the boundary. Also to see if there is any theory that can be developed. My guess is that this is a more difficult version of the problem of studying the left-hand side of the difference pattern for rule 30 from random initial conditions. As discussed on page 949R, this drifts by about 0.2428 cells per step.

🔒 | 🌐 | NKS page: 871R | Fields: computer experiment; discrete mathematics

Investigate the distribution of patches of regularity in the rule 30 pattern.

It seems likely that even starting from a single black cell, rule 30 will eventually generate all possible blocks of cells of given length with essentially equal frequency (compare page 725). This means that arbitrarily large white triangles, or triangles with features like those on page 266, should eventually appear. It would be worthwhile to investigate this assertion both empirically and theoretically.

🔒 | 🌐 | NKS pages: 871R, 725, 1127L | Fields: computer experiment; discrete mathematics | ID: r30blocks1 | See Also: r30blocks2

Generate as many steps in the center column of rule 30 as possible.

One way to generate many steps in the center column of rule 30 is just to have a fast computer with a large amount of memory. In general, to go t steps requires that one be able to store a pattern at least t cells wide, and doing about t^2 cell updates. In *Mathematica* 4.2, `CellularAutomaton[30,{{1},0},t,{All,0}]` generates the sequence for $t=10^5$ in a second or so on a 1 GHz PC. In NKS (page 871) I give results for $t=10^6$. With a large-memory computer running for a long time, one should be able to get significantly further just by using *Mathematica*. One can imagine using special purpose hardware and software to go much further. There are undoubtedly good ways to parallelize doing computations on various 2D chunks, and in fact the computation of the rule 30 sequence should be viewed as a good test of a parallel computing system. Once the sequence is generated, various tests of randomness should be applied. The simplest is to find how many 1s and 0s the sequence contains. Also to search for repetition, or approximate repetition.

🔒 | 🌐 | NKS pages: 29, 871 | Fields: programming; record-breaking

Prove that the center column of rule 30 does not repeat or show regularities.

It is known that no two columns in rule 30 can repeat together. It would be very nice to prove this about a single column.

It would also be nice to prove that the sequence seems random in other ways. An exciting result would be that the sequence eventually contains all possible blocks of any given length with equal frequency, making the sequence the analog of a "normal" digit sequence (see page 912.) Proofs of randomness with respect to other tests of regularity (see page 1084R) would also be very nice—though I do not expect any non-trivial ones will be easy to get.

Also very interesting would be to prove that no nested structure is present, or, for example, that there are no peaks in a frequency spectrum (corresponding to a form of approximate repetition).

👉 | 🕒 | **NKS pages:** 28, 871 | **Fields:** discrete math proofs

Study whether statistical properties of the rule 30 pattern are changed by using different seeds.

Find initial conditions for rule 30 that make it yield a nested pattern.

Find simple initial conditions for rule 110 that make it take longest to settle down.

With just a single cell initial condition, it takes 2780 steps for the behavior of rule 110 to settle down and give rise to essentially repetitive behavior (see page 38). From its universality one knows that there must be initial conditions that in essence never settle down. The problem is to find simple examples. I expect that there will be fairly simple examples in which, for example, there are phenomena like the register machine behavior of page 100 going on.

A worthwhile step would be to enumerate successive possible initial conditions for rule 110, on various possible repetitive backgrounds, and in each case analyze their behavior. Most will presumably eventually be repetitive. But the transients and periods may be long. And eventually there will be examples where there is no repetition. The cover image (see page 851) is an example chosen for its long period.

👉 | 🕒-🕒 | **NKS pages:** cover; 851R; 32-38 | **Field:** computer experiment

Find initial conditions for rule 110 that make it yield a nested pattern.

The universality of rule 110 implies that with some initial condition it must be possible for it to produce at least in some sense a nested pattern. The problem is to find the simplest initial condition that produces an explicitly nested pattern. A variant of this problem is to find an initial condition where at least just a single column of rule 110 exhibits a nested sequence.

👉 | 🕒 | **NKS page:** — | **Field:** computer experiment

■ The Need for a New Intuition

Codify what artistic styles are evoked by cellular automaton and other patterns.

If one looks at samples of cellular automaton patterns—particularly 2D ones—one is often struck by similarity to various historical artistic styles. It would be interesting to try to classify what styles different patterns evoke. At first one might do this by hand. Later one might try to develop some analysis methodology that would automatically try to determine the style.

👉 | 🕒 | **NKS page:** 872R | **Fields:** art history; perception; artificial intelligence

■ Why These Discoveries Were Not Made Before

Find other examples of early ornamental art based on rules.

I spent considerable effort finding the examples of ornamental art based on rules that I show on page 43. What did I miss? I have some suspicion that there may be interesting examples in Chinese or Indian traditions. But unfortunately many of the artifacts created there were probably made of materials like wood that have not lasted.

👉 | 🕒 | **NKS page:** 43 | **Field:** art history

Find the precise historical origins of the Cosmati nested patterns.

Who exactly invented the idea of nested mosaic patterns? And what influenced them? So far as I can tell, nested patterns appear right around 1226, and essentially disappear within a few years. My guess is that a specific member of the Cosmati family (probably "Cosmas") came up with the idea of nested patterns, never really explained it, and died fairly soon thereafter. Many of the Cosmati works are signed, so it should be possible to trace the history by visiting appropriate sites, etc. One would then like to know whether the specific inventor of these patterns had made trips to places with obvious Byzantine, Islamic or other influences—perhaps suggesting that there might be another origin to be found.

👉 | 🕒 | **NKS page:** 873R | **Field:** art history

Is there an analog of grammar for molding profiles?

One sees many elaborate architectural molding profiles that are used historically. Are there definite rules by which these patterns can be generated? Or at least constraints that the patterns satisfy?

👉 | 🕒 | **NKS page:** 874L | **Fields:** art history; architecture; linguistics

Could there be rules behind seemingly random ancient art?

It seems impossible that the seemingly random patterns of handprints in early cave paintings could be based on simple rules. But do we actually know for sure? It would be interesting to get a good collection of samples of ancient art, and try various methods of analysis to check that there are no simple underlying rules.

👤 | 🕒 | **NKS page:** 874L | **Fields:** art history; pattern recognition; whimsy

Find other examples of human processes that are based on simple rules.

I list quite a range of examples on pages 874-5. But I am quite certain there are others. And perhaps some of them will involve more complex behavior—or at least behavior like nesting. One area that I did not investigate, but which occurs to me as I write this, is choreography.

👤 | 🕒 | **NKS pages:** 874-5 | **Fields:** history of ideas; whimsy

What might history have been like if cellular automata had been investigated in antiquity?

What would the history of mathematics have been like if Euclid had written a book about elementary cellular automata? What if his work on geometry had not been done in antiquity? How might ideas from arithmetic, algebra, geometry and calculus have arisen if cellular automata had been investigated first?

How would knowledge of phenomena like rule 30 have affected thinking about theories in physics, biology, etc.?

How would education be different today if cellular automata had been part of the curriculum in the Middle Ages?

What would the history of technology have been if implementing cellular automata had been a major objective?

👤 | 🕒 | **NKS page:** — | **Fields:** history; whimsy

Fill in more details of the history of cellular automata.

The fact that they should have been so easy to invent and investigate makes cellular automata a particularly fascinating subject for study in the history of science. I have spent considerable effort investigating the history of cellular automata. But there are still many specific points that I have been unable to clarify.

Examples include:

- What was the exact interaction between Stan Ulam and John von Neumann that started von Neumann studying 2D cellular automata?

- What was the context in which Stan Ulam studied "recursive patterns of growth" in the early 1960s? (I discussed history with Stan Ulam in the early 1980s, but suspect I did not ask quite the right questions.)

- What discoveries about cellular automata were made in the 1950s and early 1960s in the context of cryptographic work? There is good evidence that extensive work was done, but it appears still to be classified. Is it in fact? Is it also secret in the former Soviet Union?

👤 | 🕒-🕒🕒🕒🕒 | **NKS pages:** 876-878 | **Fields:** history of science; human investigation

Was the irregularity of the primes discussed in antiquity?

Quite a few primes were certainly known in antiquity (see page 910). And it must have been noticed that they appeared quite irregularly. But were there ever comments made about this? If so, by whom, and what was concluded from them?

A related question is whether the irregularity of solutions to Diophantine equations was ever discussed.

👍 | 🕒 | **NKS pages:** 48, 910 | **Fields:** history of mathematics; classical scholarship

Was there early discussion of irregularity in the digits of transcendental numbers? ++

Explanation...

👍 | 🕒 | **NKS page:** 48

What simple programs were run on early computers?

Programs set up to implement traditional scientific ideas tended to be quite complicated. Were simple programs ever run on early computers? The few examples that I know of are listed on pages 878-890. Are there more? Did people in the Babbage or Jevons traditions ever consider the possibility of running simple programs?

👍 | 🕒 | **NKS page:** 49 | **Field:** history of computing

Study why more was not done when complexity was seen in simple programs.

Pages 878-890 list 41 examples of cases where complex behavior was seen or was almost seen in the study of simple programs. It would be interesting to understand in detail in each case why the phenomena discussed in NKS did not end up actually being discovered.

I spent considerable effort finding the various examples on pages 878-890. But there may well be at least a few more examples, and it would be interesting to find them.

👍 | 🕒 | **NKS pages:** 878-890 | **Field:** history of science

Study the code 10 cellular automaton.

