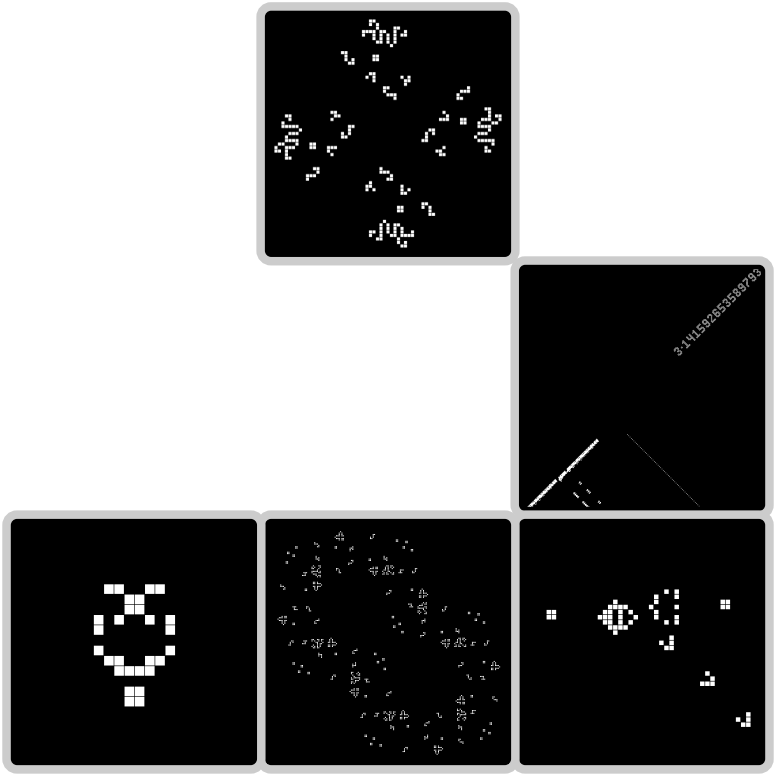
Conway's Game of Life

Mathematics and Construction



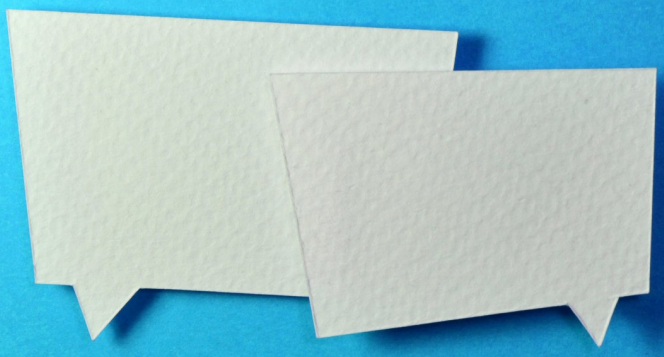
Nathaniel Johnston and **Dave Greene**

CONWAYLIFE.COM

To John Horton Conway
For giving us 50 years of Life.

Contents

Appendices and Supplements



Preface

The Goal

This book provides an introduction to Conway's Game of Life, the interesting mathematics behind it, and the methods used to construct many of its most interesting patterns. This book generally tries to avoid presenting patterns in isolation or as historical notes, but rather tries to guide the reader through the thought processes that went into creating them in the first place.

While we will largely follow the history of the Game of Life as we go through the book, we emphasize that this is *not* the primary goal of the book, but rather it is a by-product of the fact that most recently-discovered patterns build upon patterns and techniques that were developed earlier. Instead, the goal of this book is to demystify the Game of Life by breaking down the complex patterns that have been developed in it into bite-size chunks that can be understood individually.

Intended Audience

While this book does not have any formal mathematical or computer science prerequisites, it is written at a level aimed at students of at least a first-year undergraduate university level. Some high-school-level topics like logarithms, the "floor" function $f(x) = \lfloor x \rfloor$ for rounding numbers down, the binary representation of a positive integer, and summation notation like $\Sigma_{k=1}^{n} k^2$ for adding numbers, are used frequently and without much explanation. A basic understanding of how mathematical proofs and computer programming work is also expected. That is, we expect a certain level of mathematical and computer science maturity from the reader, but no specialized knowledge of either topic.

More specifically, we prove some simple theorems about the Game of Life in Chapters 1 through 5. These proofs do not use any specialized proof techniques or expect the reader to have any specific university-level mathematical knowledge, but rather just expect the reader to be able to follow a logical argument. Similarly, we introduce a programming language for building computer programs out of Life circuits in Chapter **??**, so that material will be easier to master if the reader has had prior exposure to computer programming.

Somewhat more advanced mathematical topics that we make use of are summarized in Appendix **??**, though they are typically introduced very gently in the main text as well, and we only require a very surface-level understanding of them. We make use of the greatest common divisor, the least common multiple, and Bézout's identity when discussing oscillator periods in Chapter **??**,

so we introduce these tools in Appendix **??**. Infinite series make brief appearances at the end of Chapter **??** and in Section **??**, though the reader is not expected to really have any familiarity with them or understand convergence issues. Finally, big-O notation is used to discuss the growth rate of patterns in Sections **??** and **??**, so we introduce this concept in Appendix **??**.

How to Use

Conway's Game of Life is an extremely visual game, so this book makes very liberal use of figures throughout, particularly when new patterns or techniques for creating patterns are introduced. However, the Game of Life is best observed in motion, which makes static images in a textbook less than ideal as learning tools. In order to help present the motion of patterns a bit better, we do five things:

- Figures are presented with alive cells in black and dead cells in white, but furthermore we use a gradient from blue to orange to denote cells that were alive in past generations of the pattern. Bright blue cells were just alive, whereas cells that are orange were alive in the more distant past (roughly 75 generations or more—see Figure 1).

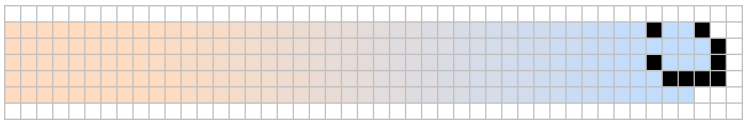


Figure 1: An object moving to the right with a gradient behind it indicating how long ago it was in each location. White cells were never alive, black cells are currently alive, blue cells were alive recently, and orange cells were alive long ago (roughly 75 generations or more).

- If we really wish to emphasize what a pattern looks like in different generations, all generations of interest will be displayed, along with arrows that specify how many generations have passed. For example, if we want to clarify exactly what the pattern in Figure 1 does as it moves from left to right, we might display it as in Figure 2.

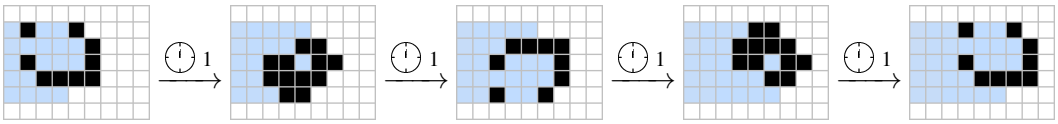


Figure 2: The same object as in Figure 1, but displayed in a more explicit fashion. This image shows how the pattern changes every time 1 generation elapses.

- We use colors to highlight various pieces of patterns, and we maintain a consistent coloring scheme throughout the book. Light pastel colors like aqua, magenta, light green, yellow, and light orange are used to highlight *around* objects—cells in these colors are dead, but highlight a region in the Life plane consisting of cells that all serve some common purpose or logically make up one "object" (see Figure 3). On the other hand, darker colors like dark green, dark orange, and dark red are used to highlight certain live cells. Dark green is typically used to highlight the input to some reaction, dark orange is typically used for the output of the reaction, and dark red is used otherwise (see Figure 4).

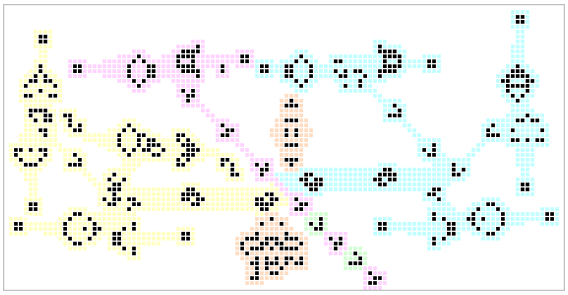


Figure 3: A glider gun made up of several different component reactions that are highlighted in different colors. In particular, gliders are created by a gun highlighted in magenta, lightweight spaceships are created by guns highlighted in aqua and yellow, and those lightweight spaceships are merged into the original glider stream by oscillators highlighted in light orange.

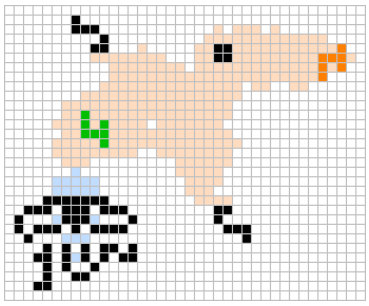


Figure 4: A dark green "Herschel" comes in from the left and creates the dark orange Herschel on the right. Cells highlighted in blue are constantly changing due to being part of an oscillator, whereas cells that are light orange are usually dead, except when the Herschel passes through.

- In the electronic version of this book, almost every figure is actually a clickable link that will open a text file containing RLE or Macrocell code for the displayed pattern (go ahead, click on any of the figures above, or even one of the five images on the cover page). This code can be copy and pasted into Life simulation software like Golly (golly.sourceforge.net) so that it can be explored and manipulated.[1] Note that clicking on figures may not work in certain PDF viewers (such as the viewers built into web browsers), so we recommend using Adobe Acrobat Reader (get.adobe.com/reader) to read this book digitally.
- RLE, LifeHistory, or Macrocell codes for all of the patterns displayed in the book are also available at the book's website (conwaylife.com/book). Furthermore, the patterns can be viewed and manipulated right on that website as well via any modern web browser, without downloading any additional software.

Exercises

We strongly encourage the reader to work through this book's many exercises that can be found at the end of each chapter. For this reason, it is extremely important to either download Life software or use the book's website to view and edit patterns while making your way through the book—Life is meant to be played, not just watched, and many of the exercises simply cannot be solved without the assistance of Life simulation software.

Roughly half of the exercises are marked with an asterisk (∗), which means that they have a solution provided in Appendix **??**.

Acknowledgments

The authors are indebted to dozens of people who opened the world of Conway's Game of Life to them. Rather than acknowledging the people who discovered the reactions and patterns that we discuss here, they are credited in footnotes and figures throughout the book.

We extend thanks to Nicolay Beluchenko, Steven Eker, Mark Niemiec, and Michael Simkin for helpful conversations about content of the book. Thanks to Andrew Trevorrow and Tomas Rokicki for creating the open-source cross-platform CA editor and simulator Golly (golly.sourceforge.net), without which many of the patterns discussed in this book would not have been discovered. Thanks to Chris Rowett for creating LifeViewer (lazyslug.no-ip.biz/lifeview), which is used on this book's website (conwaylife.com/book) to display patterns and make them interactive. Thanks

[1] For an explanation of how RLE or Macrocell code works, see conwaylife.com/wiki/Run_Length_Encoded or conwaylife.com/wiki/Macrocell.

to Velimir Gayevskiy and Mathias Legrand for the *Legrand Orange Book* LaTeX template from LaTeXTemplates.com.

Finally, the authors would like to thank their wives Kathryn and Melanie for tolerating them during their years of mental absence glued to this book, and their parents for encouraging them to care about both learning and teaching. Many thanks also to Mount Allison University for giving the first author the academic freedom to pursue a project like this one.

I Classical Topics

1. Early Life

> I used to feel guilty in Cambridge that I spent all day playing games, while I was supposed to be doing mathematics. Then . . . I realized that playing games *is* math.
>
> John H. Conway

Conway's Game of Life[1] is a process that takes place on an infinite square grid, where each square can be in one of two states: *alive* or *dead* (depicted via black and white squares, respectively, in this book). The squares (which we typically call *cells*) then evolve in discrete timesteps (called *generations* or *ticks*) according to the following two rules:

- If a cell is alive, it survives to the next generation if it has 2 or 3 live neighbors; otherwise it dies.
- If a cell is dead, it comes to life in the next generation if it has exactly 3 live neighbors; otherwise it stays dead.

We note that these rules are applied to every square in the grid simultaneously, and a "neighbor" in these rules refers to any of the 8 cells that it touches either along a side or at a corner, as in Figure 1.1. As an example of how these rules work, consider what happens to the straight line of 4 alive cells depicted in Figure 1.2. The leftmost and rightmost cells in the line both only have one live neighbor, so they die, while the two central cells above and below the line each have exactly 3 live neighbors, so they come to life. This leaves us with a 3×2 rectangle of live cells, so we say that the line of 4 live cells is a *parent* of the 3×2 rectangle of live cells, or equivalently that the 3×2 rectangle is a *child* of the line of 4 cells. We then apply the evolution rules again: this time, the two central cells in the rectangle are overpopulated and die, while the dead cells to their immediate left and right have exactly 3 live neighbors and thus come to life.

After this point, if we apply the evolution rule again, nothing changes; each of the live cells have 2 live neighbors and thus live to the next generation, and no dead cell has 3 live neighbors, so they all stay dead. A pattern like this that remains unchanged from one generation to the next is called a *still life*.

[1] Named for its inventor, John H. Conway.

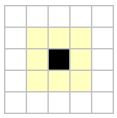


Figure 1.1: A live cell (in black) in the middle and its 8 neighbors (in yellow).

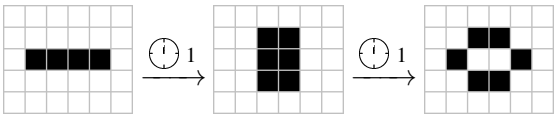


Figure 1.2: A line of four alive cells takes two generations to evolve into a stable object called a *beehive*. The 3×2 object in the middle is sometimes called a *pre-beehive*.

As we saw in this example, there was no input on our part once the pattern started evolving: we just applied the game's rules and passively watched what happened from one generation to the next. Instead, the "game" in Life is the challenge of finding new and interesting patterns that evolve in unique and unexpected ways, and that is what this book is about. We will see patterns that move back and forth periodically between a finite number of different configurations, patterns that move through the Life grid over time, and patterns that create an infinitely-growing family of other patterns. We will collide patterns with each other to create more complicated patterns, which we will then erase with even more patterns. We will construct logical circuitry with Life objects that allow us to simulate arbitrary computation within the Life universe, and we will construct patterns that do remarkable things like list the prime numbers or print out the decimal digits of π. And all of this will work simply based the two simple life and death rules we described earlier.

This all leads to a very natural question: why those rules? Why not have a dead cell come to life only in the case that it has exactly 4 live neighbors? Why not have a live cell stay alive if it has exactly 1, 2, 4, or 7 live neighbors? Indeed, there are $2^{18} = 262,144$ distinct rules that can be constructed simply by specifying different numbers of live and dead neighbors that lead to a cell staying alive or coming to life. However, the following three properties make Life special (but by no means unique):

- Its rules are *simple*. For example, having a live cell stay alive if it has exactly 2 or 3 live neighbors is a more "natural" rule than having a live cell stay alive if it has exactly 1, 2, 4, or 7 live neighbors. This can be justified a bit by arguing that, to model something like a biological system, a cell should die of overcrowding if it has "too many" neighbors (e.g., since there won't be enough resources to support all of the live cells), and it should die of isolation if it has "too few" neighbors. The exact threshold for "too many" and "too few" is debatable, but pinned down at least somewhat by the next point.
- It is *chaotic*. Many rules (e.g., almost any rule in which a cell is born when it has 2 live neighbors) cause far too many births for anything to stabilize, so it is almost impossible for us to construct interesting objects. On the other hand, most rules in which cells are *not* born when they have 3 (or fewer) live neighbors typically lead to patterns dying off extremely quickly. Life strikes a good balance of patterns typically staying alive but not overtaking the entire grid.
- It is *historical*. This is perhaps a bit of an unsatisfying reason, but part of the interest in Life simply comes from the fact that historically it is the most well-studied rule, and it is fun to see how far we can push one very well-studied rule, rather than dividing our attention and making moderate progress on multiple rules.