Even though this was probably the very first cellular automaton in which I saw complex behavior in evolution from a single black cell, I have never studied it in any detail. Someone should.

👍 | 🕒 | **NKS page:** 882

Work out how to organize human simulations of cellular automata.

Can one set up a "game" so that rows of people sitting in a lecture room (or stadium) implement cellular automaton rules? Figure out the practical details, make it happen, and take photographs of it. Perhaps each person in a particular row of an auditorium hands a black or white counter to each of the three people in front of them. Then each person puts those counters on a "rule card" to see what color they should be. There will probably be rules that work better and worse with particular practical schemes. Try to find a good scheme for rule 30, for example.

👤 | 🕒 | **NKS page:** 850L | **Field:** whimsy

3

The World of Simple Programs

■ More Cellular Automata

Find an explicit formula for the cell colors in rule 225.

Just like for rule 90 and rule 150, there should be an explicit formula for the color at step t of the cell at position x in rule 225.

👁 | 🕒 | NKS pages: 58, 885 | Fields: discrete math

Study the same questions for rule 45 as for rule 30.

Study the further properties of rule 73.

What growth rates for patterns can be achieved by simple cellular automata?

Rules like 30 and 90 give patterns that show linear growth. Rule 225 gives square root growth. Code 1599 (page 70) gives irregular growth. What other growth functions can be achieved by cellular automata with simple rules?

Linear growth by r cells per step is the fastest possible. Rational fractions of this rate should be fairly easy to achieve. Average rates corresponding to irrational fractions should also be possible (pages 614 and 661 show rates of the form $\text{Log}[p, q]$). Fractional power growth should also be possible. So should logarithmic growth (compare Turing machine (f) of page 79).

Is there a systematic way to take a growth rate function and "compile it" to a cellular automaton rule that will lead to a pattern which asymptotically grows at that rate? Note that with a sufficiently complicated rule it should be possible to get any possible growth function.

At a more experimental level, one can ask what the distribution of growth rates is for all cellular automata with a certain type of rule.

👁 | 🕒 | NKS pages: 57, 885, etc.

Study $k=3$, $r=1/2$ cellular automata.

The $k=3$, $r=1/2$ rules are in a sense the next obvious set to study after the $k=2$, $r=1$ elementary rules. There are 1734 fundamentally inequivalent cases, which is a small enough number that it should be possible to study all of them in some detail. Page 885L shows that there are some interesting phenomena to see, that are not evident with $k=2$, $r=1$ elementary rules.

👉 | 🕒 | **NKS page:** 885R | **Field:** computer experiment

Develop a classification of behaviors of cellular automaton from simple initial conditions.

Starting on page 235, I discuss how cellular automata with random initial conditions can be classified into four types of behavior. What is the analogous classification for simple initial conditions? There are several issues. One is whether the pattern grows, and if so, roughly how fast. Another is how regular or irregular the pattern is inside. It can be repetitive, or nested, or seemingly random, or have definite localized structures. It would be good to have systematic classification.

One should then be able to develop an automated system that decides for most cellular automata where they should lie in the classification. Given this, it should be possible to ask questions about how different kinds of behavior get more or less frequent as the underlying rules for the cellular automata are changed. Do certain kinds of behavior get more common when the size of the rule increases (compare p. 948L)? Do totalistic rules show substantial differences of any kind from general ones? What about symmetric rules?

👉👉 | 🕒🕒 | **NKS pages:** 57 etc. | **Fields:** computer experiment; analysis techniques

Develop automated ways to find "interesting" cellular automata.

I found most of the examples of cellular automata given in Chapter 3 by generating printouts of thousands of rules, and looking at them by hand. It should be possible to develop an automated system for finding "interesting" rules. The main difficulty, of course, is that until "interesting" behavior is identified, it is hard to know what it will be. My approach for most systems has been to make a series of filters that remove "uninteresting" (e.g. repetitive, nested, etc.) behavior, and then progressively to improve these filters until what is left over is really "interesting".

👉 | 🕒 | **NKS pages:** 66 etc. | **Fields:** computer experiment; analysis techniques

Find ways to estimate the frequencies of different types of cellular automaton behavior.

If one looks at all possible cellular automaton rules, can one make a quantitative estimate of what fraction will show repetitive, nested, etc. behavior? Is there a definite limiting fraction of rules that, say, show nested behavior, perhaps with a certain fractal dimension? In the most general case, frequencies of different forms of behavior are probably non-computable, but there may be asymptotic statistical estimates that can be given and that have at least some domain of validity.

👉👉 | 🕒 | **NKS page:** 00 | **Fields:** statistical mechanics; discrete math

What repetition periods can be achieved by simple cellular automata?

Starting from, say, a single cell, what are the possible repetition periods that can occur in the patterns generated by particular types of cellular automata? (Compare page 1186L.) Is the distribution typically exponentially damped (compare 887R for mobile automata)? If so, why? What are the outliers? There seem to be some very long ones. Consider both patterns that achieve only a finite width (and so have to repeat), and ones that, say, grow linearly, as on page 63.

👁 | 🕒 | **NKS pages:** 62, 63, etc. | **Fields:** computer experiment; discrete mathematics

What forms of nesting can simple cellular automata produce?

Page 62 shows cellular automata that generate the same nested patterns as $k=2$ and $k=3$ additive rules. What other nested patterns can occur? Page XXX argues that nesting is inevitably associated with some form of additivity. But it could well be that a fairly simple non-additive rule evolving from, say, a single cell, could emulate a quite complicated additive rule. There is also the question of nested patterns like the one from rule 225. What possible forms do they have? The main thing that will probably be difficult is to build a general tool for recognizing patterns that should be considered nested.

👁👁 | 🕒 | **NKS page:** 62 | **Fields:** computer experiment; analysis techniques

Find the long-term behavior of code 1635.

What happens if one extends the picture on page 67? Does the center region eventually become repetitive? Or does it take over the whole pattern? How sensitive are the results to the details of the initial conditions used?

👁 | 🕒 | **NKS page:** 67 | **Field:** computer experiment

Characterize the long-term behaviors of code 1599.

Page 70 shows that code 1599 can take a long time to stabilize. What general theory can be developed about its outcome from different initial conditions? Does it always eventually stabilize (I doubt it)? What persistent structures can it leave at the end? What determines how many of them will be left, and over how wide a field?

👁 | 🕒 | **NKS page:** 70 [see also Chapter 6:gliders] | **Fields:** computer experiment; discrete mathematics

Which cellular automata generate patterns that take longest to stabilize?

Code 1599 takes 8282 steps to stabilize, even starting from a single black cell. Which rules of given types yield patterns that take longest to stabilize? One can look either for patterns that actually die out, or for ones that become repetitive, either with any period, or with a given period. This is analogous to the Busy Beaver Problem discussed for Turing machines. In general the problem is undecidable—there is no way to find out in general what will happen after an infinite time (see page 754). But in practice it will be possible at least to get good candidate "winners". One can imagine trying to use automated theorem proving techniques to establish whether particular cellular automata actually stabilize or not.

👁 | 🕒 | **NKS page:** 70 | **Field:** computer experiment | **ID:** ca-stab | **See Also:** ca-stab-prove

Study compositions of cellular automaton rules and the patterns they produce.

One can imagine making a "composite" cellular automaton rule by applying two (or some sequence) of successive rules. One can ask many questions about such compositions. An example (studied in simple cases in the dynamical systems literature) is when the rules "commute", so that the same final rule is obtained regardless of the order in which the rules are composed. In general, one can imagine building up a whole algebra of rules. One can define "words" corresponding to compositions of rules as being made of "letters" corresponding to specific rules. Then one can ask questions about the equivalence of different words, etc. No doubt there are some general proofs that can be given. But a good approach will be just to look at the rule tables defined by compositions of rules, and see when they are and are not the same.

One can also study compositions of rules in a more experimental way. It would be interesting to try to develop a theory of what patterns can arise from compositions of cellular automata that individually yield patterns with certain kinds of regularities. Is the distribution of behavior seen in compositions of rules of some type (say, elementary rules) similar to the distribution seen in the individual rules?

👉👉👉 | ☉☉ | **NKS page:** 886R | **Fields:** dynamical systems theory; computer experiment

Invent other convenient types of cellular automaton rules.

In NKS I mostly consider either general cellular automaton rules, or totalistic (or outer totalistic) ones. Try to invent other convenient types of rules that capture typical behavior. These would be useful in having a basis for doing systematic explorations of large spaces of possible rules. An obvious approach is to consider generalizing the idea of totalistic rules using operations other than Plus. Another possible approach would be to consider rules that are built up as compositions of "primitive" rules. Still another would be to look at rules represented by Boolean functions with particular characteristics (say a particular type of DNF representation).

👉 | ☉ | **NKS pages:** 886L etc. | **Fields:** constructions; abstract system design

Study additive cellular automata that depend on step number.

In *Mathematica*, one can specify an additive cellular automaton by giving CellularAutomaton a rule function such as `Mod[Apply[Plus, #], k]&`. What happens if one includes dependence on step number, say as in `Mod[Apply[Plus, #]+#2, k]&`? What functional dependence on step number is necessary to get something other than a nested pattern?

👉 | ☉ | **NKS page:** — | **Field:** computer experiment

Investigate cellular automata based on complex numbers.