Nonetheless, other rules are sometimes studied as well, and for brevity they are typically described using a *rulestring* of the form Bx/Sy, where we replace "x" by all numbers of live neighbors that lead to the **b**irth of a dead cell, and we replace "y" by all numbers of live neighbors that lead to the **s**urvival of a live cell. For example, the Game of Life is described by the rulestring B3/S23. We will comment on other rules from time to time (especially in Chapter **??**), typically to highlight how they contrast with Life.

There are also many more exotic ways to change the Game of Life beyond just changing the numbers of neighbors that cause cells to be alive. For example, we could have considered a *neighbor* of a cell to only be one of the 4 cells that shares one of its sides as in Figure 1.3, not just a corner (this 4-cell neighborhood is called the *von Neumann neighborhood*, whereas the 8-cell neighborhood used by Life is called the *Moore neighborhood*). We could have considered a hexagonal grid instead of a

square one, or a 1D or 3D grid instead of a 2D one. We could have constructed the game in such a way that not only the number of live neighbors matters, but also their relative positions—such rules are known as *isotropic rules*. These are all potentially interesting modifications to make, and they fall under the general umbrella of *cellular automata*, but we will not consider them any further in this book. Instead, as we emphasize one final time, our goal is to take the Game of Life cellular automaton itself and push it to its farthest limits.

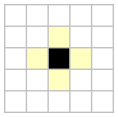


Figure 1.3: A live cell (in black) in the middle and the 4 cells in its von Neumann neighborhood (in yellow).

1.1 Our First Technique: Random Fumbling

There are three main techniques used to construct objects in Life that behave in interesting and unusual ways:

1) We can write a computer program that searches for patterns with particular properties;
2) we can combine different already-known objects in such a way as to create new composite objects; or
3) we can put some random garbage on the Life board and evolve it, with the hope that something interesting pops out.

Option (1) typically requires a fair bit of effort, as well as some knowledge of what exactly it is that we're searching for. Similarly, option (2) is not yet possible for us since we are just starting out with Life and do not yet know of any objects that can be combined in an interesting way. We thus start with the extremely not clever option (3): we try evolving a bunch of random patterns and see what is left of them after their chaos dies down.[2] Our first example is presented in Figure 1.4, which is a random assortment of live cells that takes 116 generations before stabilizing into several distinct objects. Random starting configurations like this one are sometimes called *soup*, and the objects that they leave behind are called *ash*.[3]

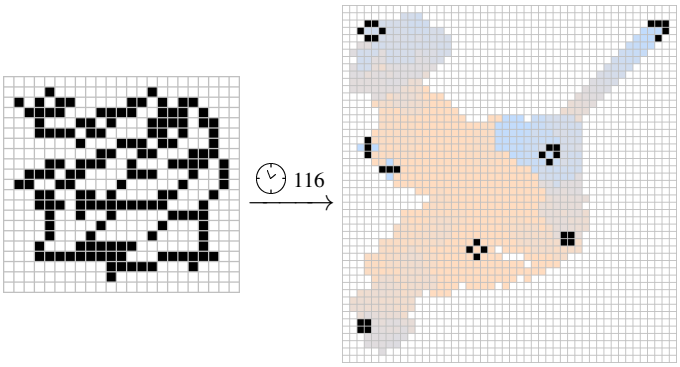


Figure 1.4: Evolving random junk is a decent way to find some small patterns to get us started in Life. The debris on the right introduces us to several still lifes, our first *oscillator*, and the *glider*.

In the ash, we see the beehive that we were introduced to earlier, as well as three other still lifes (i.e., patterns that do not change from one generation to the next). We also see an object called the *blinker* that rotates itself by 90 degrees every generation. Patterns like this one, which cycle between

[2] This is not only *our* approach to getting started with Life, but was also the historical approach: many of the patterns found in the first year or two of Life were found just by evolving many small configurations and looking at the results.

[3] The term "ash" refers to the fact that it is what is left after a pattern stops "burning".

finitely many different configurations, are called *oscillators*, and the configurations that they take on in individual generations are called *phases*. The most exotic object that we see in the ash is the one that is traveling by itself toward the top-right corner of the Life plane. An object that moves through the plane on its own is called a *spaceship*, and this particular one is called the *glider*.

Of course, there is nothing remotely special about the particular random configuration that we started with; we could generate other random starting configurations and see what types of ash pop out of them as well. In fact, Life is so chaotic and unpredictable that we could just change a single cell of the first configuration that we used, and we will likely get a completely different set of ash. Indeed, Figure 1.5 demonstrates the drastic change that can often be made by altering just one cell in a pattern: by just changing one cell from alive to dead, we have made a new pattern that takes almost 3,000 generations to stabilize and results in ash that has over 20 times as many live cells. This new ash also has three additional still lifes in it that were not present in the previous ash. A summary of the ash objects we have seen so far is given in Figure 1.6.

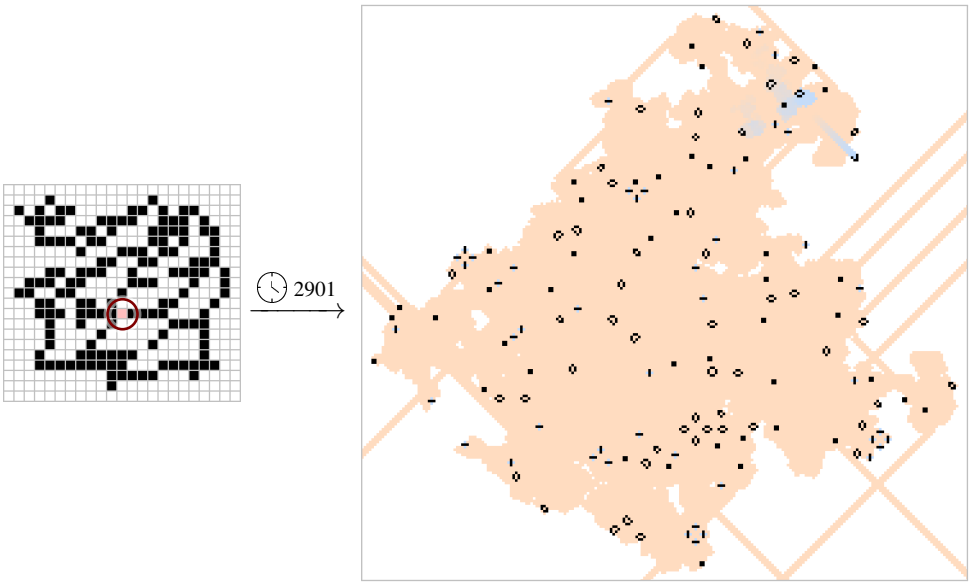


Figure 1.5: Changing just a single cell (circled on the left in red) from alive to dead in our random starting configuration causes its evolution to become completely different. It now takes over 25 times as long to stabilize and its ash contains three new types of still lifes (ponds, ships, and loaves).

Still lifes, oscillators, and spaceships are the three most basic types of objects that we will study in Life, and they form the building blocks of all of the more complicated patterns that we will construct. At this point though, there are some natural questions that we can ask about them. For example, we already saw the blinker, which oscillates back and forth between 2 phases every 2 generations. Do there exist oscillators that take longer to return to their original configuration? In other words, do there exist oscillators with *period* larger than 2?

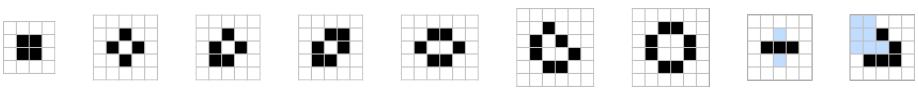


Figure 1.6: From left to right: seven still lifes, called the *block*, *tub*, *boat*, *ship*, *beehive*, *loaf* and *pond*, an oscillator called the *blinker*, and a spaceship called the *glider*. All of these objects frequently appear in the ash left behind by chaotic patterns.

To answer this question, we simply continue not being clever: we evolve more and more random configurations of junk and record new objects that appear in the ash as we find them. Most of the new objects that we will find by doing this are additional still lifes, but new oscillators crop up from time to time as well. For example, the oscillator depicted in Figure 1.7, which is called the *pulsar* and has period 3, appears rather frequently in random ash for its size.

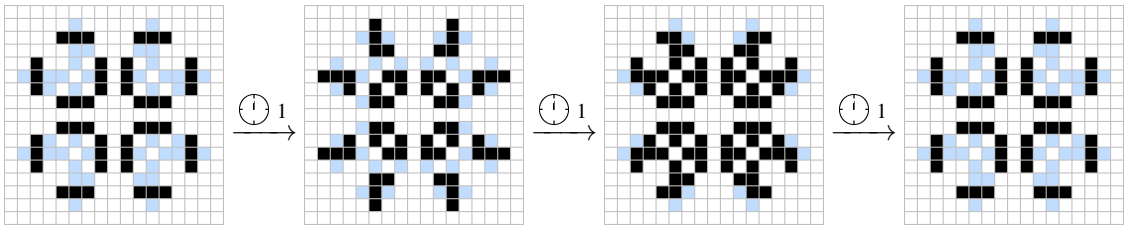


Figure 1.7: A period 3 oscillator called the *pulsar*.

Some other oscillators that appear reasonably often in random ash are depicted in Figure 1.8. Of particular note is the *pentadecathlon*, which has a surprisingly large period of 15. This oscillator behaves somewhat differently than the other ones we have seen, as it pulsates and changes in size as it moves through its period—a property that will be very useful for us later on.

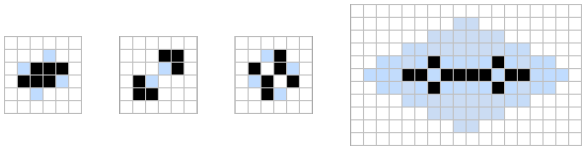


Figure 1.8: From left to right: three oscillators with period 2, called *toad*, *beacon* and *clock*, and a period 15 oscillator called *pentadecathlon*.

In the process of finding these oscillators, it is extremely likely that we will have also come across some other commonly-occurring objects, such as the *lightweight spaceship* (or *LWSS* for short) depicted in Figure 1.9. This is another spaceship, but this one moves orthogonally (i.e., directly north, south, east, or west), unlike the glider, which moves diagonally.

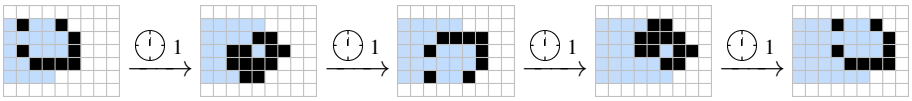


Figure 1.9: The *lightweight spaceship*, which has period 4 and moves orthogonally to the right by 2 cells every 4 generations.

Slightly more rare than the lightweight spaceship are the *middleweight spaceship* (or *MWSS*) and the *heavyweight spaceship* (or *HWSS*),[4] displayed in Figure 1.10. There are a handful of other objects that a lucky Life enthusiast might see while evolving random junk, but for the most part these are the "standard" naturally-occurring objects. Some examples of random soups that generate more exotic objects are presented in Exercises 1.1 and 1.7.

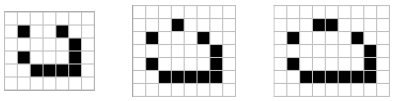


Figure 1.10: From left to right: a lightweight, middleweight, and heavyweight spaceship. They all have period 4 and travel to the right 2 cells every 4 generations.

1.2 Common Evolutionary Sequences

We have seen plenty of interesting patterns so far simply by looking at the results of running random soups until they stabilize. However, we can also learn a lot by carefully watching the evolution of soups as they happen, as not only are there commonly-occurring stable ash objects, but there are

[4] In keeping with the "____weight spaceship" naming convention, the glider was sometimes called the *featherweight spaceship* in the early days of Life, though this name is very rarely used now.

also commonly-occurring unstable objects that explode and drive the evolution of those patterns. For example, one such object is the *T-tetromino*,[5] which is a 4-cell object that appears (for example) in generations 206, 395, and 568 of the evolution of the soup from Figure 1.5. This tiny object (displayed in Figure 1.11) explodes into an arrangement of 4 blinkers that is called a *traffic light*. In fact, three traffic lights (and part of a fourth) can be seen in the ash of Figure 1.5, and the common T-tetromino explosion is exactly why blinkers appear in this formation so frequently.

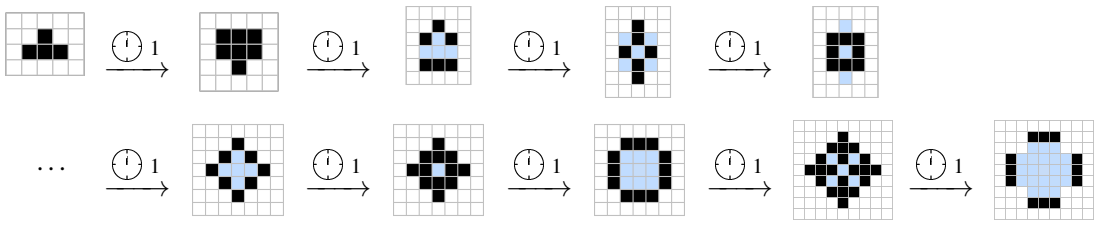


Figure 1.11: A *T-tetromino* (top-left) takes 9 generations to evolve into an arrangement of four blinkers called a *traffic light* (bottom-right).

The T-tetromino also illustrates how symmetry spontaneously forms within Life patterns. While it starts off with only left-right symmetry, after four generations it develops top-bottom symmetry, and then after one more generation it also develops diagonal symmetry. Since Life's rules do not care about the orientation of patterns, symmetry can never be broken once it has formed, which explains why all of the later generations of the T-tetromino also have 8-way symmetry, and why so many of the interesting and/or stable patterns that have been found in Life are symmetric.

Another unstable object that behaves similarly is the one displayed in Figure 1.12, which explodes over the course of 17 generations in order to create a commonly-occurring arrangement of 4 beehives called a *honeyfarm*. Appropriately enough, the 7-cell object that starts this evolution is called the *pre-honeyfarm* (as is any other small pattern that follows the same evolution). While only one honeyfarm appears in the ash in Figure 1.5, many other pre-honeyfarms appeared earlier (for example, at generations 749 and 1,159) and were subsequently destroyed.