Cellular automata like rule 90 can be thought of as being based on applying simple arithmetic operations to integers modulo 2. One can also set up cellular automata that are based on finite sets of complex numbers that happen to be closed under particular arithmetic operations. Do these always yield simple nested patterns? When they do, what is the relationship between the properties of the field and the fractal dimension? What is their general behavior, starting not necessarily always just from a single nonzero cell? Are there examples in which the complex numbers involve algebraic extensions to the ordinary integers?

👉👉 | ☉ | **NKS page:** 886R | **Fields:** number theory; abstract algebra; computer experiment

Investigate cellular automata based on quaternions.

Just as one can set up cellular automata based on finite sets of complex numbers (see [[project above]]), so one can also do this for quaternions, etc. Investigate the properties of the resulting systems. Under what circumstances are nested patterns obtained?

 |  | **NKS page:** 886R | **Fields:** number theory; abstract algebra; computer experiment

Investigate cellular automata based on systems from abstract algebra.

As shown on page 886R, one can take any "multiplication table" and use it as a cellular automaton rule. Investigate how the types of patterns produced by the cellular automaton relate to properties of the multiplication table, and properties of the algebraic system it describes. Investigate in more detail the case (c) at the bottom of page 886R, produced by the multiplication table of the group S_3 . (Compare page 945.)

 |  | **NKS page:** 886R | **Fields:** abstract algebra; group theory; computer experiment

■ Mobile Automata

Trace as many steps as possible of the mobile automaton on page 75.

What regularities occur? To what extent is the path of the active cell a random walk? See whether its outer boundaries continue to grow like \sqrt{t} . Try to prove that they always do. Page 887R mentions that after a billion steps, the pattern is 57,014 cells wide.

 |  | **NKS pages:** 75, 887R | **Field:** programming

Make systematic studies of mobile automata.

I have studied elementary cellular automata in considerable detail. Do the same level of investigation of simple mobile automata. Find what regularities can be found. Try to characterize different general types of behavior. See in what ways a more global theory can be made than in cellular automata.

 |  | **NKS pages:** 72 etc. | **Fields:** computer experiment; analysis techniques

Study the effect of different initial conditions on mobile automata.

In the book, I consider almost exclusively the case of mobile automata with blank initial conditions. Study what happens when more complicated initial conditions are allowed. Look at both structured (e.g. repetitive or nested) initial conditions. Can some of the 65,536 simplest mobile automata show complex behavior when their initial conditions are not blank?

 |  | **NKS pages:** 72 etc. | **Field:** computer experiment

Understand the distribution of repetition periods for mobile automata.

Page 887 shows a roughly exponential decay of repetition periods for behavior produced by mobile automata with all possible rules of certain kinds. Develop some form of statistical or other theory that explains this exponential, and perhaps gives corrections to it.

 |  | **NKS page:** 887R | **Fields:** discrete math; statistical physics

Find simple mobile automata whose active cells trace out various types of paths.

The active cell must move by only one cell at each step. But what kinds of paths can it follow? Can one find—by search or construction—mobile automata whose active cells trace out simple algorithmic or number theoretical paths?

 |  | **NKS page:** 887R | **Fields:** construction; computer experiment | **ID:** ma-paths | **See Also:** tm-paths

Investigate the basic phenomenology of generalized mobile automata.

NKS gives just a few examples of generalized mobile automata. Investigate their general phenomenology. Does the number of active cells tend either to be quite small on each step, or fixed fraction of maximal? What is the correlation between number of active cells and likelihood of different types of behavior? Find simple generalized mobile automata where the number of active cells varies in a complicated way as the system evolves.

 |  | **NKS page:** 77 | **Field:** computer experiment

Investigate the connection between complex behavior and number of active elements.

Fewer mobile automata than cellular automata seem to exhibit complex behavior. Looking at generalized mobile automata—which in a sense interpolate between cellular automata and mobile automata—suggests that complex behavior becomes more common when the number of active cells can be larger. Investigate this connection in more detail, and try to understand its origins. Does it apply to systems of all types? Can one give a quantitative estimate, perhaps based on statistical arguments, for what fraction of rules should show complex behavior when there is a certain density of active cells?

 |  | **NKS page:** 76 | **Fields:** computer experiment; analysis

Investigate mobile automata with more than two colors for each cell.

None of the 65,536 mobile automata with rules of exactly the kind shown on page 71 exhibit complex behavior. What happens if one allows more than two colors for each cell? What is the simplest rule that leads to complex behavior?

 |  | **NKS page:** 72 | **Field:** computer experiment

Investigate mobile automata with more than nearest-neighbor rules.

None of the 65,536 mobile automata with rules of exactly the kind shown on page 71 exhibit complex behavior. What happens if one allows the active cell to move by more than one position at each step? Can one get complex behavior? What is the simplest rule that produces it?

👉 | 🕒 | **NKS page:** 72 | **Field:** computer experiment

■ Turing Machines

Trace as many steps as possible of the Turing machine on page 81.

What regularities occur? To what extent is the path of the head a random walk? See whether its outer boundaries continue to grow like \sqrt{t} . Try to prove that they always do.

👉 | 🕒 | **NKS pages:** 81, 888R | **Field:** programming

Make systematic studies of simple Turing machines.

I have studied elementary cellular automata in considerable detail. Do the same level of investigation of simple Turing machines. Find what regularities can be found. Try to characterize different general types of behavior. See in what ways a more global theory can be made than in cellular automata. See what happens if something other than a blank tape is used for the initial conditions.

👉 | 🕒🕒 | **NKS pages:** 79 etc. | **Fields:** computer experiment; analysis techniques

Find simple Turing machines whose heads trace out various types of paths.

The head must move by only one cell at each step. But what kinds of paths can it follow? Can one find—by search or construction—Turing machines whose heads trace out simple algorithmic or number theoretical paths?

👉👉 | 🕒 | **NKS page:** 887R | **Fields:** construction; computer experiment | **ID:** tm-paths | **See Also:** ma-paths

Investigate numbering schemes for Turing machines.

There is a very obvious way to number possible cellular automaton rules. For Turing machine rules, it is less obvious how to do it. The way I use in NKS seems as good as any. Study what the issues are with other numbering schemes.

👉👉 | 🕒 | **NKS page:** 888R | **Field:** discrete math

Define interesting subclasses of Turing machines to investigate.

Totalistic rules represent an interesting subclass of all possible cellular automata. Find similar subclasses of Turing machines. Totalistic Turing machines are one possibility. Also consider, for example, Turing machines with just 2 states and 2 colors, but where the head can move by more than one position.

👉 | 🕒 | **NKS page:** 888R | **Field:** computer experiment

Investigate Turing machines with periodic initial conditions.

Most of the pictures I show for Turing machines assume a blank tape initial condition. What happens if the tape initially has a repetitive pattern on it? What kinds of analogs of localized structures can develop?

👉 | 🕒 | **NKS pages:** 888R, 889L | **Field:** computer experiment

Did Alan Turing ever simulate an explicit Turing machine?

Alan Turing used Turing machines as the basis for various theoretical constructions. In the early 1950s (late in his life), he did various computer simulations of biological systems, mainly using PDEs. From the directions he followed, it seems conceivable that he might have tried looking at explicit simple Turing machines, but so far as I can tell, he never actually did. Is this correct? I have tried looking at archives and secondary sources, and have never seen any evidence that Turing did any explicit simulations.

👉 | 🕒 | **NKS page:** 889L | **Field:** history

Find minimal Turing machines that act as base k counters.

Machine (f) on page 79 acts as a base-2 counter. Find the smallest machines that act as base-k counters for successive k.

👉 | 🕒 | **NKS pages:** 79, 888R | **Fields:** computer experiment; construction

Investigate localized structures in systems like Turing machines.

Page 889L shows examples of localized structures that can exist on various repetitive backgrounds in the Turing machine from page 81. For what Turing machine rules can such structures exist? Find backgrounds where there are a wide range of structures.

👉 | 🕒 | **NKS page:** 888R | **Field:** computer experiment

■ Substitution Systems

Invent other ways to display the behavior of substitution systems.

The book uses several graphical ways to display the behavior of substitution systems: fixed size elements (page 82), subdividing elements (page 83), trees (page 84), paths (page 892L), 2D arrays (page 892R). None is completely satisfactory. Are there better ways? Page 989L gives some additional ways to represent trees. Can variants of these be used to display the behavior of substitution systems?

👁 | 🕒 | **NKS pages:** 82-84, 892 | **Fields:** graphic design; exposition

Make detailed analyses of a range of simple substitution systems.

Neighbor-independent substitution systems must yield nested behavior. And in some sense this is always simple enough that it will admit complete analysis. But often that analysis is quite difficult to do in terms of traditional mathematical constructs. The Notes do analysis for some specific substitution systems. It would be good to extend that to many more substitution systems. This is made particularly worthwhile by how often substitution system behavior seems to show up in all sorts of systems.

👁 | 🕒 | **NKS pages:** 83, 84, 890, 892 | **Fields:** analysis; discrete math

Find the simplest generating function for the Thue-Morse sequence.

The generating function I give for the Thue-Morse sequence on page 890L is fairly complicated. I did a fairly extensive search, but failed to find a simpler one. I suspect, however, that simpler ones do exist, and it would be interesting to find the very simplest generating function that can be used. One could imagine developing a general theory of simplest generating functions for substitution systems.