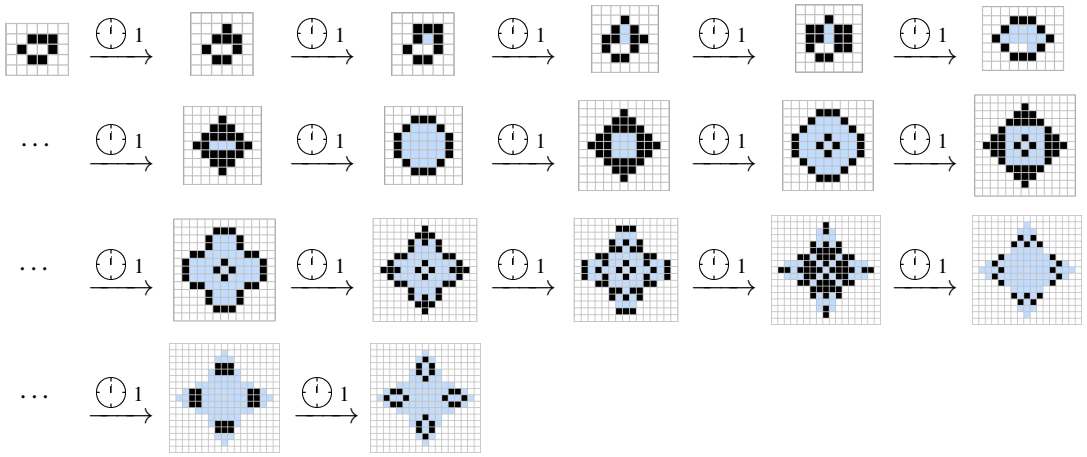


Figure 1.12: A *pre-honeyfarm* (top-left) takes 17 generations to evolve into an arrangement of four beehives called a *honeyfarm* (bottom-right).

A slightly messier explosion that is sometimes useful is the one that begins with the *stairstep hexomino* displayed in Figure 1.13. This object takes 63 generations to stabilize into an arrangement of four blocks called the *blockade*. However, since the blockade is somewhat larger than the traffic

[5] A *polyomino* is a pattern made up of orthogonally connected live cells, and a *tetromino* is a polyomino with 4 live cells. More generally, polyominoes with $2, 3, 4, \ldots, 8$ live cells are called *dominoes*, *triominoes*, *tetrominoes*, *pentominoes*, *hexominoes*, *heptominoes*, and *octominoes*.

light and the honeyfarm, and it also takes much longer to form, it is less commonly seen in ash. For example, the stairstep hexomino can be seen at generation 1,005 of the evolution of the soup in Figure 1.5, but it is interfered with before the blockade can form.

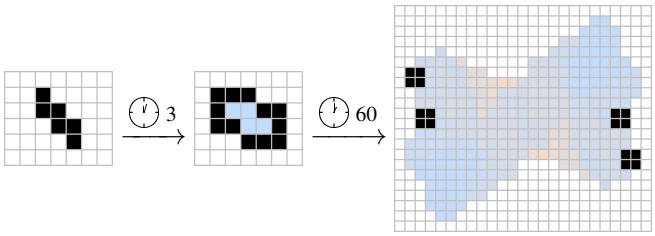


Figure 1.13: The *stairstep hexomino* (left) evolves into an arrangement of four blocks called the *blockade* in 63 generations. This evolution, and any of the intermediate patterns, are often called *lumps of muck*.

It is also often the case that this evolution begins with another small pattern, such as the third generation of the stairstep hexomino, rather than the stairstep hexomino itself. For this reason, this evolutionary sequence, and any of the patterns that appear during its 63-generation explosion, are given a common name: *lumps of muck*.

As one final example, consider the extremely common and extremely messy *pi-heptomino* displayed in Figure 1.14, which appears at generation 61 in the evolution of the soup in Figure 1.4 and numerous times in the evolution of the soup in Figure 1.5 (e.g., in generations 46, 164, 206, 208, 233, ...). While it does not evolve into any particularly interesting stable objects (a rather unremarkable collection of 6 blocks, 5 blinkers, and 2 ponds), it has the interesting property that it moves forward by 9 cells after 30 generations, while leaving behind some messy debris. While this is not quite useful by itself, it can be made useful by combining the pi-heptomino with other objects that destroy the debris before it gets in the way. By doing so, we can use the pi-heptomino to transmit a signal from one place in the Life plane to another or act like a spaceship. We will explore these ideas in Chapter **??** and Section **??**.

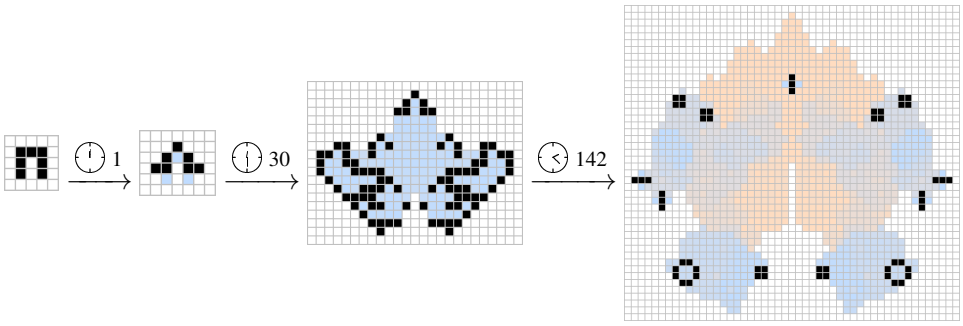


Figure 1.14: The *pi-heptomino* is a commonly-occurring unstable object that moves forward by 9 cells after 30 generations (compare the middle two patterns), but leaves behind debris that prevents it from moving ahead another 9 cells.

1.3 The Queen Bee

Sometimes, it is possible to "fix" unstable evolutionary patterns by tweaking them in some simple way, thus creating interesting objects that do not often appear naturally by themselves. Perhaps the most useful example of such a pattern is the *queen bee*, which is a commonly-occurring object[6] that reflects itself after 15 generations, but leaves behind a beehive in the process. It follows that after 30 generations, the queen bee is back where it started, but with an additional beehive on either side of it (see Figure 1.15). Shortly after 30 generations, the queen bee collides with the first beehive that it created, resulting in its self-destruction.

[6]For example, it appears in generation 1,185 of the soup in Figure 1.5.

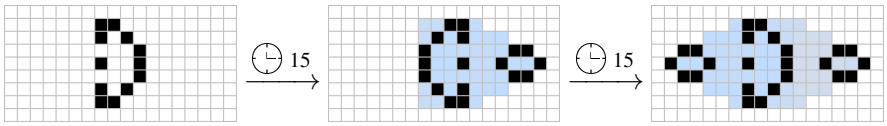


Figure 1.15: A *queen bee* (left) reflects itself and leaves a beehive behind every 15 generations. After doing this twice, the beehives start interfering with the queen bee, causing it to explode.

In order to prevent this self-destruction and turn the queen bee into something useful (an oscillator), we need a way to delete the beehives as they are created. Possibly the simplest way to do this is to use the reaction displayed in Figure 1.16, in which a block being placed next to a beehive results in the beehive being destroyed and the block surviving unharmed (we thus say that the block *eats* the beehive[7]).

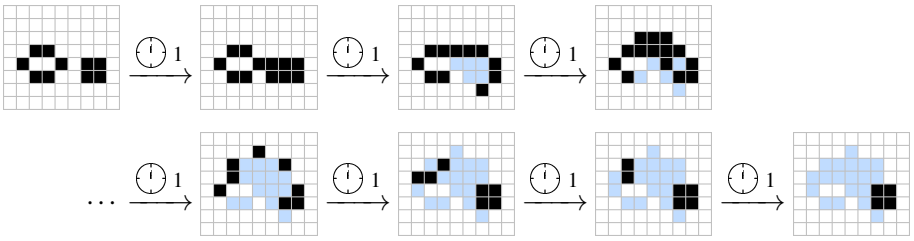


Figure 1.16: Placing a block next to a beehive destroys the beehive in 7 generations, without permanently damaging the block.

We can now simply paste these two reactions together to create an oscillator in which a queen bee (initially facing right) creates a beehive and is reflected to the left, then a block destroys the beehive while the queen bee creates another beehive and is reflected back to the right, then a block destroys the second beehive and the whole process repeats. This period 30 oscillator is called the *queen bee shuttle*[8] and is displayed in Figure 1.17.

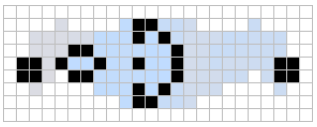


Figure 1.17: The *queen bee shuttle* is a period 30 oscillator constructed by combining the reactions in Figures 1.15 and 1.16.

The queen bee shuttle is our first example on an engineered object—a pattern that we didn't discover "naturally", but rather one that we specifically constructed by piecing together simpler reactions that we had observed. This is how most recent discoveries in the Game of Life have been made, and it is also the route that we will take through most of the later chapters in this book: we will combine simple patterns and reactions that we have already seen into more complicated patterns that serve a new purpose, and then we will combine those objects into even more sophisticated patterns, and so on.

The queen bee can also be stabilized by some objects other than blocks (see Exercise 1.10). Perhaps the most interesting and useful way to stabilize a queen bee is to use another (very carefully positioned) queen bee. If lined up and timed just right, the queen bees bounce off of each other, and instead of each producing a beehive, their collision produces a glider. This is remarkable, since we can then place stabilizing blocks on the other side of each queen bee so as to create a pattern that oscillates at period 30, but also creates an endless stream of gliders (see Figure 1.18). Patterns that

[7] We will look at eaters in more depth in Section **??**.

[8] The term "shuttle" typically refers to oscillators in which an unstable object moves back and forth between stabilizing objects (e.g., in this case, the unstable queen bee moves back and forth between the stabilizing blocks on either side of it).

create glider streams are called *glider guns*, and this particular one is called the *Gosper glider gun*.[9] This is the first pattern that we have seen that grows indefinitely (and indeed, it was the first such pattern ever to be discovered), and we will make extensive use of it when constructing more complex objects in later chapters.[10]

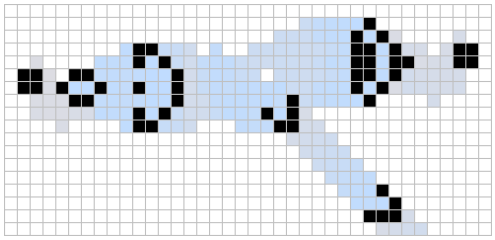


Figure 1.18: The *Gosper glider gun* creates a never-ending stream of gliders by bouncing two queen bees back and forth between two blocks (the object at the top-right is indeed a queen bee, just in a different phase than in the other figures).

1.4 The B-Heptomino and Twin Bees

Another commonly-occurring[11] unstable pattern that is very similar in flavor to the queen bee and the pi-heptomino is the *B-heptomino*. This object, like the pi-heptomino, has the interesting property that it behaves somewhat like a spaceship—after 10 generations it evolves into a copy of itself 5 cells forward, plus some extra junk behind it (see Figure 1.19). However, it is then quickly killed by its trailing debris before it has a chance to repeat this process.

Stabilizing the B-heptomino in order to turn it into a useful object is slightly more involved than it was for the queen bee, but fortunately nature does some of the hard work for us: B-heptominoes often occur in pairs, in a configuration that is called the *twin bees*,[12] which are displayed in Figure 1.20. When in pairs like this, the B-heptominoes survive longer than when on their own, and in fact they now reflect themselves, just like the queen bee did (albeit after 23 generations instead of 15) while leaving some junk behind in the process. Unlike the queen bee, the twin bees do not survive long enough to reflect themselves a second time, since the mess they leave behind explodes and destroys them very shortly after the first reflection.

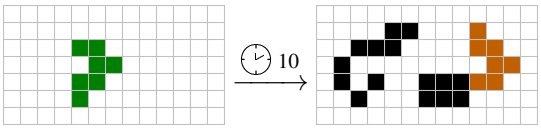


Figure 1.19: A *B-heptomino* (depicted in green on the left) takes 10 generations to move forward by 5 cells (in orange on the right) and create a bunch of junk behind it. The trailing junk subsequently destroys the B-heptomino, preventing it from traveling any farther.

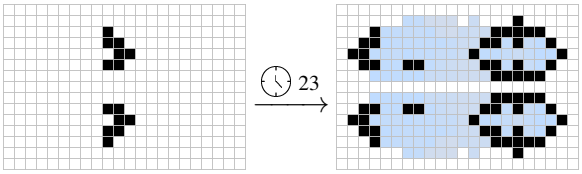


Figure 1.20: Twin bees reflect themselves and leave behind a chaotic mess every 23 generations (the two-cell objects in the middle of the image on the right die off in the next generation and thus have no effect).

In order to stabilize the twin bees, we take a cue from the queen bee and try placing blocks in such a way as to eat the mess that is left behind. This works very well, and the stabilization displayed in Figure 1.21 was already known in the very early days of Life.[13]

[9] Named after Bill Gosper, who found it in November 1970.

[10] Whether or not there are finite patterns that grow arbitrarily large was one of the major open questions in the early days of Life, and this glider gun's discovery earned Gosper a $50 reward from Conway himself.

[11] For example, it occurs at generation 106 of the soup from Figure 1.4 and generation 210 of the soup from Figure 1.5.

[12] The name "twin bees" refers both to the fact that many of its properties and uses are completely analogous to the queen bee, and also to the fact that it is made up of two B-heptominoes, so its name can be read as "twin Bs".

[13] This stabilization was found by Bill Gosper in 1971.

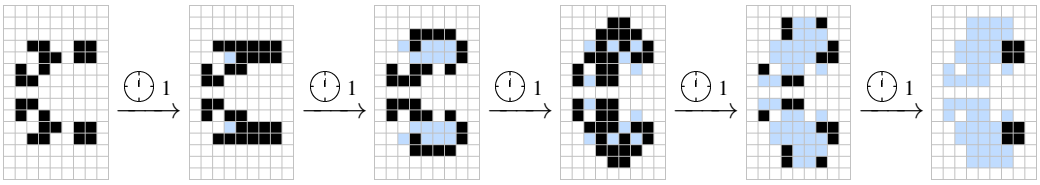


Figure 1.21: Placing two blocks next to the debris left behind by twin bees destroys that debris in 5 generations, while leaving the blocks intact.

We now do exactly what we did with the queen bee: we place blocks on *both* sides of the twin bees so that they reflect from the right to the left, their debris is destroyed, then they reflect from the left to the right, their debris is destroyed again, and then they repeat. This whole process takes $23+23=46$ generations to complete, and results in the period 46 oscillator known as the *twin bees shuttle* displayed in Figure 1.22.

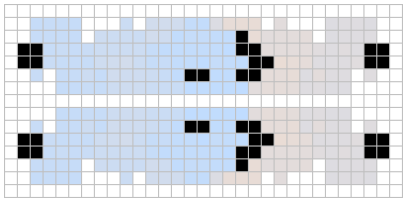


Figure 1.22: The *twin bees shuttle* is a period 46 oscillator constructed by combining the reactions in Figures 1.20 and 1.21.

As with the queen bee, there are many other ways to stabilize the twin bees shuttle besides using two blocks on each side as in 1.21, and some of these alternate stabilizations are investigated in Exercise 1.4. In fact, we can even collide two pairs of twin bees with each other to create a period 46 glider gun in almost the exact same way that we constructed the Gosper glider gun by colliding two queen bees. This new glider gun,[14] which we call the *twin bees gun*,[15] is shown in Figure 1.23.