👁👁 | 🕒 | **NKS page:** 890L | **Fields:** computer experiment; computer algebra

Investigate Fibonacci sequences modulo n .

$\text{Mod}[\text{Fibonacci}[n], n]$ yields a remarkable complicated sequence. But I suspect that it should be possible to give a fairly complete analysis of it, for example representing the result for a given n in terms of standard number-theoretical functions. I also suspect that there are fairly simple primitive recursive, register machine and perhaps Turing machine rules that will give the sequence. Quite probably this has been investigated before—but I was not able to find literature on it.

👁👁 | 🕒 | **NKS page:** 891L | **Field:** number theory

Analyze paths generated by simple substitution systems.

Even very simple neighbor-independent substitution systems are able to generate visually complex paths (see especially the third example on page 892L) that show no particular trace of nesting. What are the asymptotic shapes obtained from various simple substitution systems? When are they obviously nested? What kind of general characterizations can be given of them? Is there a fast way to find out whether a particular link on the lattice will be filled in at a particular step? What is the growth rate of the number of distinct links that have been filled in? (It is common for particular links to be visited multiple times by the path.)

👉 | 🕒 | **NKS page:** 892L | **Fields:** computer experiment; analysis

What does it take to make two-color neighbor-dependent substitution systems show complex behavior?

Among neighbor-dependent substitution systems with two colors, and depending on one neighbor, page 86 states that there are none that have linear or below growth, and that show anything more than purely repetitive behavior. What happens if one allows slightly faster growth? What is the simplest rule that shows complex behavior? How complicated do the initial conditions need to be?

👉 | 🕒 | **NKS page:** 86 | **Field:** computer experiment

Investigate substitution system (d) on page 87.

System (d) on page 87 is remarkable for showing what appears to be almost continuous overall behavior. What is it doing? Is there a simple function which gives the number of elements at step t ?

👉 | 🕒 | **NKS page:** 87 | **Fields:** computer experiment; analysis

Study the general phenomenology of simple neighbor-dependent substitution systems.

Page 87 gives a few specific examples of neighbor-dependent substitution systems with at most linear growth, but interesting behavior. There should be a systematic study done of simple neighbor-dependent substitution systems. Can one classify the types of behavior that occur? What growth rates occur? Are there many examples of continuum-like behavior of the kind seen in rule (d) on page 87?

👉 | 🕒 | **NKS page:** 87 | **Fields:** computer experiment

Find the simplest substitution system rules that achieve particular growth rates.

Neighbor-independent substitution systems have growth rates that are determined by fairly straightforward linear algebra. But neighbor-dependent substitution systems can have arbitrary growth rates (though not more than exponential). Find the simplest rules that yield particular forms of growth, either for all t , or for large t . Examples to seek would include \sqrt{t} , $\log t$, solutions to various recurrence and differential equations and running times of various algorithms. Perhaps one could even imagine a "compiler" that would take a specified primitive recursive function, and generate a substitution system with that growth rate. (The literature of L systems includes some specific examples of rules with particular growth rates, but usually based on complicated constructions that lead to complicated rules.)

 |  | **NKS page:** 890R | **Fields:** computer experiment; algebra

Investigate alternative formulations of neighbor-dependent substitution systems.

There is a certain arbitrariness in the way the first and last elements of neighbor-dependent substitution systems are handled in the formulation I use in the book. Investigate alternative formulations. Most can be set up just by adding more arguments to the Partition used in the implementation on page 889R. One obvious case to try is cyclic boundary conditions.

 |  | **NKS page:** 889R | **Field:** computer experiment

■ Sequential Substitution Systems

Make a systematic study of simple sequential substitution systems.

Investigate the general classes of behavior in sequential substitution systems. How important are initial conditions? What growth rates can be achieved? Among rules with 2 colors and 2 replacements, can complex behavior ever be found?

 |  | **NKS pages:** 91 etc. | **Field:** computer experiment

Make a detailed study of the sequential substitution system on page 92.

The sequential substitution system on page 92 was the simplest example I found that shows significantly complex behavior. Make a detailed analysis of it, finding whatever regularities do exist. Is the compressed evolution equivalent to some other system, like a cellular automaton? What is the asymptotic growth rate of the number of elements? How sensitive are the results to the initial condition used? Does the system ever evolve to simple behavior?

 |  | **NKS page:** 92 | **Fields:** computer experiment; analysis

Find simple sequential substitution systems that perform recognizable tasks.

There is a very simple sequential substitution system that sorts sequences of 0s and 1s. Look for, or perhaps construct, other simple sequential substitution systems that implement other simple algorithms. Examples might include various kinds of arithmetic. It is fairly straightforward to compile programs to complicated sequential substitution systems. The question is whether simple sequential substitution systems can be found.

👉 | 🕒 | **NKS page:** 894L | **Fields:** computer experiment; construction | **See Also:** emul-ca-1

Investigate alternative formulations of sequential substitution systems.

The setup I use in the book for sequential substitution systems is based on a particular order of using replacements, that is the way /. works in *Mathematica*. In Chapter 9 I discuss some alternatives. Investigate the phenomenology of simple rules that are based on alternative orderings. Try to find a good parametrization for different possible orderings.

👉👉 | 🕒🕒 | **NKS page:** 894L | **Field:** computer experiment

■ Tag Systems

Systematically study all the simplest type of tag systems.

If one allows blocks up to length 3, there are just 50,625 possible tag systems. Systematically investigate all of them. How common are different forms of behavior? What is the distribution of halting times?

👉 | 🕒🕒 | **NKS pages:** 94 etc. | **Field:** computer experiment

Study the long-term behavior of the tag systems on page 94.

What is the long-term behavior of tag systems (d), (e) and (f)? Are there ultimately regularities? Is there an interpretation in terms of other systems (cellular automata, number theory, etc.) that can be given of their behavior? What is the asymptotic growth rate of the number of elements generated?

👉 | 🕒 | **NKS pages:** 94, 894R | **Field:** programming; analysis

Can Post's original tag system yield anything other than repetitive behavior?

Around 1921, Emil Post looked at a particular tag system that deletes three elements at each step. For initial conditions up to size 28, the system always eventually settles down to repetitive behavior. Investigate larger initial conditions. Try to find one that leads to more complex behavior, or prove that such behavior is not possible.

👉👉 | 🕒 | **NKS page:** 895L | **Fields:** programming; record-breaking

■ Cyclic Tag Systems

Systematically study all the simplest type of cyclic tag systems.

Cyclic tag systems have the feature that a large fraction of possible simple rules seem to lead to complex behavior. Systematically investigate all the simplest types of rules. How common are different forms of behavior? What is the distribution of growth rates?

👉 | 🕒 | **NKS pages:** 96 etc. | **Field:** computer experiment

Study the long-term behavior of the cyclic tag systems (d) and (e) on page 96.

What is the long-term behavior of cyclic tag systems (d) and (e)? Does the growth rate approach half an element per step? Are there ultimately regularities? Is there an interpretation in terms of other systems (cellular automata, number theory, etc.) that can be given of their behavior?

👉 | 🕒 | **NKS pages:** 96, 895R | **Fields:** programming; analysis

Build a mechanical cyclic tag system.

Cyclic tag systems have rules that admit a fairly straightforward mechanical implementation. Build an actual kinetic sculpture that is based on a cyclic tag system. It should be interesting to watch: it will have a simple mechanism, yet it will produce a seemingly random sequence of colored balls. Will the mechanism get stuck for cyclic tag systems that have complex behavior?

👉 | 🕒 | **NKS page:** 895 | **Fields:** whimsy; mechanical building

Investigate what happens with more blocks and more colors.

Cyclic tag systems with two blocks and two colors already exhibit highly complex behavior. See whether there are any qualitatively new forms of behavior that occur if the number of blocks or number of colors is increased.

👉 | 🕒 | **NKS page:** 895L | **Field:** computer experiment

■ Register Machines

Study the further phenomenology of register machines.

In NKS I looked at all register machines with 2 registers and up to 8 instructions. Catalog the behaviors seen in these machines, then find out what happens if one goes further. When does behavior more obviously complex than page 100 start to occur? Consider also what happens if one starts with nonzero starting values in the registers.

👉 | 🕒 | **NKS pages:** 100 etc. | **Field:** computer experiment

What is the halting time distribution for simple register machines?

Find the distribution of halting times for register machines with progressively longer programs. Is there some simple statistical argument that yields a good approximation to the result? Limiting properties when arbitrarily many register machines are included will be non-computable, and related for example to Chaitin's Ω .

🔒 | 🕒 | **NKS page:** 896L | **Fields:** computer experiment; analysis

Find small register machines that take longest to halt.

Page 896 gives a register machine with a program of length 8 that takes 1280 steps to halt. Find what register machines with longer programs take longest to halt. This is analogous to the Busy Beaver Problem for Turing machines. One expects that as the length of program increases, the longest halting time will eventually increase very rapidly. But at least for short programs the halting times are still short enough to investigate—so it should be possible to trace them somewhat further than for Turing machines.

🔒 | 🕒 | **NKS page:** 896L | **Fields:** programming; record-breaking

Find out what functions can be computed by small register machines.

One can immediately view any register machine that halts as computing a definite function. If the initial value in, say, the first register is n , and the final value in that register is $f[n]$, then the register machine can be said to compute the function f . What distinct functions are computed by register machines with successively longer programs? (Compare the analogous discussion for Turing machines on page 762.)