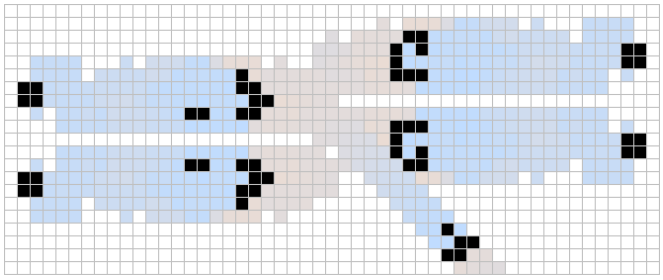


Figure 1.23: The *twin bees gun* is a period 46 glider gun that was constructed by bouncing two twin bees back and forth between each other and two stabilizing pairs of blocks. Note that the pair of objects in the middle-right are indeed twin bees, just in a different phase than the other figures.

1.5 The Switch Engine

One final commonly-occurring unstable pattern that is worth introducing at this point is the *switch engine*, which is the configuration of 8 live cells displayed in Figure 1.24. Just like the queen bee and twin bees, the switch engine leaves behind some debris and reappears later on in its evolution (this time after 48 generations), but in a different location and orientation. However, unlike the queen bee and twin bees, the switch engine does not return to its original position after repeating this reaction.

[14] This gun was also found by Bill Gosper.

[15] A slight variant of this gun was originally found in 1971 and called the *new gun*, since it was the first gun to be found after the Gosper glider gun. However, this name is woefully out of date at this point.

Instead, the switch engine continually moves farther and farther away from where it started, just like a spaceship.

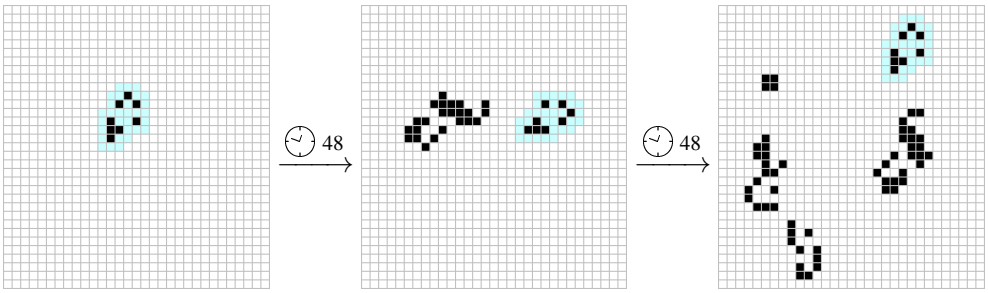


Figure 1.24: A *switch engine* (outlined in aqua) moves and reflects itself across the diagonal every 48 generations, but leaves behind some chaotic junk in the process. After 96 generations, the switch engine is back in its original orientation, but 8 cells northeast of where it started. The switch engine appears for the last time in generation 1,152, after repeating this entire process 12 times. The chaotic junk then finally collides with the switch engine and destroys it.

As with several other patterns that we have now explored, the switch engine will be destroyed by its debris if we do not stabilize it somehow. Because of the continued movement of the switch engine, we don't expect to be able to convert it into a stationary object like an oscillator or glider gun, like we did with the queen bee or twin bees from the previous sections, but rather we hope to turn it into some sort of "useful" moving object like a spaceship.

It turns out that turning it into a spaceship is actually somewhat tricky, so we defer that discussion to Section **??**. However, there are simple ways to stabilize it into a moving object that is *not* a spaceship. For example, if we place a block next to a switch engine as in Figure 1.25, then it evolves into an object called the *block-laying switch engine*—a switch engine whose debris settles down into an ever-lengthening stream of blocks (8 blocks every 288 generations).

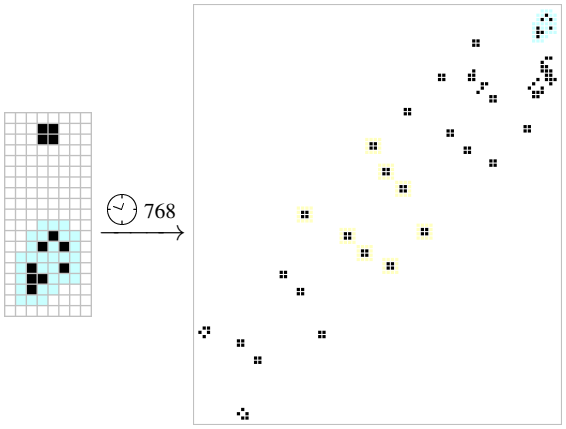


Figure 1.25: A switch engine (outlined in aqua) with a single block strategically placed next to it evolves into a *block-laying switch engine*—a pattern that repeatedly lays a configuration of 8 blocks (outlined in yellow) every 288 generations as it moves northeast.

An object like this one, which moves but leaves periodic junk behind it, is called a *puffer*. Another puffer can be constructed from a switch engine by instead placing a block next to it as in Figure 1.26. In this case, the pattern evolves into something called the *glider-producing switch engine*, which is a switch engine that leaves behind four blinkers, three blocks, two beehives, a boat, a ship, a loaf, and a glider every 384 generations. The glider travels in the same direction as the switch engine, but because the glider travels faster (at a pace of 3 cells every 12 generations, versus the switch engine's 1 cell every 12 generations), it does not take long for it to pass the switch engine.

The block-laying and glider-producing switch engines are remarkable for the fact that, not only do they grow infinitely, but they are also "natural" (as opposed to engineered, like the Gosper glider gun).

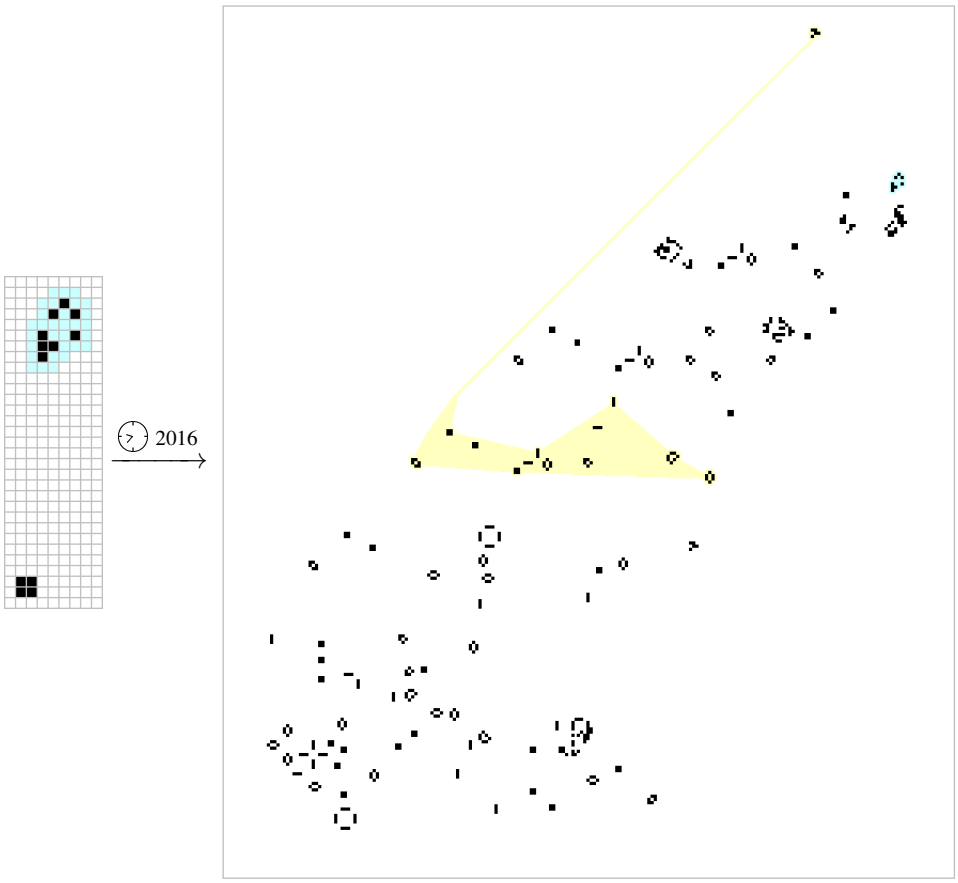


Figure 1.26: A switch engine (outlined in aqua) with a single block strategically placed next to it can also evolve into a *glider-producing switch engine*, which repeatedly lays a messy pattern of 4 blinkers, 8 still lifes, and one glider (outlined in yellow) every 384 generations, as it moves northeast.

In fact, puffers based on switch engines are the only infinitely-growing patterns that have ever formed as a result of randomly filling some portion of the Life plane and then evolving it (see Exercise 1.1). The ease with which these switch engine puffers appear (compared to other infinitely-growing patterns like the two glider guns we saw earlier) can be explained by noting that they have some very small predecessors, like the 12-cell objects shown in Figures 1.25 and 1.26,[16] whereas glider guns require a comparatively large number of "coordinated" live cells to form.

1.5.1 Arks

Perhaps an even more natural way to stabilize a switch engine is to use additional switch engines; after all, using a stationary object (like a block) to stabilize a moving object seems somewhat nonsensical, and only worked for us because the switch engine was able to turn the block into a moving stabilization (this is why the bottom-left corners of Figures 1.25 and 1.26 are irregular: the block itself wasn't the stabilization, but rather it just reacted chaotically with the switch engine's debris in such a way that it eventually created a moving stabilization).

On the other hand, using a moving object to stabilize another moving object (or using two moving objects to stabilize each other) is a common technique that we will use repeatedly. In this particular case, any puffer created by using two switch engines is called an *ark*, and dozens can be found just by trying different positions and phases of nearby switch engines. For example, if we place two switch engines next to each other as in Figure 1.27, the result is an ark that leaves behind a multitude of different objects, but does so in a periodic way: the same collection of still lifes, oscillators, and gliders is created every 576 generations, offset by 48 cells diagonally.

[16] These 12-cell predecessors can be turned into 11-cell predecessors by simply removing a single cell from the block.

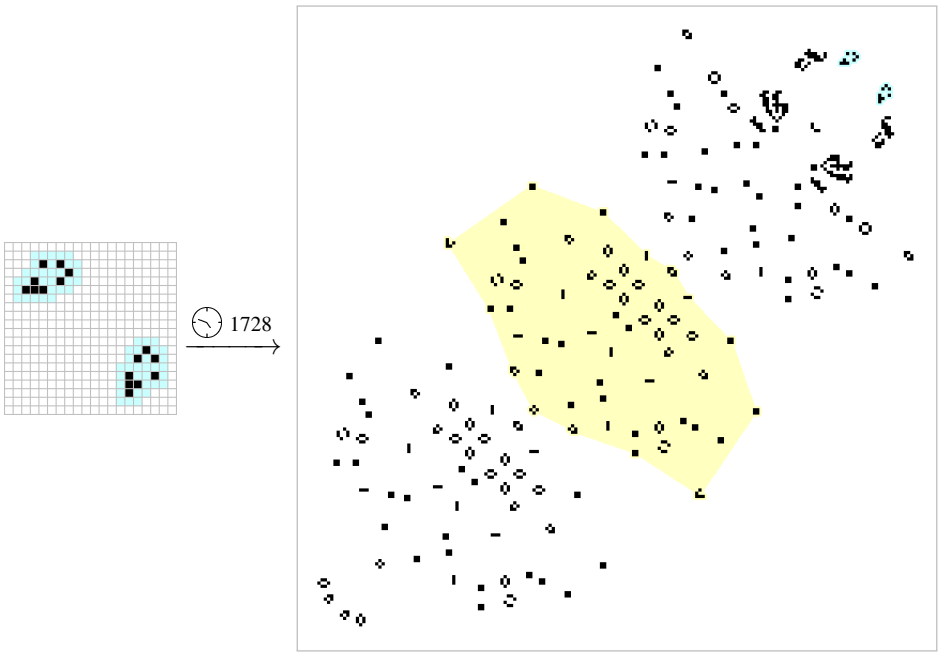


Figure 1.27: Two switch engines (outlined in aqua) can be used to stabilize each other and create an *ark*. In this particular ark, the switch engines leave behind two gliders, two toads, eight blinkers, and 42 still lifes (outlined in yellow) every 576 generations.

1.6 Methuselahs and Stability

Some of the patterns that we have investigated are extremely chaotic and take a long time to stabilize,[17] such as the B-heptomino and the random configuration depicted in Figure 1.5, which take 148 and 2,901 generations to stabilize, respectively. More extreme than either of these examples is the 8-cell switch engine, which takes 3,911 generations to stabilize—considerably longer than most other patterns with 8 or fewer cells. It seems somewhat natural to ask how far these examples can be pushed—that is, how long can a pattern of a given size take to stabilize? A pattern that takes exceptionally long to stabilize, relative to other similarly-sized patterns, is called a *methuselah*.[18]

Before looking at more specific examples of methuselahs, we make it very clear up front that this definition is very imprecise: there is no completely objective way to say that some patterns are methuselahs while other patterns aren't. Instead, the classification usually takes place simply by comparing the lifespan of the pattern with the longest known lifespan of similarly-sized patterns. Here, "size" may refer to either the number of live cells in the pattern, or to the area of its *bounding box*, which is the smallest rectangle that contains it.

No good search methods are known for finding methuselahs, so all of them have been found essentially by random guessing or exhaustive search. As an example, consider the *R-pentomino*, which is depicted in Figure 1.28. This pattern is interesting for the fact that it takes considerably longer to stabilize than any other pattern that fits with in a 3×3 bounding box, and also much longer to stabilize than any other polyomino with 5 or fewer live cells (see Exercise 1.3).[19] The evolutionary sequence that evolves from it is also extremely important—it produces a B-heptomino (well, actually a *B-heptaplet* that evolves in the same way) in generation 28, an important object called a *Herschel* in generation 48, and the very first glider that was ever observed in Life[20] in generation 69.[21]

We could similarly ask what the longest-lived pattern with n cells for $n = 6, 7, 8, \ldots$ are. The

[17] We will see shortly that properly defining what it means for a pattern to "stabilize" is very troublesome, but for now it just means that the pattern has broken down into non-interacting still lifes, oscillators, and spaceships.

[18] Named after the Biblical man Methuselah who reportedly lived the longest of any human—969 years.

[19] However, there are 5-cell patterns that are *not* polyominoes that last slightly longer, as demonstrated in Table 1.1.

[20] First noticed by Richard K. Guy in 1970.

[21] In fact, all of the unstable objects from Section 1.2 appear during the R-pentomino's evolution—see Exercise 1.5.

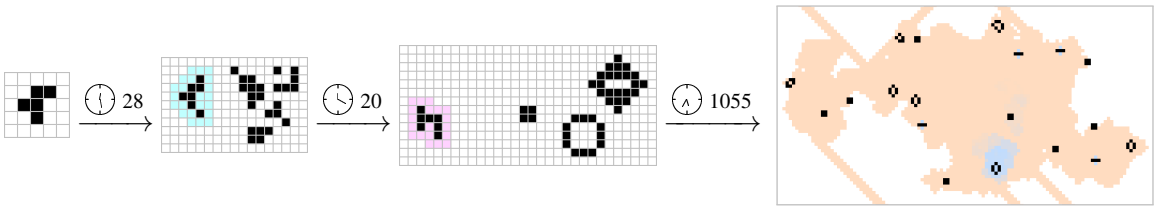


Figure 1.28: An R-pentomino takes 1,103 generations to stabilize—much longer than any other polyomino with 5 or fewer cells. Along the way, it produces numerous other important patterns such as the *B-heptaplet* highlighted in aqua and the *Herschel* highlighted in magenta.

longest-lived such patterns that are known are summarized in Table 1.1, as long as we restrict our attention only to patterns with reasonably small bounding boxes (which of course is subjective). However, these patterns are only known to be optimal for $n \leq 9$;[22] the longest-lasting known 10-cell methuselah in a small bounding box was not found until 2019.

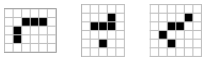
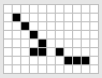
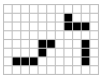
Cells	Methuselah	Name	Lifespan
5		R-pentomino grandparents	1,105
6		R-pentomino great-great-great grandparents	1,108
7		acorn	5,206
8		–	7,468
9		bunnies 9	17,410
10		bunnies 10b	17,431
11		–	23,334
12		–	23,801
13		Lidka	29,126

Table 1.1: The longest-lived known methuselahs with 5–13 cells. The methuselahs with 7 or fewer cells were all known by no later than 1971: the R-pentomino relatives were fairly well-known, so their exact discoverer is difficult to pin down, but the acorn was found by Charles Corderman. The methuselah with 8 cells was found by Tomas Rokicki in 2005, bunnies 9 was found by Paul Callahan in 1997, bunnies 10b was found by Nick Gotts in November 2019, and the 11-, 12-, and 13-cell methuselahs were found by Simon Ekström in May 2016, February 2017, and March 2017, respectively.

To highlight the difficulty of defining methuselahs in a rigorous way, note that there are 8- and 9-cell patterns that can be made to have lifespans as long as we like, since we can just aim a glider at a far away blinker or a block. However, these somewhat trivial examples can be avoided by requiring that a methuselah have a suitably small bounding box (which also rules out 8-cell methuselahs like

[22] Exhaustive computer searches for methuselahs with $n \leq 9$ cells were carried out by Paul Callahan in 1997, Tomas Rokicki in 2005, and Nick Gotts in 2019.

the one displayed in Exercise 1.15). If we go this route and focus on the bounding box of a pattern, rather than its number of cells, then we find patterns like the one displayed in Figure 1.29,[23] which fits within a 16×16 bounding box and has a lifespan of 47,575 generations, which is longer than any other known pattern of this size.

Another potential problem that arises when discussing methuselahs and stability comes from the fact that we already saw patterns with just 12 cells that grow indefinitely: the block-laying and glider-producing switch engines[24] of Section 1.5. This is not too difficult to take care of: even though these patterns grow forever, they do so in a regular and predictable way. For example, to know what a block-laying switch engine looks like in generation 5,000,000, we don't actually need to evolve it 5,000,000 times: we could simply move the switch engine the correct number of spaces to the northeast and paste a trail of blocks behind it. We could thus say, for example, that the block-laying switch engine stabilizes once it enters the periodic portion of its evolution where it repeatedly lays 8 blocks every 288 generations.

These problems might seem simple enough to overcome by tweaking our definitions carefully, but they are just the tip of the iceberg when it comes to what can go wrong when trying to rigorously define methuselahs and what it means for a pattern to stabilize. As yet another example, consider the remarkable 16-cell pattern[25] depicted in Figure 1.30, which takes 736,692 generations to stabilize.

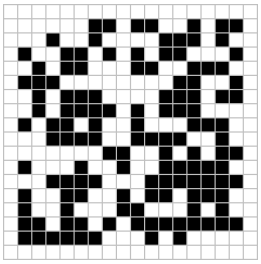


Figure 1.29: A methuselah that fits within a 16×16 bounding box and takes 47,575 generations to stabilize.

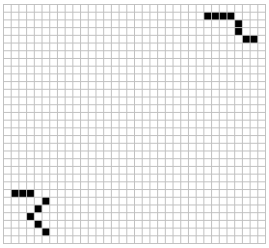


Figure 1.30: A 16-cell pattern that takes a whopping 736,692 generations to stabilize. The 8-cell objects at the top-right and bottom-left individually are just predecessors of the switch engine, so this pattern is an ark.

The "problem" with this pattern is that its non-stabilization is rather artificial; it evolves into adjacent block-laying and glider-producing switch engines, which interact in such a way as to fire a glider backward toward some debris every 2,304 generations. Every time this backward glider hits the debris, it transforms it into something slightly different and unpredictable. This process repeats more than 250 times before it breaks down and a glider destroys the debris enough that future gliders pass right through without touching it at all. However, by this point the pattern has been growing for over 700,000 generations into the huge mess shown in Figure 1.31.

While patterns like this one are certainly interesting, they are not quite in the spirit of what we want from methuselahs—even though the previous pattern did not completely stabilize for hundreds of thousands of generations, it was "mostly" stable for the vast majority of that time. Rather, we would like methuselahs to take a long time to stabilize based primarily on chaos that we cannot completely break down into simple reactions like gliders repeatedly hitting some debris. We will not try to make this precise, but rather just leave it as a phenomenon that we know when we see.

To briefly suggest some even more extreme examples of the weirdness that happens with stability, note that we will learn in Chapter **??** how to construct a pattern that computes the prime numbers. Should such a pattern be considered stable? Perhaps even more striking than this, in Section **??** we will then create a pattern whose long-term behavior we do not understand: it might continue to grow in a reasonably regular fashion indefinitely, or it might stabilize into a finite pattern. What we do know is that it continues to grow for at least its first $10^{2,500,000,000}$ generations, so if it does eventually

[23] Found by Adam P. Goucher in February 2019.

[24] This can easily be reduced to 11 cells by removing one of the cells in the block in either pattern.

[25] Found by Nick Gotts in 2005.

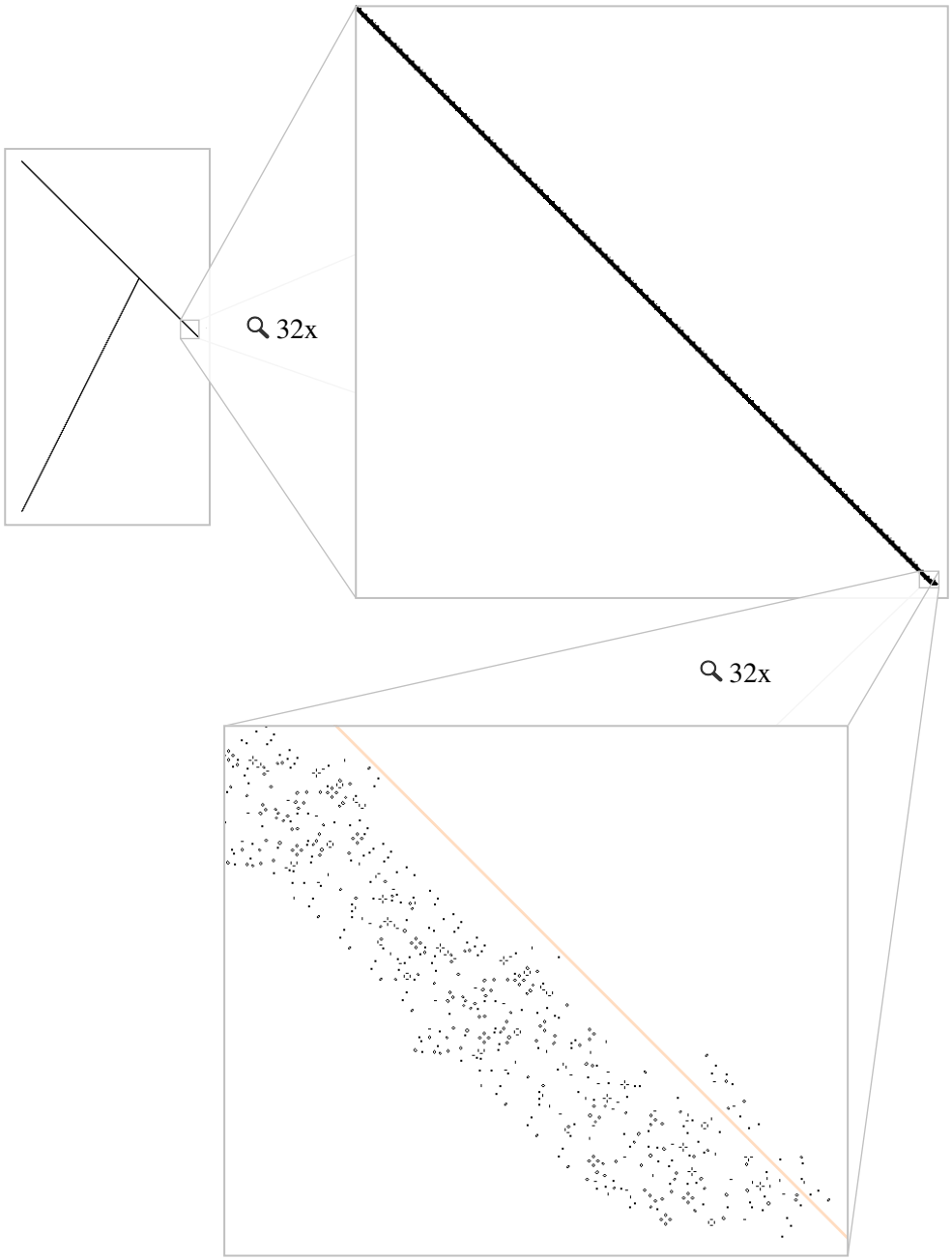


Figure 1.31: The pattern from Figure 1.30 leaves behind this absolutely massive ash after 736,692 generations. We have zoomed in by a factor of $32 \times 32 = 1{,}024$ on the corner of the ash that the pattern started growing from, which is also the corner that contains the debris that gliders are fired at throughout its extensive lifespan. The orange line on the bottom-right depicts the southeastward path taken by gliders that now (just barely) miss hitting the debris.

stabilize and remain finite, it must do so after that point (and we have no hope of actually simulating the pattern long enough to find out the answer).

Even worse than this, in Chapter **??** we will show that there exist patterns for which it is not *possible* to know, even in principle, whether or not they stabilize. In other words, there exist patterns with the unsettling property that, no matter how long we run them, and no matter how clever we are, there is absolutely no mathematical way to know if they will eventually stabilize.

1.7 Gardens of Eden

Oftentimes, a lot can be learned by considering Life in reverse: instead of investigating what a pattern evolves into, we investigate what evolves into it. That is, we look for parents (and grandparents, and great-grandparents, ...) of the pattern that we are interested in.[26] Some patterns, such as the block, have numerous parents (see Figure 1.32), which is part of the reason why they appear so frequently in the ash of random soups.

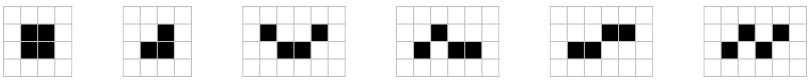


Figure 1.32: Six small parents of the block. From left-to-right, the first three of these objects are the block, *pre-block*, and *grin*, respectively, while the other three do not have names.

Other patterns, such as the clock, have relatively few parents and thus appear less frequently in random soups (recall that the clock has only six cells and period 2, yet appears less frequently than some much larger objects like the period 3 pulsar or the period 15 pentadecathlon). It seems natural to ask whether or not there exists a pattern that does not have even a single parent. That is, does there exist a pattern that cannot ever appear in the evolution of any other pattern, but rather can only ever appear in generation 0?[27] A pattern with this property is called a *Garden of Eden*,[28] and we begin by showing that they do indeed exist.

The rough idea of why Gardens of Eden must exist is that some patterns (such as blocks) have lots of parents, so there are not enough parents left over for all other patterns. To prove this explicitly, we will barely need to use anything more than the fact that blocks and pre-blocks both evolve into the same thing.[29]

Theorem 1.1 There exist Gardens of Eden in Conway's Game of Life.

Proof. Let $n \geq 1$ be an integer and consider all patterns that fit within a $6n \times 6n$ square on the Life grid. The contents of the central $(6n-2) \times (6n-2)$ square in generation 1 only depend on the contents of the original $6n \times 6n$ square in generation 0. This central $(6n-2) \times (6n-2)$ square contains $(6n-2)^2$ cells, each of which can be alive or dead, so there are $2^{(6n-2)^2}$ distinct patterns that could fill this central square. We will now show that, if n is large enough, some of these patterns must have no parent.

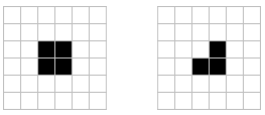


Figure 1.33: Two 6×6 tiles that evolve the same way no matter what is placed outside of the tiles.

To this end, partition the $6n \times 6n$ square into n^2 tiles, each of size 6×6. Each tile contains 36 cells, each of which can be alive or dead, so there are 2^{36} different tiles. However, the two tiles displayed in Figure 1.33 evolve in the same way regardless of what other tiles are placed around them. We thus conclude that in any pattern that uses the first tile, we could replace it by the second tile without changing the evolution of the overall pattern (see Figure 1.34). It follows that if we want to catalog all

[26] Strictly speaking, if a pattern has one parent then it must have infinitely many, since we could just add any pattern that dies in one generation far away. When discussing how many parents a pattern has, we will always ignore parents that have dying ash like this that has no effect on their evolution.

[27] Generation 0 refers to the starting configuration, before applying the Life rules to it. Generation 1 is the pattern that is obtained by applying the Life rules once, and generation n is the pattern obtained by applying the Life rules n times.

[28] This term was coined in the context of cellular automata by John W. Tukey in the 1950s, long before Conway's Game of Life was introduced.

[29] We could instead use two other objects that evolve into the same thing in the proof: we just use blocks and pre-blocks because they are so simple.

possible patterns that can be present in generation 1, we only need to consider patterns made up of one of $2^{36}-1$ (not 2^{36}) different tiles in generation 0.

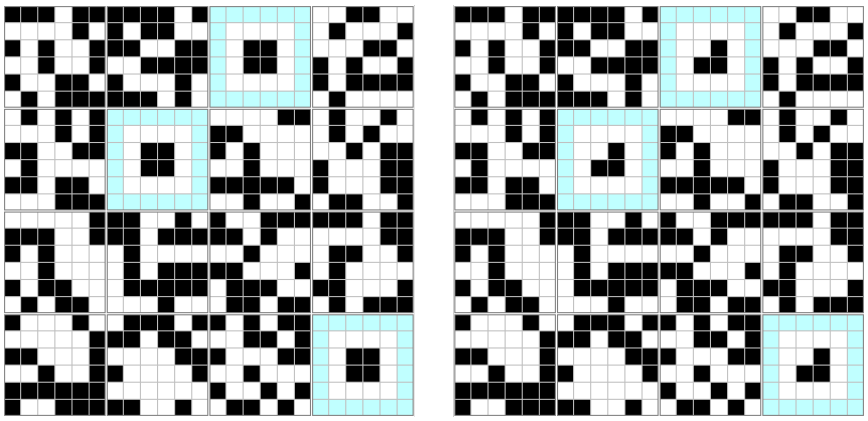


Figure 1.34: A $6n \times 6n$ pattern in the $n = 4$ case, broken down into 6×6 tiles. Each of the tiles containing only a block in the center (outlined in aqua) can be replaced by a tile with a pre-block in the center without affecting the evolution of the overall 24×24 pattern.