🔒 | 🕒 | **NKS pages:** 100 etc. | **Field:** computer experiment

Find minimal register machines that implement number theoretical problems.

The register machine on page 100 effectively implements a number theoretical problem close to the $3n+1$ problem. Find the smallest register machine that actually implements the $3n+1$ problem. Find minimal machines that implement other standard number theoretical problems and operations. (See also page 1114.)

🔒 | 🕒 | **NKS page:** 100 | **Fields:** computer experiment; number theory

Construct an algebra of equivalences between register machines.

Many register machines programs are ultimately equivalent. Come up with a way to see these equivalences using a system of transformations on the programs. The basic methodology may be similar to the way optimizing compilers make transformations on code. But this is a minimal example, potentially much more amenable to analysis. There will be all sorts of equivalence transformations—or equations—between register machine programs. Try to turn these into some form of systematic algebra, in terms of which equivalence and other theorems can be proved. Note that the general problem of equivalence between systems like register machines—or, say, combinators—has been known for many decades to be undecidable.

🔒🔒 | 🕒 | **NKS page:** 896L | **Fields:** software engineering; theorem proving

Investigate register machines with alternative instruction sets.

In Chapter 3 I give explicit examples only for a particular, simple, register machine instruction set. I tried other instruction sets, but did not immediately find qualitatively different results. Investigate alternative instruction sets more systematically, finding in each case the smallest register machine with seemingly random behavior.

🔒 | 🌐 | **NKS pages:** 101, 896L | **Field:** computer experiment

Make a study of actual machine-code programs chosen at random.

Register machines are a simple idealization of actual machine code. What happens if one takes actual machine code for some computer system, constructs short programs at random from it, and then runs them? For simplicity, one could assume that the programs operate only on registers. What is the halting-time distribution for these programs? What functions can be computed? The general issue of studying random machine-code programs has connections to issues in computer security.

One can also consider the same questions for programs constructed, say, from sequences of Java byte codes.

🔒 | 🌐 | **NKS page:** 102 | **Fields:** computer experiment; programming

Study the smallest possible procedural programs, say in C or Java.

If one restricts variables and structure of programs, it becomes realistic to enumerate all possible programs, even in languages like C or Java. What is their typical behavior? How many of them compute the same functions? What fraction of small programs halt? What is the shortest program that generates seemingly random output? (Compare the minimal cellular automaton programs on page 884, and the primitive recursive functions on page 908.)

🔒 | 🌐 | **NKS page:** 102 | **Fields:** programming; computer experiment

Enumerate and study small *Mathematica* programs of particular types.

Given a particular set of basic *Mathematica* functions, it is quite straightforward to enumerate all possible small programs that can be constructed from them. Try this for various sets of *Mathematica* functions. See what the shortest program is with nontrivial behavior. See what fraction of the enumerated programs compute the same functions. Try to find short programs for recognizable functions.

Starting with different sets of *Mathematica* functions is like doing programming in different styles. Compare results with different "styles of programming". One can imagine measuring which styles are more powerful, by looking at how long a program is needed to compute a particular function, or to achieve a certain level of complex behavior. One can also imagine measuring questions like which styles are "safer", in the sense, say, that they are less prone to generate infinite loops.

🔒 | 🌐 | **NKS page:** 102 | **Fields:** computer experiment; *Mathematica*

■ Symbolic Systems

Make a systematic study of the phenomenology of simple symbolic systems.

Enumerate all symbolic systems with the simplest rules, and see what they do. One will often have to adjust the initial condition to make sure that the rule applies at least once. Can one classify the behavior that occurs? What growth rates can be achieved? What halting time distributions occur? What functions are computed?

👉 | 🕒 | **NKS page:** 104 | **Field:** computer experiment

Find good ways to visualize the evolution of symbolic systems.

The rules for symbolic systems are fundamentally non-local, which makes it difficult to visualize their behavior. The rules are somewhat more local when applied to trees. Page 897R shows an example of evolution in terms of trees. But despite its greater locality, this representation does not seem especially illuminating. Find a better representation, perhaps by representing the form of the tree in a two-dimensional way.

👉 | 🕒 | **NKS pages:** 104, 897R | **Fields:** graphic design; exposition

Find a symbolic system whose halting times grow even faster than those on page 103.

The simple symbolic system on page 103 can take n iterated powers of 2 steps to halt if given input of size n . Find a simple symbolic system whose halting time grows even faster, perhaps as fast, say, as the lengths of Goodstein sequences (see page 1163L). Note that if the halting times grow that fast, then it may be unprovable in the context of an axiom system like Peano arithmetic that the system halts at all.

👉👉 | 🕒 | **NKS pages:** 103, 897 | **Fields:** computer experiment; recursive function theory

Study order dependence in symbolic systems.

The symbolic systems in Chapter 3 all apply their rules by using the *Mathematica* $/.$ operation. This corresponds to a particular scheme and particular order of rule application. Study the effect of using other application orders and schemes.

👉👉 | 🕒 | **NKS page:** 898L | **Fields:** computer experiment; mathematical logic

Find order-independent symbolic systems.

As mentioned on page 898L, some symbolic systems have the so-called Church-Rosser property, which means that their final outcome is always the same for a given input, regardless of the order in which the individual rules are applied. Find the simplest symbolic systems that have this order-independence property. How common is order independence? Is it easy to characterize classes of symbolic systems that do and do not have it? Do the simple order-independent symbolic systems have interpretations in terms of more familiar order-independent operations (like those of arithmetic)?

👉👉 | 🕒 | **NKS page:** 898L | **Fields:** computer experiment; mathematical logic

Study operator systems formed by iterating transformations on patterns.

Page 898R shows behavior from a few very simple operator systems. Make a systematic study of such systems. Enumerate all possible simple *Mathematica* transformations for patterns, and see what happens if one applies them repeatedly. It will probably be necessary to look not only at the linear representations of what can be generated, but also, for example, at tree forms.

 |  | **NKS page:** 898R | **Fields:** computer experiment

Study symbolic systems on networks.

Ordinary symbolic systems in effect correspond to making local transformations on trees. Study systems that instead make transformations on networks. One way to set up such systems—as extensively discussed in Chapter 9 (e.g. page 508)—is to have rules that make transformations on fixed subnetworks. Another potential approach—more directly analogous to symbolic systems—is to have transformations that apply to subnetworks defined by patterns. *Mathematica* patterns are largely oriented towards the identification of subtrees in expression trees. But one can imagine generalizing them to represent classes of subnetworks in networks.

  |  | **NKS page:** 898R | **Fields:** computer experiment; discrete mathematics

■ Some Conclusions

Analyze typical changes in behavior as more complex rules are allowed.

A basic observation (ultimately captured in the Principle of Computational Equivalence) is that above some low threshold, the behavior that classes of systems exhibit no longer seem to get fundamentally more complex. Can one make explicit measurements that reflect this phenomenon? Given a scheme for identifying different types of cellular automaton behavior, for example, one can easily see to what extent more complex types of behavior get more common when the number of colors or the range of the rule is increased. One can also look at other measures of randomness and complexity, as discussed in Chapter 10. Then in each case one can try seeing how these measures change when one looks at more complicated classes of underlying rules, whether for cellular automata or other kinds of systems.

  |  | **NKS page:** 106 | **Fields:** computer experiment; analysis

■ How the Discoveries in This Chapter Were Made

Develop well-codified principles for computer experiments.

In traditional experimental science there are various principles of good experimentation that have emerged. One basic example is that experiments should be set up to be as repeatable as possible. Pages 109 etc. give an informal discussion of analogous principles for good computer experimentation. Try to codify these principles as much as possible. Consider how they relate to ideas about the nature of experimental science, that have been discussed for example in the philosophy of science.

👁️ | 🕒 | **NKS pages:** 109 etc. | **Fields:** computer experiment; philosophy of science

Catalog surprises in computer experiments.

In an effort to get as good an intuition as possible about computer experimentation, it would be useful to have a catalog of surprises that have occurred in computer experiments. An example would be a situation where some simple rule appears to be giving a definite form of behavior—but suddenly, after a great many steps, a different form of behavior emerges. Another example would be when, after many different initial conditions, one initial condition suddenly gives very different behavior than those that precede it (e.g. page 289). The general phenomenon of undecidability makes it ultimately inevitable that such surprises much occur. But as a practical matter in doing computer experiments, it is important to get a sense of how common they are. Can one make any kind of statistical argument that gives any sense of their frequency?

👁️ | 🕒 | **NKS page:** — | **Fields:** computer experiment; exposition

4

Systems Based on Numbers

■ The Notion of Numbers

Understand the connection between Gray codes and bitwise operations.

Page 901 mentions that in a standard Gray code, the number at position i is given by $BitXor[i, Floor[i/2]]$. Are there other simple combinations of bitwise functions that also lead to Gray code sequences? Is there a cellular automaton rule that yields Gray code sequences? Or some other simple bitwise iterative program?

👍 | 🕒 | **NKS page:** 901L | **Field:** discrete math

■ Elementary Arithmetic

Study the pattern of digit sequences in negative bases.

Explanation...

👍 | 🕒 | **NKS page:** 902R | **Field:** computer experiment

Study the properties of multiplicative digit sequences.

Explanation...

👍 | 🕒 | **NKS page:** 902R | **Fields:** number theory; discrete math

Analyze powers of 3 in base 2.

Extend the characterizations of the pattern of digits of powers of 3 in base 2. Do statistical analyses of the distribution of sizes of triangles. Look for the frequencies at which different sequences and patches occur in the pattern. Try to understand the results using number theoretical methods.

👍 | 🕒🕒 | **NKS pages:** 120, 903L | **Fields:** computer experiment; analysis

Study properties of $\text{FractionalPart}[(3/2)^n]$.

There have been attempts since the 1940s to show equidistribution for this quantity. Make a statistical study of the total sizes of numbers, and of leading digits. Try to find any regularities that may exist.

 |  | **NKS page:** 903L | **Fields:** computer experiment; statistical analysis; number theory

Study the effect of initial conditions on $(3/2)^n$.

With an initial condition u , the result at step n is just $u (3/2)^n$. How are specific digits affected by digits in u ? Can one find a u that makes, for example, the leading digit of $\text{FractionalPart}[u (3/2)^n]$ have some particular behavior? How does the distribution of $\text{FractionalPart}[u (3/2)^n]$ vary with u ? It appears that for some u , the distribution goes to zero below some value. How large can that value be made? What type of numbers are needed to achieve this?

 |  | **NKS page:** 903R | **Fields:** computer experiment; number theory

Find classes of h for which definite statements can be made about $\text{FractionalPart}[h^n]$.

Pisot numbers such as `GoldenRatio` are known to lead to a non-uniform distribution of values—since their powers eventually get arbitrarily close to integers. Find other classes of numbers that yield other kinds of non-uniform distributions. Look at numbers constructed by explicit algebraic or transcendental operations, and see what distribution of values they give.

There have been attempts for many years to find numbers that will provably lead to uniform distribution. Investigate numbers whose digit sequences are given by simple programs (for example being nested), and see whether any may lead to provably uniform distribution.

 |  | **NKS page:** 903R | **Fields:** computer experiment; number theory

Find functions $a[n]$ for which $\text{FractionalPart}[a[n], 1]$ has different empirical distributions.

Most choices of $a[n]$ seem to give either rather trivial distributions of possible values, or distributions that seem uniform (as in cases like $\text{Sqrt}[n]$ or $n \text{Log}[n]$). Look at different symbolic forms for $a[n]$, including one built from algebraic and transcendental functions, and see which of them lead to apparently uniform distributions, and which do not. Try to find cases where the distribution is ultimately smooth, but not uniform.

 |  | **NKS page:** 904L | **Field:** computer experiment

Investigate the cellular automaton for the $3n+1$ problem.

As described on page 904, the $3n+1$ problem can be formulated as a question about the long-time behavior of a particular cellular automaton. Try using the same methods of experimentation and analysis on this cellular automaton as I have discussed for other cellular automata. Try to establish versions of computational irreducibility for this cellular automaton. A major achievement would be a demonstration of universality, which might show that the $3n+1$ problem is in some sense formally undecidable.

 |  | **NKS page:** 904R | **Fields:** computer experiment; analysis; constructions

Make a systematic study of the 5/2 number theoretical system on page 123.

For what initial conditions does the system show repetitive behavior? Are there special initial conditions that yield regular behavior, but with growth? When growth is observed, is it always statistically at the same asymptotic rate? Are there initial conditions for which one can prove that growth will always continue to occur?

👉 | 🕒 | **NKS pages:** 123, 124 | **Fields:** computer experiment; number theory

Investigate further the reversible 3/2 system of page 905.

The reversible system of page 905 is a rather interesting variant of the $3n+1$ problem, with rich phenomenology. Try to characterize different classes of behavior obtained from different initial conditions. For what initial conditions is there not unbounded growth?

👉 | 🕒 | **NKS page:** 905L | **Field:** computer experiment

Investigate the long-term behavior of the reversal-addition system with initial condition 512.

Does the system ever repeat? Try to run for as many steps as possible. See if there are approaches that allow proofs of behavior to be made.

What is the distribution of the lengths of the regular sequences on the left and right-hand sides? Why does the rule act in a seemingly quite local way on the digit sequences? What happens if one perturbs a few digits in the pattern? How long do their effects last? What kinds of localized structures exist? What makes localized structures get created, and destroyed?

👉 | 🕒 | **NKS pages:** 126, 127 | **Fields:** programming; computer experiment; analysis

Make a systematic study of reversal-addition and related systems.

Search for initial conditions that lead to repetitive behavior with periods other than 4. Search for initial conditions that lead to apparently unbounded growth. Look at bases other than 2. Look at digit sequence operations other than pure reversal (perhaps built from RotateLeft, Partition, Transpose, Flatten, etc). Search for cases with intermediate rates of growth.

👉 | 🕒🕒 | **NKS pages:** 125, 905 | **Field:** computer experiment

Make a systematic study of iterated run-length encoding.

See what happens with different initial conditions, and slight changes of the run-length encoding function (e.g. adding or subtracting 1 from the length or element). Is there an easy way to understand that the behavior obtained will always ultimately be nested? Can one for example find a tree representation of the sequences and transformations to them that will make this obvious?

👉 | 🕒 | **NKS page:** 905R | **Fields:** computer experiment; analysis

Study digit count sequences.

Does the digit count sequence ultimately have a regular nested structure? What happens with different initial conditions, or different bases? Under what circumstances are what sequence growth rates observed?

 |  | **NKS page:** 905R | **Field:** computer experiment

Study systems based on iterated bitwise operations.

Rule 60 corresponds to iterating $\text{BitXor}[2n, n]$. Make a systematic of $\text{BitXor}[a n + b, c n + d]$. Under what circumstances do the results correspond to additive cellular automata? Study also iterations of combinations of other bitwise functions, such as BitOr . When is it possible to find the outcome after t steps with less effort than just running the system and seeing what happens?

 |  | **NKS page:** 906L | **Field:** computer experiment | **ID:** 906-bitwise | **See Also:** 871-bitwise

■ Recursive Sequences

Study in more detail recursive sequences (e) through (h) from page 130.

What regularities exist in the sequences? What is the tree of values accessed to find a given value? What is the origin of the seemingly continuous large-scale structure, particularly in case (g)? Is there some approximate version of the sequence that captures that structure? Try to follow each of the sequences as far as possible. Is there some kind of transform (Fourier, wavelet, etc.) that decomposes the sequence in a useful way?

 |  | **NKS pages:** 130, 906R | **Fields:** discrete math; computer experiment

Prove that recursive sequences (e) through (h) from page 130 never become undefined.

At least for the first million steps, none of the sequences (e) through (h) ever try to access values below $f[1]$. Will they ever try to access such values? First, try to find good empirical evidence for or against this. Then try to find a rigorous proof. The proof will undoubtedly be rather different for each of the cases. Quite possibly it will end up relying on statements in number theory—that may or may not already have been proved, and that may or may not be provable, say within the standard Peano axioms.

 |  | **NKS pages:** 130, 906R | **Fields:** discrete math; proof; programming

Systematically explore variants of the recursive sequences from page 130.

Investigate what happens with different initial conditions, or different rules. Many initial conditions and rules lead to functions that quickly become undefined. How common is it for the functions to stay defined? Can different initial conditions with a single rule lead to very different forms of behavior? See what happens when one systematically changes parameters in the rules—or the structure of the rules. What kinds of behavior occur? How common is repetitive behavior? Nested behavior? Apparently random behavior? What overall growth rates occur?

 |  | **NKS page:** 130 | **Field:** computer experiment

Look for simple rules that yield recursive sequences corresponding to recognizable functions.

Case (d) is closely related to the *DigitCount* function. Can one find a simple rule for recursive sequences that gives exactly the *DigitCount* function? What about other number theoretical functions? One could imagine conceivably a function that goes down whenever its argument is prime, and up when it is not. Is there a simple rule for such a function? One can consider both trying to construct recursive sequences that correspond to recognizable functions, and simply searching for them.

 |  | **NKS pages:** 130 etc. | **Fields:** construction; computer experiment; number theory

When does evaluation scheme affect whether recursive sequences remain defined?

Whether a definite value can be found from a recursive definition can in general depend on the evaluation scheme used to apply the recursive definition. *Mathematica* by default applies one such scheme. It is quite easy to implement other schemes that potentially allow more functions to remain defined. Do there start to be many more underlying rules that work if one uses different evaluation schemes? One should be able to represent different evaluation schemes in terms of directed graphs formed from trees in which some branches recombine.

 |  | **NKS page:** 906L | **Fields:** recursive function theory; computer experiment

How can one extend recursive function definitions to continuous numbers?

What is the continuous analog of the Ackermann function? The symbolic forms of the Ackermann function with a fixed first argument seem to have obvious interpretations for arbitrary real or complex values of the second argument. But is there a general way to extend these kinds of recursive definitions to continuous cases? Given a way to do this, how does it apply to recursive definitions like those on page 130? What happens to all the irregularities when one is between integer values? Or is it only possible to find simple continuous generalizations to functions that show fundamentally simple behavior? Can this be used as a characterization of when the behavior is simple?

 |  | **NKS page:** 906 | **Fields:** functional analysis; recursive function theory

Study the behavior of the primitive recursive function $r[z, r[s, r[s, r[s, p[2]]]]$.