Since the $6n \times 6n$ square has n^2 of these tiles, there are at most $(2^{36}-1)^{n^2}$ different possible children (i.e., patterns present in generation 1). Since there are $2^{(6n-2)^2}$ distinct patterns that could fill the central $(6n-2) \times (6n-2)$ square, but at most $(2^{36}-1)^{n^2}$ of them appear in children of patterns in the $6n \times 6n$ square, all that remains is show that $(2^{36}-1)^{n^2} < 2^{(6n-2)^2}$ when n is sufficiently large. Taking the n^2-th root of both sides of the inequality reduces it to $2^{36}-1 < 2^{(6-2/n)^2}$, which is indeed true when n is sufficiently large, since we can make $2/n$ arbitrarily close to 0 and hence $2^{(6-2/n)^2}$ arbitrarily close to $2^{6^2} = 2^{36}$ (in particular, we can make it larger than $2^{36}-1$).[30] ■

It is worth noting that the method used in the proof of Theorem 1.1 is very general and applies to any cellular automaton for which two distinct (finite) patterns evolve into the same pattern.[31] Also, even though it is non-constructive, it can be used to find (extremely loose) bounds on how large Gardens of Eden must be—see Exercise 1.14.

Now that we know that Gardens of Eden exist, our next goal is to actually find an explicit example of one.[32] Unfortunately, our method of proof of Theorem 1.1 is not of much help here, since it only shows that Gardens of Eden exist in extremely large regions (the smallest n for which the inequality $(2^{36}-1)^{n^2} < 2^{(6n-2)^2}$ holds is somewhere around $n \approx 10^{12}$), and it does not actually tell us how to find one in such a region. Our method of construction is to build a Garden of Eden one cell at a time, repeatedly choosing the new cell that we add to the pattern so as to result in it having as few parents as possible. More explicitly, we build this Garden of Eden as follows:[33]

- We consider the two possible states of a single cell. Out of the $2^9 = 512$ possible configurations of the 3×3 square centered at this cell, 372 lead to the central cell being dead and 140 lead to it being alive. Thus we make that central cell alive, so as to have fewer parents.
- We then consider the two possible states of a cell that is adjacent to the starting (alive) cell. Out of the $140 \times 2^3 = 1,120$ possible configurations of the 4×3 rectangle centered at those two cells that lead to the first cell being alive, 703 lead to the second cell being dead and 417 lead to it being alive. We thus make the second cell alive, so as to have fewer parents.

[30] In fact, not only is it the case that $(2^{36}-1)^{n^2} < 2^{(6n-2)^2}$ when n is large, but the ratio $(2^{36}-1)^{n^2}/2^{(6n-2)^2}$ can be made as small as we like by making n sufficiently large (see Exercise 1.18). In other words, not only do Gardens of Eden *exist*, but in fact almost all large patterns are Gardens of Edens.

[31] This more general theorem was proved by Edward F. Moore and John Myhill in the early 1960s [Moo62, Myh63], so Gardens of Eden were known to exist in Conway's Game of Life right from the moment it was introduced.

[32] The first explicit Garden of Eden in the Game of Life was constructed by Roger Banks in 1971.

[33] This method is due to Nicolay Beluchenko [Slo11].

- We continue in this way in a clockwise spiral around the initial cell, deciding whether each new cell will be alive or dead based on which of the two possibilities results in a pattern with fewer parents.

While this method might seem not to yield anything useful at first—after 40 cells, each of them has been chosen to be alive, and the number of parents has exploded to 4,624,592—the number of parents does eventually start to dwindle, and after 266 cells the pattern has no parents at all (and is thus a Garden of Eden).[34] This Garden of Eden is displayed in Figure 1.35.

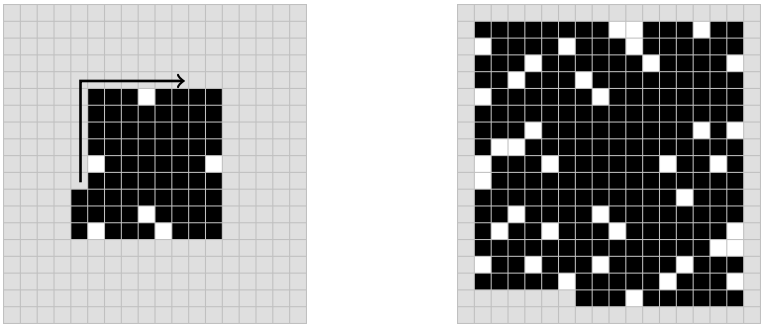
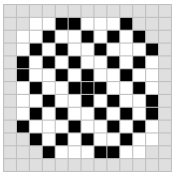


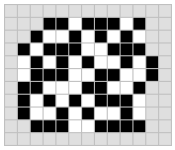
Figure 1.35: A partial Garden of Eden (left) that is being constructed one cell at a time by placing cells in a clockwise spiral, choosing whether they should be alive or dead based on which state results in fewer parents. The completed Garden of Eden (right) has 41 dead cells and 225 alive cells (the light gray cells are unspecified—the pattern remains a Garden of Eden regardless of whether they are alive or dead).

The pattern that we have constructed is interesting in that it is not only a Garden of Eden, but it remains a Garden of Eden regardless of what live and dead cells we place outside of the central 266-cell spiral. Patterns like this one that cannot be *any* part of the evolution of another pattern (equivalently, they are still a Garden of Eden no matter what other cells are placed around them) are called *orphans*.[35] Every pattern containing an orphan is (by definition) a Garden of Eden, but it is currently unknown whether or not there exists a Garden of Eden that does not contain an orphan.

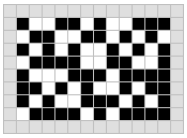
Now that we know that small Gardens of Eden (and orphans) exist, we would like to know how small and simple they can be. A complete answer to this problem is still open (and perhaps is currently one of the most actively-researched problems in Life), but many partial results are known. The smallest known orphans and Gardens of Eden (in terms of number of cells and bounding box size) are presented in Figure 1.36.[36]



(a) A Garden of Eden with 45 live cells, found by Nicolay Beluchenko in 2009.



(b) An orphan with 88 specified cells, found by Steven Eker in 2017.



(c) A Garden of Eden in a bounding box with area $96 = 8 \times 12$, found by Steven Eker in 2016.

Figure 1.36: The current smallest-known Gardens of Eden by (a) number of live cells, (b) total number of live and dead cells specified in the orphan, and (c) bounding box area.

On the other hand, there are also results that show that orphans cannot be "too" small. For example,

[34] In step 262 there is a tie between the number of parents if the cell is chosen to be alive or dead. We (arbitrarily) chose the cell to be alive.

[35] This term was coined by Conway. Also, the proof of Theorem 1.1 actually demonstrates the existence of orphans, not just Gardens of Eden.

[36] These patterns can be verified to be orphans via a computer program called *JavaLifeSearch*—see `conwaylife.com/wiki/JavaLifeSearch` and Exercise 1.19.

exhaustive computer searches have shown that there does not exist an orphan that fits within a 6×7 bounding box.[37] In a similar vein, we now show that there does not exist a Garden of Eden that has a bounding box with height one.[38]

Theorem 1.2 In Conway's Game of Life, every pattern that fits within a bounding box of height 1 has a parent. In other words, there do not exist Gardens of Eden with height 1.

Proof. We prove the theorem by explicit construction: we show how to find a parent of any 1-cell-thick pattern. The key step in our construction is to build a border that dies off completely in one generation and guarantees that any junk placed inside of it does not expand outward in that generation. Figure 1.37 shows one such border: no matter what we place inside the $3 \times n$ box in the middle of the border, no cell outside of that $3 \times n$ box will be alive in the next generation.

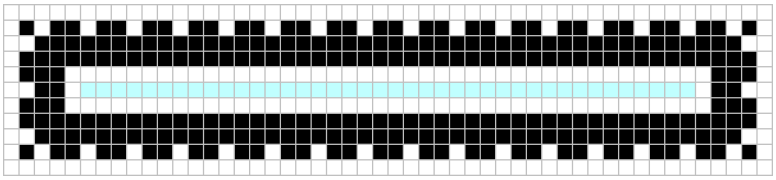


Figure 1.37: A border that can be used to help construct parents of 1-cell-thick patterns. No matter what we place in the central $3 \times n$ region, no cell outside of the $3 \times n$ region will be alive in the next generation. We will fill it in such a way so that the desired $1 \times (n-2)$ pattern evolves in the center row, highlighted in aqua).

To complete the proof, we just need to find a way to fill that $3 \times n$ box in such a way that it evolves into any $1 \times (n-2)$ pattern that we desire. One simple method that *almost* works is to put the desired pattern itself in the central row of the $3 \times n$ box, and its negation (i.e., flip all alive cells to dead and vice-versa) in the top and bottom rows of the $3 \times n$ box. It is straightforward to check that the cells in the top and bottom rows of the $3 \times n$ box will always be overpopulated, and thus those two rows will always be dead in the next generation. Furthermore, the central row evolves in the correct way in 7 out of every 8 possible configurations of three adjacent alive/dead cells (see Figure 1.38).

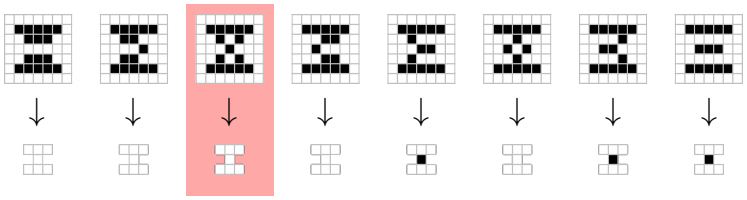


Figure 1.38: All except for one of the 8 different 3×3 squares in which the top and bottom rows are the negation of the middle row evolve so that the top and bottom rows die and the middle cell stays the same.

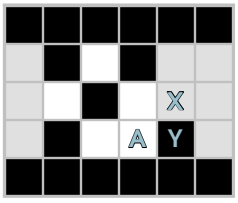


Figure 1.39: In order to correct the problematic 3×3 block from Figure 1.38, we set cell A to be dead and cell Y to be alive, regardless of the state of cell X.

To fix the problem that occurs when a single alive cell is between two dead cells in the middle row, simply set the bottom-right cell (i.e., cell A in Figure 1.39) of that 3×3 block as dead, rather than alive. To compensate for this cell being dead, we also set the cell to its immediate right (i.e., cell Y in that figure) to be alive, regardless of whether cell X is alive or dead. It is then straightforward to verify that the resulting pattern will evolve into the $1 \times (n-2)$ pattern that we desire: all that needs to be checked is that the central cell in this 3×3 block now stays alive (since it now has exactly 3 live neighbors) and that moving the alive cell from A to Y does not affect the evolution of any of the other 3×3 blocks to its right (which can be done via a simple case analysis). ■

[37] This computation was carried out independently by Steven Eker and Marijn Heule in 2016.

[38] This theorem was originally proved by Jean Hardouin–Duparc in 1974 [HD74].

The method used in the proof of Theorem 1.2 is illustrated in Figure 1.40, where we start with a (randomly-chosen) pattern with height 1 and explicitly construct a parent of it.

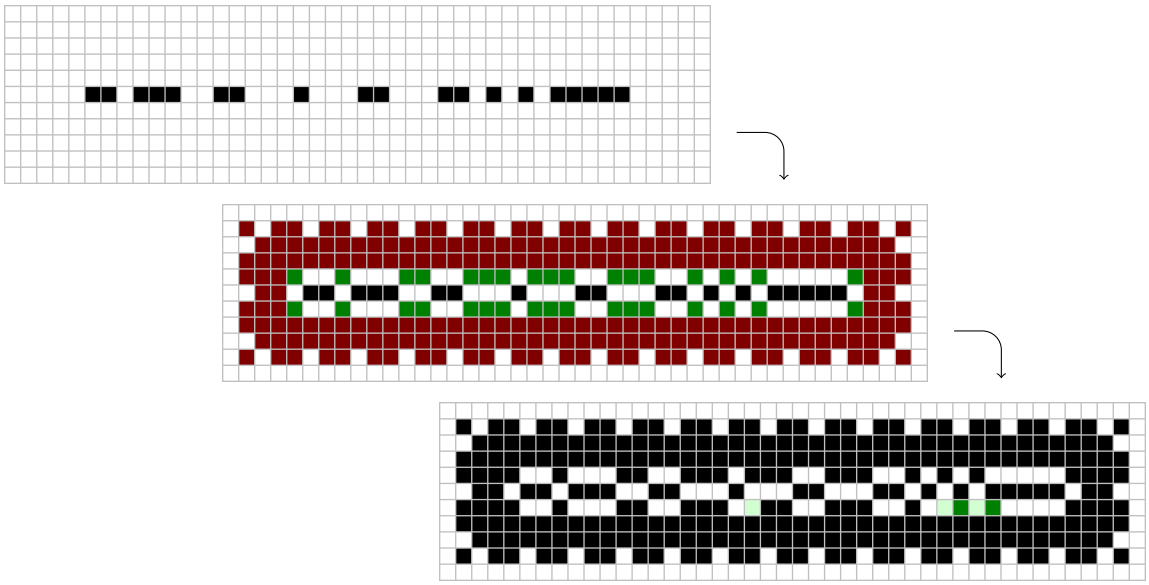


Figure 1.40: How to use the proof of Theorem 1.2 to construct a parent (at the bottom) of a pattern with height 1 (at the top). First, we place the border around the pattern (displayed in dark red in the middle image) and place the pattern's negation in the top and bottom rows of the box (displayed in dark green in the middle image). We then adjust the bottom row slightly (highlighted in light green in the bottom image) whenever we see the troublesome configuration from Figure 1.39.

More sophisticated techniques have been used to prove that there similarly do not exist orphans whose bounding boxes are 2 or 3 cells high.[39] On the other hand, there does exist an orphan (and thus a Garden of Eden) with height 5, as shown in Figure 1.41.[40] The question of whether or not there exists an orphan whose bounding box is 4 cells high remains open.

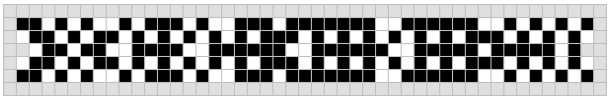


Figure 1.41: A Garden of Eden and orphan that fits within a 5×45 bounding box. No Garden of Eden with height less than 5 is currently known.

1.7.1 A Pattern with no Grandparent

There are many very difficult follow-up questions that can be asked about Gardens of Eden or patterns with properties related to those of Gardens of Eden. For example, does there exist a pattern that has a parent but no grandparent? In other words, does there exist a pattern with the property that all of its parents are Gardens of Eden? Although there is no known non-constructive argument like the one used in the proof of Theorem 1.1 that shows the existence of such patterns, it turns out that the answer to this question is "yes", and an example of such a pattern is presented in Figure 1.42. This pattern has the remarkable property that it has 17,920 distinct parents, every single one of which is a Garden of Eden—one of these parents is displayed in Figure 1.43.[41]

[39] This was proved by Steven Eker in 2016, using the same computer-assisted methods introduced by Jean Hardouin–Duparc.

[40] This orphan was also found by Steven Eker in 2016.

[41] Whether or not such a pattern exists was originally asked by Conway in October 1972. It was not resolved until May 2016, when the pattern displayed in Figure 1.42 was found by user "mtve" on the ConwayLife.com forums. Later that month, they even found a pattern that has a grandparent but no great-grandparent and a pattern with a great-grandparent but no great-great-grandparent, though the question remains open of whether or not, for every integer $n \geq 1$, there exists a pattern with a great^n-grandparent but no great^{n+1}-grandparent.

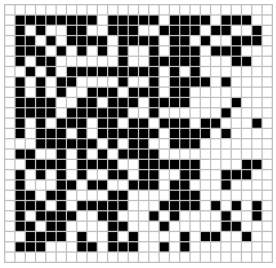


Figure 1.42: A pattern that has a parent but no grandparent.

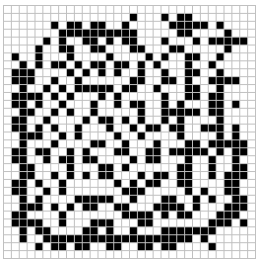


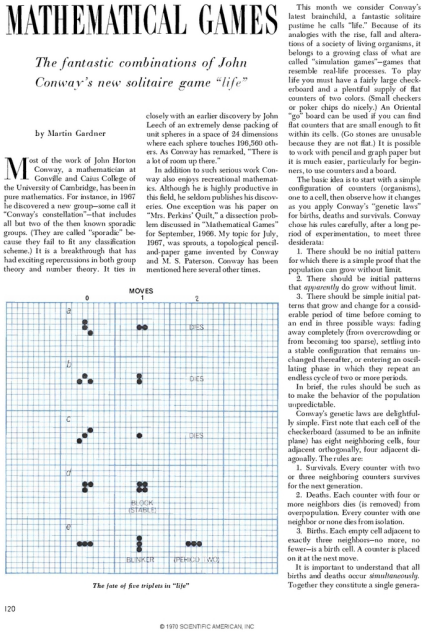
Figure 1.43: A parent of the grandparentless pattern from Figure 1.42.

The method used to construct this pattern is almost identical to the method we used to construct the Garden of Eden in Figure 1.35. That is, it was constructed one cell at a time, and each new cell that was added was chosen to be either alive or dead based on which of the two options resulted in the pattern having fewer grandparents.

Notes and Historical Remarks

Conway's Game of Life was initially studied by John Conway and some of his collaborators at Cambridge University, as well as a research group led by Bill Gosper at MIT, in 1969 and 1970. It then received mainstream attention in 1970 due to an article that Martin Gardner wrote about it in the magazine *Scientific American* [Gar70] (see Figure 1.44). Early discoveries by enthusiasts were sent to Gardner, who then shared those discoveries with Conway and, due to the overwhelming response to his article, wrote a follow-up article in February 1971 [Gar71].

It was unsustainable for Martin Gardner to continue to serve as curator for all Life discoveries, and Scientific American would only agree to so many articles on the subject, so in March 1971 a quarterly newsletter called *Lifeline* was announced, which was edited and distributed by Robert Wainwright (see Figure 1.45). This newsletter had 11 editions, and included the announcement of many of the objects that we saw in this chapter, including the twin bees gun, the switch engine (and its block-laying and glider-producing variants), and the acorn methuselah. Its final issue was dated September 1973.



MATHEMATICAL GAMES

The fantastic combinations of John Conway's new solitaire game "life"

by Martin Gardner

Most of the work of John Horton Conway, a mathematician at Gonville and Caius College of the University of Cambridge, has been in pure mathematics. For instance, in 1967 he discovered a new group—some call it "Conway's constellation"—that includes all but two of the then known sporadic groups. (They are called "sporadic" because they fail to fit any classification scheme.) It is a breakthrough that has had exciting repercussions in both group theory and number theory. It ties in closely with an earlier discovery by John Leech of an extremely dense packing of unit spheres in a space of 24 dimensions where each sphere touches 196,560 others. As Conway has remarked, "There is a lot of room up there."

In addition to such serious work Conway also enjoys recreational mathematics. Although he is highly productive in this field, he seldom publishes his discoveries. One exception was his paper on "Mrs. Perkins' Quilt," a dissection problem discussed in "Mathematical Games" for September, 1966. My topic for July, 1967, was sporets, a topological pencil-and-paper game invented by Conway and M. S. Paterson. Conway has been mentioned here several other times.

This month we consider Conway's latest brainchild, a fantastic solitaire pastime he calls "life." Because of its analogies with the rise, fall and alterations of a society of living organisms, it belongs to a growing class of what are called "simulation games"—games that resemble real-life processes. To play life you must have a fairly large checkerboard and a plentiful supply of flat counters of two colors. (Small checkers or poker chips do nicely.) An Oriental "go" board can be used if you can find flat counters that are small enough to fit within its cells. (Go stones are unusable because they are not flat.) It is possible to work with pencil and graph paper but it is much easier, particularly for beginners, to use counters and a board.

The basic idea is to start with a simple configuration of counters (organisms), one to a cell, then observe how it changes as you apply Conway's "genetic laws" for births, deaths and survivals. Conway chose his rules carefully, after a long period of experimentation, to meet three desiderata:

1. There should be no initial pattern for which there is a simple proof that the population can grow without limit.
2. There should be initial patterns that apparently do grow without limit.
3. There should be simple initial patterns that grow and change for a considerable period of time before coming to an end in three possible ways: fading away completely (from overcrowding or from becoming too sparse), settling into a stable configuration that remains unchanged thereafter, or entering an oscillating phase in which they repeat an endless cycle of two or more periods.

In brief, the rules should be such as to make the behavior of the population unpredictable.

Conway's genetic laws are delightfully simple. First note that each cell of the checkerboard (assumed to be an infinite plane) has eight neighboring cells, four adjacent orthogonally, four adjacent diagonally. The rules are:

1. Survivals. Every counter with two or three neighboring counters survives for the next generation.
2. Deaths. Each counter with four or more neighbors dies (is removed) from overpopulation. Every counter with one neighbor or none dies from isolation.
3. Births. Each empty cell adjacent to exactly three neighbors—no more, no fewer—is a birth cell. A counter is placed on it at the next move.

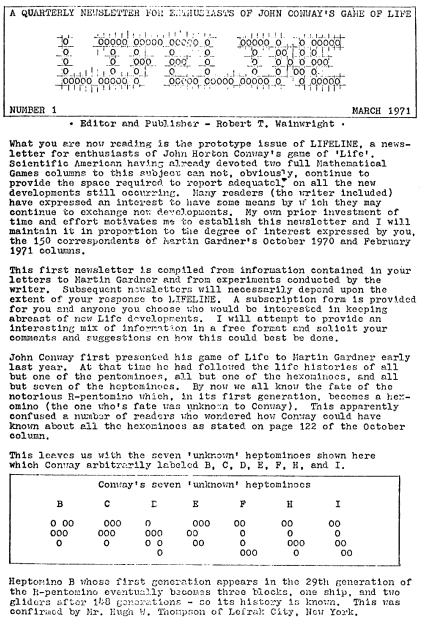
It is important to understand that all births and deaths occur *simultaneously*. Together they constitute a single genera-

The fate of five triplets in "life"

120

© 1970 SCIENTIFIC AMERICAN, INC

Figure 1.44: Conway's Game of Life was popularized by the October 1970 issue of *Scientific American*.



A QUARTERLY NEWSLETTER FOR ENTHUSIASTS OF JOHN CONWAY'S GAME OF LIFE

NUMBER 1 MARCH 1971

• Editor and Publisher - Robert T. Wainwright •

What you are now reading is the prototype issue of LIFELINE, a newsletter for enthusiasts of John Horton Conway's game of 'Life'. Scientific American having already devoted two full Mathematical Games columns to this subject can not, obviously, continue to provide the space required to report adequately on all the new developments still occurring. Many readers (the writer included) have expressed an interest to have some means by which they may continue to exchange new developments. My own prior investment of time and effort motivates me to establish this newsletter and I will maintain it in proportion to the degree of interest expressed by you, the 150 correspondents of Martin Gardner's October 1970 and February 1971 columns.

This first newsletter is compiled from information contained in your letters to Martin Gardner and from experiments conducted by the writer. Subsequent newsletters will necessarily depend upon the extent of your response to LIFELINE. A subscription form is provided for you and anyone you choose who would be interested in keeping abreast of new Life developments. I will attempt to provide an interesting mix of information in a free format and solicit your comments and suggestions on how this could best be done.

John Conway first presented his game of Life to Martin Gardner early last year. At that time he had followed the life histories of all but one of the pentominoes, all but one of the hexominoes, and all but seven of the heptominoes. By now we all know the fate of the notorious R-pentomino which, in its first generation, becomes a hexomino (the one who's fate was unknown to Conway). This apparently confused a number of readers who wondered how Conway could have known about all the hexominoes as stated on page 122 of the October column.

This leaves us with the seven 'unknown' heptominoes shown here which Conway arbitrarily labeled B, C, D, E, F, H, and I.

Conway's seven 'unknown' heptominoes

B C D E F H I

Heptomino B whose first generation appears in the 29th generation of the R-pentomino eventually becomes three blocks, one ship, and two gliders after 148 generations - so its history is known. This was confirmed by Mr. Hugh M. Thompson of Lefrak City, New York.

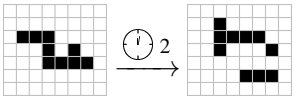
Figure 1.45: The front page of the first issue of *Lifeline*, a newsletter for Life enthusiasts that ran from March 1971 to September 1973.

From 1974 to 2009, Life enthusiasts mostly shared patterns via various private mailing lists, and collections of those patterns made their way to various personal web pages in the 1990s. In 2004, the LifeNews website was launched at pentadecathlon.com, which posted most interesting new Life discoveries as they were made. Since about 2009, most new discoveries in Life are now reported on the forums at conwaylife.com.

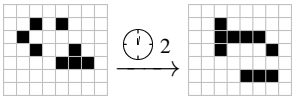
Although more advanced techniques are now known for constructing interesting Life objects (indeed, these techniques comprise the majority of the rest of this book), the idea of simply running randomly-generated soups to completion to see what remains of them after they stabilize has been a consistent one. However, because computing power was at much more of a premium in 1970 than it is now, and there was no pre-made Life simulation software for early enthusiasts to use, these patterns were often evolved by hand using graph paper, checkers, or the board and stones from the game Go, to represent the Life grid and its live cells.

Furthermore, the methodology that was used back then was somewhat more systematic—instead of generating large random starting configurations, early Lifers investigated the evolution of all possible small starting configurations. These investigations were exactly where the original interest in the T-tetromino, R-pentomino, B-heptomino, stairstep hexomino, and other small polyominoes came from. In fact, the switch engine was originally found by Charles Corderman when he was investigating the evolution of nonominoes (i.e., polyominoes with 9 live cells), one of which evolves in the exact same way as the switch engine (see Figure 1.46).

Solomon W. Golomb invented the term "polyomino" in 1953, and Martin Gardner wrote a popular Mathematical Games article about them a decade before the first article on Conway's Game of Life. Perhaps partly because of this long-standing interest in polyominoes in the mathematical community, the fates of N-cell polyominoes were exhaustively researched very early on, giving rise to the common names of many of the active patterns discussed above. The Moore-neighborhood equivalent of polyominoes, the N-cell *polyplets*—cell groups that are kingwise-connected rather than rookwise-connected— are technically just as relevant as a potential source of novelty, but their evolutionary fates were much less thoroughly explored in the early years than the polyominoes.



(a) A nonomino.



(b) A switch engine.

Figure 1.46: The nonomino (a) evolves in the same way as the switch engine (b) after 2 generations.

Building upon this idea of finding objects by evolving random starting configurations, Achim Flammenkamp ran an automated computer search in 1994 that repeatedly generated random soups and evolved them, keeping a list of all objects that it found in the resulting ash. This search ran until it had accumulated 5×10^9 (non-distinct) ash objects, which contained a total of 48 distinct oscillators [Fla94]. He then performed this search again in 2004, this time accumulating 5×10^{10} ash objects, and found over 3,500 distinct still lifes and over 80 different oscillators [Fla04].

Andrzej Okrasinski created a similar program that acted as a screensaver in November 2003. Over the subsequent 5 years, this screensaver catalogued over 4.7×10^{11} ash objects, including over 8,000 distinct still lifes and about 180 oscillators, but still the only spaceships that turned up were the four that we have already seen (the glider, LWSS, MWSS, and HWSS) [Okr03]. The screensaver also kept track of how long each starting configuration took to stabilize, and found some new methuselahs in the process, including a 15-cell version of Lidka, which was presented in Table 1.1 (it was improved to a 13-cell version by David Bell and then to the more compact 13-cell version presented in that table by Simon Ekström).

In 2009, Nathaniel Johnston created a distributed online search to continue in this vein, called *The Online Life-Like CA Soup Search* (or *TOLLCASS* for short). This script had the advantage that multiple people could run it simultaneously on different computers, and their results were automatically uploaded to a central server that organized them. TOLLCASS also worked not just with Conway's

Game of Life, but also with a handful of other Life-like cellular automata. The major downside of this script was that it was quite slow, and over the two years that it was active, it catalogued only about one third as many objects as Okrasinski's earlier search.

Finally, Adam P. Goucher launched another distributed online search in 2014 called *apgsearch*,[42] which is still active today[43] and is the current state-of-the-art when it comes to soup searching. Like TOLLCASS, it can be used with several different Life-like CA, but with the main advantage of being extremely fast. In the six years since it began, it has catalogued more than 2,500 times as many ash objects as all of the previous searches combined (and this number will likely become out of date very quickly). A summary of these various soup searches that have taken place over the years is provided by Table 1.2.

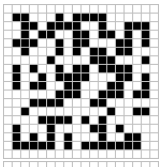
Search Name	Year(s)	Ash Objects Found	Notes
Flammenkamp	1994	5.0×10^9	first automated soup search
Flammenkamp	2004	5.0×10^{10}	–
Okrasinski	2003–2008	4.7×10^{11}	also found methuselahs
TOLLCASS	2009–2011	1.7×10^{11}	online search, multiple CA
apgsearch	2014–present	4.9×10^{14}	online search, multiple CA

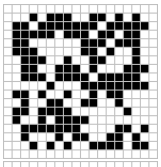
Table 1.2: A summary of the different soup searches that have taken place over the years.

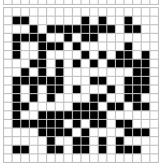
Exercises

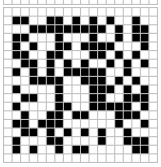
solutions to starred exercises on page ??

∗ **1.1** Evolve each of the following randomly-generated 20×20 soups and describe the most unusual object that forms.

(a) 

(b) 

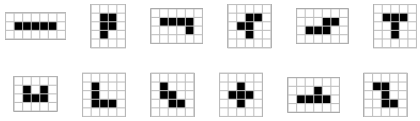
(c) 

(d) 

1.2 What is the longest lifespan of a pattern with:

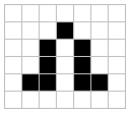
(a) 1 or 2 live cells?
(b) 3 live cells?
(c) 4 live cells? [Hint: You could try writing a computer program to help you.]

1.3 All 12 different pentominoes are displayed below. What are their lifespans?



1.4 In this exercise, we will find some alternate stabilizations of the twin bees shuttle from Figure 1.22.

(a) Try removing some of the blocks from the twin bees shuttle. Which combinations of blocks can you remove while preserving it as a period 46 oscillator?
(b) The still life below is called a *hat*. Use a hat to stabilize one side of the twin bees shuttle.



∗ **1.5** Find a specific generation at which each of the following unstable objects appears in the evolution of the R-pentomino.

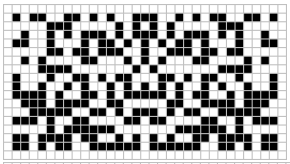
(a) A T-tetromino.
(b) A pre-honeyfarm.
(c) A pi-heptomino.
(d) A queen bee.
(e) Lumps of muck (in particular, generation 3 of the stairstep hexomino).