As discussed on page 908, this is the first primitive recursive function to show complex behavior. Study its behavior in detail. Try to relate it to known number theory. Try to find a formula for its zeros. The last picture on page 908L shows that there are definite statistical regularities in the values obtained. Try to understand these, perhaps with some statistical approximation. There seem to be hints of nesting in the behavior. Is there in fact exact nesting at some level?

 |  | **NKS page:** 908L | **Fields:** discrete math; number theory; recursive function theory

Systematically catalog the behavior of simple primitive recursive functions.

Given the enumeration of primitive recursive functions on page 907, what is the distribution of different kinds of behavior? What is the distribution of different growth rates? When do different growth rates first appear?

 |  | **NKS page:** 908 | **Field:** computer experiment

Use explicit enumeration to look for functions that are not primitive recursive.

Given an explicit enumeration of the primitive recursive functions, it becomes conceivable to look empirically for the "first" functions that are not primitive recursive. Essentially, one has to be able to go far enough in the enumeration that one can ensure that a given sequence of values will not occur in any of the primitive recursive functions. It may require some fairly intricate symbolic manipulation of the sequence of possible primitive recursive functions to prove that certain sequences can never be generated from any of the functions. But if this could be done, it would give some interesting new insight into the boundary between primitive and general recursive functions.

👉 | 🕒 | **NKS pages:** 907 etc. | **Fields:** computer experiment; recursive function theory; programming

Study the diagonal non-primitive recursive function of page 908.

The diagonal function on page 908 is a rare example of a function that does not grow rapidly, yet can be shown not to be primitive recursive. Study what regularities and irregularities it has. Is the reduction of the function modulo 2 also not primitive recursive?

👉 | 🕒 | **NKS page:** 908R | **Fields:** computer experiment; recursive function theory

Is there a good continuous analog of the primitive recursive functions?

The definition of the primitive recursive functions, with successor functions and primitive recursion, seems quite tied to integers. Is there a way to generalize things so that a continuous analog of the primitive recursive functions can be defined?

👉 | 🕒 | **NKS page:** 907 | **Fields:** functional analysis; recursive function theory

Study and generalize the Ulam sequences of page 908.

Try to characterize the randomness in the fluctuations of the Ulam sequence shown. See whether the $13.5n$ estimate continues to hold. Try to find a statistical argument for it. Look at variants of the Ulam sequence, allowing more than two numbers in the sum, or more than one way to form the result.

👉 | 🕒 | **NKS page:** 908R | **Fields:** computer experiment; analysis

■ The Sequence of Primes

Find the simplest formula involving Floor that generates primes.

A pure polynomial cannot generate primes. But a formula that combines polynomials with Floor presumably can. Find the simplest such formula that successfully generates the primes.

👉 | 🕒 | **NKS page:** 909L | **Fields:** computer searching; number theory | **See Also:** XXXX | 162

Find a formula to describe the outcomes of decimation systems with $k > 2$.

For $k=2$, there is a fairly simple description of the outcome of a decimation system. It seems likely that the outcomes of decimation systems with $k > 2$ can also be described in terms of standard number-theoretical functions.

👁👁 | 🕒-🕒🕒 | **NKS page:** 909L | **Fields:** number theory

Apply randomness tests to number-theoretical sequences.

Pages 133, 135 etc. show curves generated from various well-studied problems in number theory. The curves have many seemingly random features. Apply various standard randomness tests to these sequences. Try to understand any deviations from randomness in terms of number-theoretical results about the underlying problems. Continue the curves as far as possible, and try to find regularities that are not explained by known number theory.

👁👁 | 🕒🕒 | **NKS page:** 133, 135, etc. | **Fields:** number theory; statistics

Look for empirical evidence from curves related to perfect numbers.

Whether odd perfect numbers exist is probably the single oldest unsolved problem in number theory. Curve (b) on page 135 in a sense provides an empirical view of the problem: if the curve ever reaches zero for odd n , then there is an odd perfect number. I traced various properties of this curve up to n of 500 million (see page 911R). Continue for larger n . Study such issues as what values do occur, and how often. Try to explain the results, perhaps using statistical number theory arguments.

👁 | 🕒-🕒🕒🕒 | **NKS pages:** 135, 911R | **Fields:** computer experiment; analysis; number theory

Look for empirical evidence from curves related to the Goldbach conjecture.

Proving the Goldbach conjecture is a famous unsolved problem in number theory. For the conjecture to be false, curve (e) on page 135 would have to reach the axis. This certainly does not look likely to happen from the data plotted on page 135. First, continue the data much further, and see whether the trends on page 135 continue. Then look at the distribution of fluctuations in the curve, and try to assess the "probability" that the curve goes as far as the axis, making the Goldbach conjecture false. Make pictures analogous to 135 (e) for the case of three primes—where a generalization of the Goldbach conjecture has been proved—and see how they compare to the usual two-prime case.

👁👁 | 🕒-🕒🕒🕒 | **NKS pages:** 135, 911L | **Fields:** number theory; programming

Study analogs of primes associated with non-rectangular arrays.

With n objects, the only case where one fails to be able to lay them out in a non-trivial rectangular array is when n is prime. What are analogous classes of numbers for other forms of arrays? For trapezoidal arrays, the analogous numbers are powers of two. What about other two-dimensional or higher arrays with definite layouts, say based on crystallographic arrangements?

👁 | 🕒 | **NKS page:** 911L | **Fields:** computer experiment; number theory

Study the spectra of number theoretical functions.

As shown on page 911L, number theoretical functions often give fairly complex spectra. Try to understand these spectra in terms of known number theoretical results. What special features do they have? (Compare flat spectra discussed on page 1081.)

 |  | **NKS page:** 911L | **Field:** number theory

Make a further study of iterated aliquot sums.

Page 911R discusses the old problem of what happens when one iteratively forms the aliquot sum $Total[Divisors[n]] - n$. For most initial n , the outcome is simple. But for some n , one seems to get a sequence that never repeats. The first potential example is $n=276$. Trace this example longer. Look at other examples. Try to prove anything about the possible halting times for the sequences.

 |  | **NKS page:** 911R | **Fields:** programming; number theory

■ Mathematical Constants

Run larger classes of randomness tests on digit sequences like those of π .

Billions of digits of constants like π have now been computed. Some basic statistical tests have been done on them. But many more could be done. In particular, it would be worth trying to enumerate all possible simple programs of particular kinds (such as cellular automata), and using these to process the digit sequences. (See page 597.)

 |  | **NKS pages:** 139, 912R | **Fields:** computer experiment; statistics

Search for mathematical constants with regularities in their digit sequences.

One knows that rational numbers have repetitive digit sequences. But are there other mathematical constants which can be specified by fairly simple symbolic expressions in *Mathematica*, yet which have digit sequences that show regularities? I have done quite a few searches, and not found anything. But vastly more could be done, and a positive result would be of great interest.

I would set up a system for enumerating possible trees of expressions containing specified *Mathematica* functions. I might start with elementary functions like *Power*, *Exp* and *Sin*. But then I would also consider all sorts of special functions. For each expression—say $\text{AiryAi}[5]^{(1/5)}$ —I would then compute a numerical approximation to, say, 50 digits. Then I would see whether this digit sequence has regularities of the kind I am looking for. In the unusual cases where it does, I would then go on and compute progressively more digits.

Every so often, there will be a "hit" with these procedures, when some expression happens to correspond, say, to a rational number. In effect, each one of these "hits" represents an identity among particular values of mathematical functions—potentially one that has not been seen before.

One knows that purely periodic digit sequences occur only in rational numbers. As possible regularities, I would consider nesting (as revealed, say, by pointer-based encoding data compression), as well as the kinds of regularities that can be detected by statistical tests—both standard ones, and ones based on enumerating possible simple programs. Although probably less significant, a potential test of a search system would be to look for digit sequences that have the longest runs of identical digits.

 |  | **NKS page:** — | **Field:** large-scale searches

Study carries in procedures to compute nth digits directly.

In the mid-1990s various schemes for computing nth digits in constants like π were found. But so far as I can tell, all these schemes ultimately have to rely on the fact that certain chains of carries will eventually die out. Study the distributions of the relevant chains of carries, and try to prove that they will in fact always die out.

 |  | **NKS page:** 912L | **Fields:** number theory; computer experiment

Search for numbers that can have their nth digits computed directly.

There are certain classes of sums that have the property that the nth digit in their results can potentially be found without having to compute other digits. Enumerate classes of these sums, and see which of them correspond to combinations of known mathematical constants. A straightforward way to do this is just to use the *Sum* function in *Mathematica*. I expect that quite a few sums that allow nth digits to be computed directly will be able to be evaluated symbolically using *Sum*. If so, try to understand what types of combinations of mathematical constants allow direct computation of nth digits.

 |  | **NKS page:** 912L | **Fields:** number theory; computer experiment

Try to develop automated procedures for generating provably transcendental numbers.

Given particular mathematical constants, it has been notoriously difficult to prove that they are transcendental. But the methods used to prove that constants like π and e are transcendental are more than a century old. Try to set up parametrized versions of these proof methods. By changing "parameters" in the proofs, it would then potentially be possible to generate other provably transcendental numbers.

 |  | **NKS page:** 912R | **Field:** number theory

Study simple conditional integer recurrences.

Page 141 shows that a very simple integer recurrence rules involving a conditional can generate the digits of square roots. What do other similar recurrences do? Study for example $\{r, s\} \rightarrow \text{If}[r > s, m1 \cdot \{r, s\} + v1, m2 \cdot \{r, s\} + v2]$. For which matrices m_i and vectors v_i is nontrivial behavior seen?

 |  | **NKS page:** 141 | **Fields:** computer experiment; number theory

Find simple rules for computing the digits of mathematical constants.

The procedure on page 141 for computing digits of square roots is extremely simple—and probably the simplest possible for square roots. Find analogously simple procedures for computing digits of other mathematical constants. Consider other cases involving a fixed number of registers, to whose values simple operations are applied, with the operations chosen for example by simple conditionals. Is there a procedure of this type for computing $\text{Log}[2]$? Consider doing a search through all possible rules of certain kinds.

 |  | **NKS pages:** 141, 913L | **Fields:** number theory; constructions

Are there nested digit sequences that correspond to known mathematical constants?

Enumerate nested digit sequences and compare them with a large library of combinations of known mathematical constants. Are there any matches? I have tried doing small versions of such searches, and have found nothing. But I would not be surprised if by searching trillions of cases, something could be found.

 |  | **NKS page:** 913L | **Field:** large-scale searches

Find summation formulas for numbers with nested digit sequences.

Page 913 gives an explicit summation formula for the number with digit sequence given by the substitution system $\{1 \rightarrow \{1,0\}, 0 \rightarrow \{0,1\}\}$. Most likely similar formulas can be set up for all Sturmian nested sequences. But what about other ones? Can there be a summation formula, say, for the Thue-Morse sequence?

 |  | **NKS page:** 913L | **Fields:** number theory; symbolic mathematics

What is the fastest way to find the nth digit in a concatenation sequence?

Page 913R gives slightly complicated procedures for quickly computing nth digits in concatenation sequences. Are there faster procedures? Concatenation sequences are simple enough in their specification that it seems conceivable that useful lower bounds could be proved on computing their nth digits.

 |  | **NKS page:** 913 | **Fields:** number theory; algorithms

Study concatenation sequences based on various number sequences.

Page 913 shows concatenation sequences based on digits of successive integers. One can also consider concatenation sequences based on successive values $f[i]$ of any function. Consider simple polynomials, as well as various number-theoretical functions. Look at the cumulants formed from the resulting digit sequences. It will presumably always show some form of nesting for polynomials. Characterize its dimension and other properties in terms of f . How do their properties depend on the base used to represent the numbers?

 |  | **NKS page:** 913 | **Fields:** computer experiment; number theory

Study concatenation sequences based on various number representations.

Page 913 shows concatenation sequences based on ordinary digit sequences, and Gray codes. What happens with other number representations? Consider for example fractional bases, or any of the representations from page 560. Presumably the patterns produced are always ultimately nested.

 |  | **NKS page:** 913 | **Field:** computer experiment

Study numbers with digit sequences having simple run-length representations.

Page 914 mentions a small result I got that a number whose digit sequence consists of successively longer alternating runs of 0's and 1's can be expressed in terms of EllipticTheta functions. What about other numbers whose digit sequences have simple run-length representations? When can "closed form" representations be found for them? Consider numbers where successive run lengths are given, say, by polynomials, number-theoretical functions, or perhaps quantities reduced modulo k .

 |  | **NKS page:** 914L | **Fields:** number theory; symbolic mathematics

Catalog typical leading digit distributions.

The leading digits of quantities like $n!$ and $\text{Fibonacci}[n]$ are known asymptotically to follow a definite logarithmic law. Those of $\text{Prime}[n]$ and $\text{Log}[n]$ are known not to. Investigate many different functions, represented, say, as simple *Mathematica* symbolic expressions, and see which seem to follow a logarithmic law for their leading digits, and which do not. Among cases where the logarithmic law is not followed, are there a small number of other laws that frequently apply?

 |  | **NKS page:** 914L | **Fields:** computer experiment; number theory

Search for regularities in continued fraction sequences.

Generate very long continued fraction sequences for mathematical constants, and try to find regularities in them. As of version 5.0, *Mathematica* should be able to generate a billion terms in the continued fraction expansions of constants like π , at least if given appropriate time and computer environment. Do this for a variety of constants, and search for regularities in the resulting sequences. Setting up procedures to find regularities is made slightly more complicated by the fact that the elements in the sequences are integers of arbitrary size. A first step is just to look at the distribution of sizes, and see if it follows the standard logarithmic law. Assuming it does, one can use the logarithmic law to map the numbers into what should be a uniform distribution in the interval 0 to 1. Then one can apply standard statistical tests of randomness. Perhaps more interesting, however, is to look for nested structure in the original sequence of integers. One should try looking at continued fractions for a variety of kinds of constants—not only roots and elementary functions, but also various kinds of special functions. (That it could be worthwhile to look at special function is definitely supported by the fact that arithmetic sequences occur in the continued fractions arise of BesselI functions.)

🔍 | 🕒 | **NKS page:** 914 | **Field:** computer experiment

Study numbers whose continued fraction terms are given by polynomials and other simple functions.

Continued fraction sequences given by arithmetic sequences correspond to numbers given by BesselI functions. As I say on page 914, I suspect that any polynomial continued fraction sequence will correspond to a number given by some kind of generalized hypergeometric function. Work this out in detail. Try to find under what circumstances there are simpler representations for the numbers found.

What happens for continued fraction sequences given by functions other than polynomials? What about sequences generated by recurrence relations? When do the sequences correspond to numbers that can be represented "in closed form"?

🔍🔍🔍 | 🕒 | **NKS page:** 914R | **Fields:** number theory; special functions

Study numbers with nested continued fractions.

Page 914 gives a particular case (due to Jeff Shallit) in which a nested sequence of continued fraction terms corresponds to a number with a simple digit sequence. Look at other nested sequences, and see when similar results can be obtained. Try enumerating simple nested sequences, then apply `FromContinuedFraction` to each of them, and see when there are regularities in the digit sequences of the results.

Use symbolic summation and other symbolic methods to try to find "closed forms" for numbers obtained from nested continued fractions. Failing this, try comparing numbers with nested continued fractions against a library of values of combinations of mathematical constants.

🔍🔍🔍 | 🕒 | **NKS page:** 914R | **Fields:** computer experiment; symbolic computation

Study the continued fractions for numbers whose digits form concatenation sequences.

Page 915 shows that the continued fractions for concatenation-sequence numbers have some rather remarkable properties. Try to understand these properties. In particular, try to find accurate estimates for the large continued fraction terms that occur, or for the distribution of sizes of terms.

 |  | **NKS page:** 915L | **Fields:** discrete math; number theory

Investigate what types of numbers can be generated by what Egyptian fractions.

Find a characterization of numbers that can be generated by Egyptian fractions in which $a[n]$ is given by functions from some definite class. Polynomials should give PolyGamma-type results. What about polynomials in $n!$ or k^n ? Sum in *Mathematica* should be able to get results in many cases. Try to find a general characterization of the types of functions that appear in the results.

 |  | **NKS page:** 915R | **Fields:** symbolic computation; special functions

Investigate nested radical representation of numbers.

It appears that digits 0, 1, 2 are sufficient to represent all numbers in the interval 1 to 2 using nested radicals. Try to prove this.

Look at infinite digit sequences with simple structures, and try to find closed forms for the numbers they represent. (Page 915 gives the result for a constant sequence of digits.) Try to find mathematical constants that have simple representations in terms of nested radicals.

 |  | **NKS pages:** 915R, 916L | **Fields:** symbolic computation; algebra

Investigate operator representations of integers.

I was quite surprised by what I figured out about representing integers using various operators. Make a systematic investigation of what operators are capable of representing all integers. For the bitwise operators I show, find exact and asymptotic results for how many operators it can take to represent an integer n . Try to characterize what integers are "algorithmically simple" for what classes of operator representations. Investigate what happens if one allows a unary operator as well as a binary one, and if one allows two binary operators, or a ternary operator.

 |  | **NKS page:** 916 | **Fields:** computer experiment; discrete math

Study operator representations for larger classes of numbers.

The concept of operator representations can be extended for numbers other than integers. A case similar to integers is Gaussian integers. Other cases include various forms of algebraic numbers, as well as general real or complex numbers. Find operators for which trees exist that yield all numbers in the required class. In general, infinite trees will be required. Perhaps there are relations to the trees studied for example in connection with surreal numbers.

 |  | **NKS page:** 916 | **Field:** number theory

Characterize what numbers are common on the web.

If one does a search (say in Google) for almost any 5-digit number, one gets a great many hits. As the length of the number one is searching for increases, the number of hits decreases. For some numbers, though, there are still many hits. Try to characterize the numbers for which there are many and few hits. Does the leading-digit distribution law apply?

👁️ | 🕒 | **NKS page:** 916R | **Fields:** whimsy; statistics

What numbers can be obtained by integrating or summing multivariate rational functions?

Explanation...

👁️👁️ | 🕒 | **NKS page:** 916R | **Fields:** symbolic computation; number theory; special functions

■ Mathematical Functions

What are the spacings between zeros of sums of three sine functions?

There is a simple formula for the zeros of a sum of two sine functions—and a definite theory of their distribution (see page 147). But as shown on page 917, sums of three sine functions seem much more complicated. Try to derive the spacing of zeros for $\sin[x] + \sin[\sqrt{2}x] + \sin[\sqrt{3}x]$. Can one even explain the origin of the cusps? Under what circumstances does such a spacing distribution of zeros have what level of continuity?

👁️👁️ | 🕒 | **NKS pages:** 146, 917 | **Fields:** algebra; number theory