1.6 Use two blocks and three queen bees to create a "double" Gosper glider gun: a gun that emits two streams of gliders.

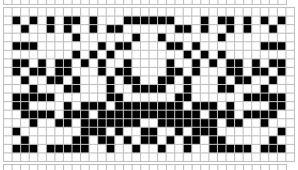
[42] Officially, the "apg" in apgsearch are not its author's initials, but rather stand for **a**sh **p**attern **g**enerator.
[43] It is available at `catagolue.appspot.com`.

∗**1.7** Because so many interesting Life objects display some form of symmetry, it is often fruitful to investigate random starting configurations that are also symmetric. Evolve each of the following randomly-generated symmetric soups and describe the most unusual object that forms.

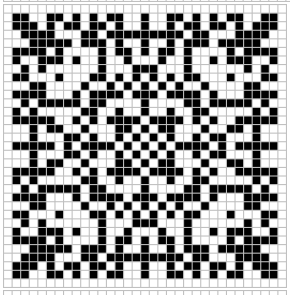
(a)



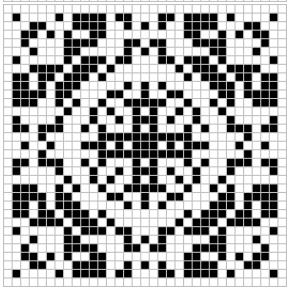
(b)



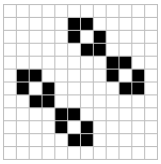
(c)

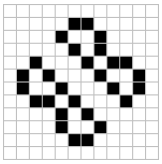


(d)



∗**1.8** There are 5 formations of 4 simple objects that commonly occur in ash: the traffic light, honey farm, and blockade that we saw in Section 1.2, and the *fleet* and *bakery* displayed below on the left and right, respectively. These arrangements are collectively called the *familiar fours*.

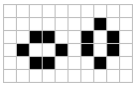




(a) Evolve Lidka until you see a fleet, and use this evolution to find a 7-cell predecessor of the fleet.

(b) There are at least two different 8-cell objects that evolve into a bakery. Find one of them. [Hint: Try evolving random soups until you find a bakery, or write a computer program that evolves many different 8-cell objects.]

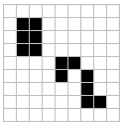
∗**1.9** This problem concerns the configuration of two beehives displayed below.



(a) Evolve the pattern until it stabilizes. What happens to the beehives?

(b) Create a period 30 oscillator that uses this reaction to stabilize two queen bees that are rotated 90 degrees from each other.

∗**1.10** This problem concerns the pattern displayed below. The top-left object is a pre-beehive, while the bottom-right object is called *eater 1*.



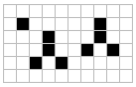
(a) Evolve the pattern until it stabilizes. What happens to the pre-beehive and eater 1?

(b) Evolve the pre-beehive and eater 1 individually (i.e., not next to each other). Describe what happens to them.

(c) Evolve a queen bee until you find a pre-beehive that it leaves behind.

(d) Use an eater 1 (instead of a block) to stabilize one side of the queen bee shuttle.

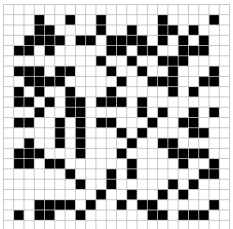
1.11 The 9-cell methuselah displayed below is called *bunnies*.[44]



(a) What is its lifespan?

(b) It eventually evolves in the same way as each of bunnies 9 and bunnies 10b from Table 1.1. Find the first generation of each object's evolution where they match.

1.12 The methuselah displayed below is named *Edna*.[45]



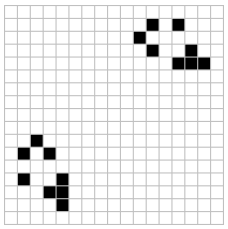
(a) What is its lifespan?

(b) What objects can be found in its ash after this pattern stabilizes? Are there any objects that we did not give names to in this chapter?

[44] Its 1-generation successor, which is called *rabbits*, was found by Andrew Trevorrow in 1986. This version was subsequently found independently by Robert Wainwright and Andrew Trevorrow.

[45] Found by Erik de Neve in January 2010.

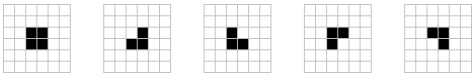
1.13 An ark called *Noah's ark* is displayed below.[46] What is the period of this pattern, and what objects does it leave behind as it moves?



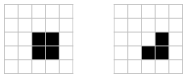
∗**1.14** In the proof of Theorem 1.1, we claimed that the inequality $(2^{36}-1)^{n^2} < 2^{(6n-2)^2}$ holds whenever n is large enough.

(a) Find a formula for the smallest value of n for which this inequality holds, and then use computer software to compute its exact value.

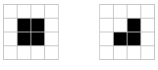
(b) We can get slightly smaller values of n that lead to Gardens of Eden by considering more tiles that evolve in the same way. Instead of just using the two 6×6 tiles that we presented in the proof of Theorem 1.1, we could have used the set of *five* tiles displayed below without changing the method of proof significantly. What is the relevant inequality that we need to check? What is the smallest value of n for which this new inequality holds?



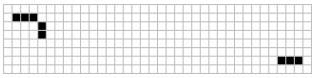
(c) Alternatively, we can get smaller values of n leading to Gardens of Eden by using 5×5 tiles. If we re-do the proof of Theorem 1.1 using the 5×5 tiles displayed below, what is the relevant inequality that we need to check? What is the smallest value of n for which this new inequality holds?



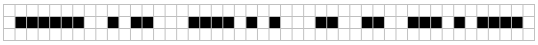
(d) Why can't we replace the 6×6 or 5×5 tiles by the 4×4 tiles displayed below? What part of the proof breaks down?



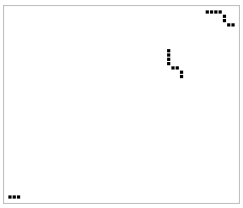
1.15 An 8-cell methuselah with a longer lifespan than the 8-cell methuselah from Table 1.1 (but with a significantly larger bounding box) is displayed below.[47] What is its lifespan?



1.16 Find a parent of the following 1-cell-thick pattern:



∗**1.17** The 19-cell pattern displayed below consists of two switch engine predecessors and a blinker,[48] and takes a staggering 6,526,589 generations to stabilize.



(a) Evolve the pattern to see how it behaves. What effect does the blinker have that makes it live so much longer than the ark in Figure 1.30?

(b) What causes this ark to stabilize? [Hint: Compare the evolution of this ark with the evolution presented in Figure 1.31, which shows the stabilization happening as a result of a path being cleared for the backward-flying gliders.]

1.18 In the proof of Theorem 1.1, we claimed that the inequality $(2^{36}-1)^{n^2} < 2^{(6n-2)^2}$ holds whenever n is large enough. Prove the stronger statement that

$$\lim_{n\to\infty} \frac{(2^{36}-1)^{n^2}}{2^{(6n-2)^2}} = 0,$$

and hence the vast majority of large patterns are Gardens of Eden. [Hint: Use the fact that $a^b = 2^{b\log_2(a)}$.]

1.19 A computer program called *JavaLifeSearch*[49] can be used to find predecessors of Life patterns (or show that none exist). Usage instructions and download links can be found at `conwaylife.com/wiki/JavaLifeSearch`.

(a) Use JavaLifeSearch to find a predecessor of the following pattern:

(b) Use JavaLifeSearch to verify that the patterns from Figure 1.36 really are orphans.

(c) Use JavaLifeSearch to verify that the pattern from Figure 1.42 really does not have any grandparents.

[46] This ark was found in 1971 by Charles Corderman, and was the first-discovered ark. Its name refers to the fact that the debris it leaves behind contains pairs of many different objects. The general term "ark" is derived from the name of this pattern.

[47] Found by Nick Gotts in 2019.

[48] This pattern was found by Nick Gotts in February 2005.

[49] Created by Karel Suhajda, based on earlier programs by David Bell, Dean Hickerson, and Jason Summers.

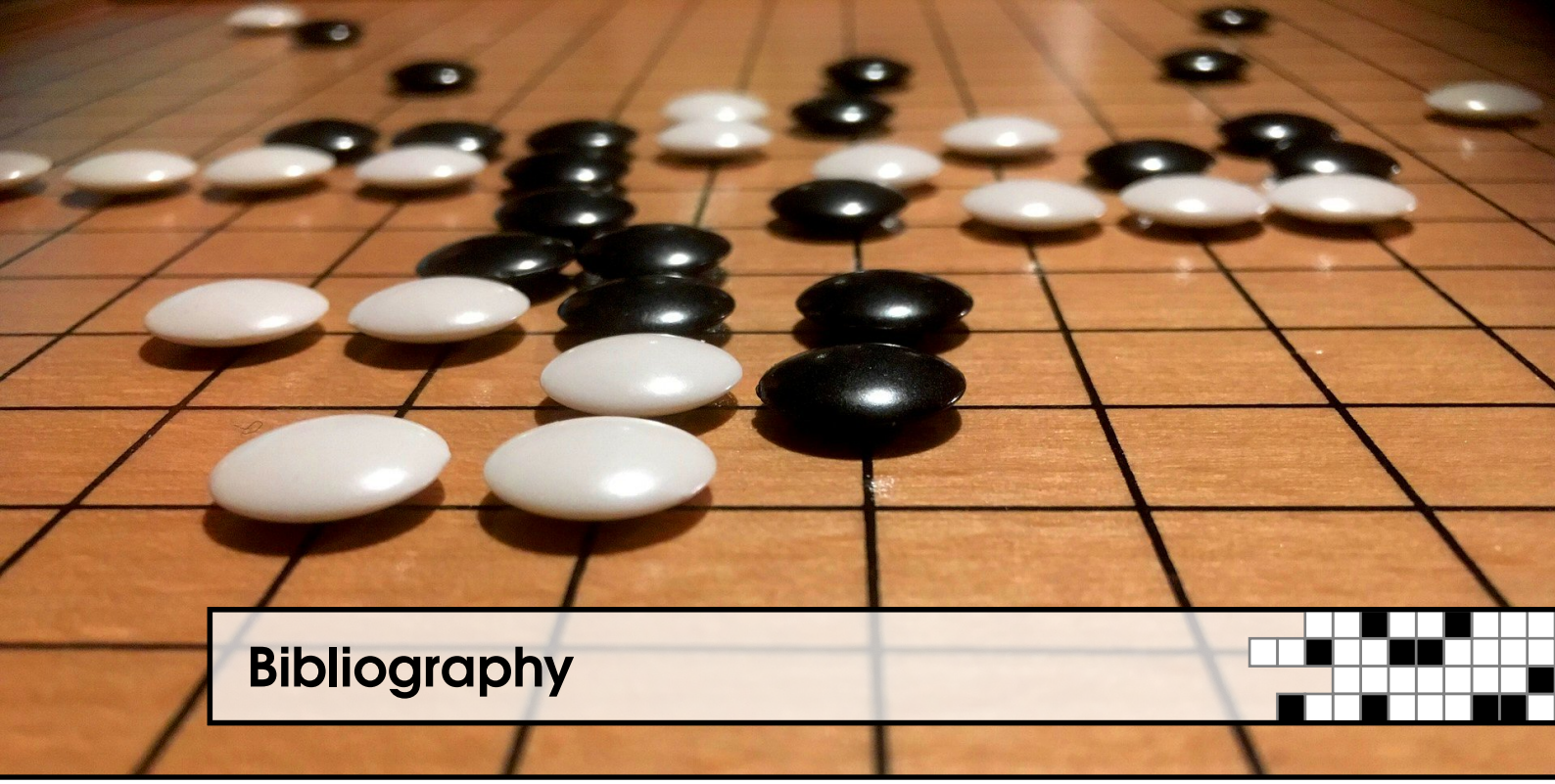
II

Circuitry and Logic

III

Constructions

Appendices and Supplements



Bibliography

[BC98] David J. Buckingham and Paul B. Callahan. Tight bounds on periodic cell configurations in Life. *Experimental Mathematics*, 7(3):221–241, 1998.

[BCG82] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays, volume 2: Games in Particular*, chapter 25. Academic Press, 1982.

[Bos99] Robert A. Bosch. Integer programming and Conway's game of Life. *SIAM Review*, 41(3):596–604, 1999.

[BT04] Robert Bosch and Michael Trick. Constraint programming and hybrid formulations for three life designs. *Annals of Operations Research*, 130:41–56, 2004.

[Coo03] Matthew Cook. *Still life theory*, pages 93–118. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 2003.

[CS12] Geoffrey Chu and Peter J. Stuckey. A complete solution to the Maximum Density Still Life Problem. *Artificial Intelligence*, 184:1–16, 2012.

[CSdlB09] Geoffrey Chu, Peter J. Stuckey, and Maria Garcia de la Banda. Using relaxations in maximum density still life. In *Proceedings of the Fifteenth International Conference on Principles and Practice of Constraint Programming*, pages 258–273, 2009.

[Elk98] Noam D. Elkies. The still-life density problem and its generalizations. In *Voronoï's Impact on Modern Science, Book I*, volume 21 of *Proceedings of the Institute of Mathematics of the National Academy of Sciences of Ukraine*, pages 228–253. Institute of Mathematics, 1998.

[Epp02] David Eppstein. Searching for spaceships. In *More Games of No Chance*, volume 42 of *MSRI Publications*, pages 433–453. Cambridge University Press, 2002.

[Fla94] Achim Flammenkamp. Natural grown oscillators out of random soup. http://wwwhomes.uni-bielefeld.de/achim/oscill.html, 1994. [Online; accessed May 6, 2016].

[Fla04] Achim Flammenkamp. Frequency of Game of Life objects in settled random areas. http://wwwhomes.uni-bielefeld.de/achim/freq_top_life.html, 2004. [Online; accessed May 6, 2016].

[Gar70] Martin Gardner. The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123, 1970.

[Gar71] Martin Gardner. On cellular automata, self-reproduction, the Garden of Eden and the game of "life". *Scientific American*, 224:112–117, 1971.

[Gib06] J. Gibbons. Unbounded spigot algorithms for the digits of pi. *The American Mathematical Monthly*, 113(4):318–328, 2006.

[Gos84] R. Wm. Gosper. Exploiting regularities in large cellular spaces. *Physica*, 10D:75–80, 1984.

[HD74] Jean Hardouin-Duparc. Paradis terrestre dans l'automate cellulaire de Conway. *Revue française d'automatique, informatique, recherche opérationnelle*, 8:63–71, 1974.

[LMN05] Javier Larrosa, Enric Morancho, and David Niso. On the practical use of variable elimination in constraint optimization problems: 'still-life' as a case study. *Journal of Artificial Intelligence Research*, 23:421–440, 2005.

[Moo62] Edward F. Moore. Machine models of self-reproduction. *Proceedings of Symposia in Applied Mathematics*, 14:17–33, 1962.

[Myh63] John Myhill. The converse of Moore's Garden-of-Eden theorem. *Proceedings of the American Mathematical Society*, 14:685–686, 1963.

[Nie03] Mark D. Niemiec. Synthesis of complex Life objects from gliders. In *New Constructions in Cellular Automata*, pages 55–78. Oxford University Press, London, 2003.

[Nie10] Mark D. Niemiec. Object synthesis in Conway's Game of Life and other cellular automata. In *Game of Life Cellular Automata*, chapter 8, pages 115–134. Springer London, 2010.

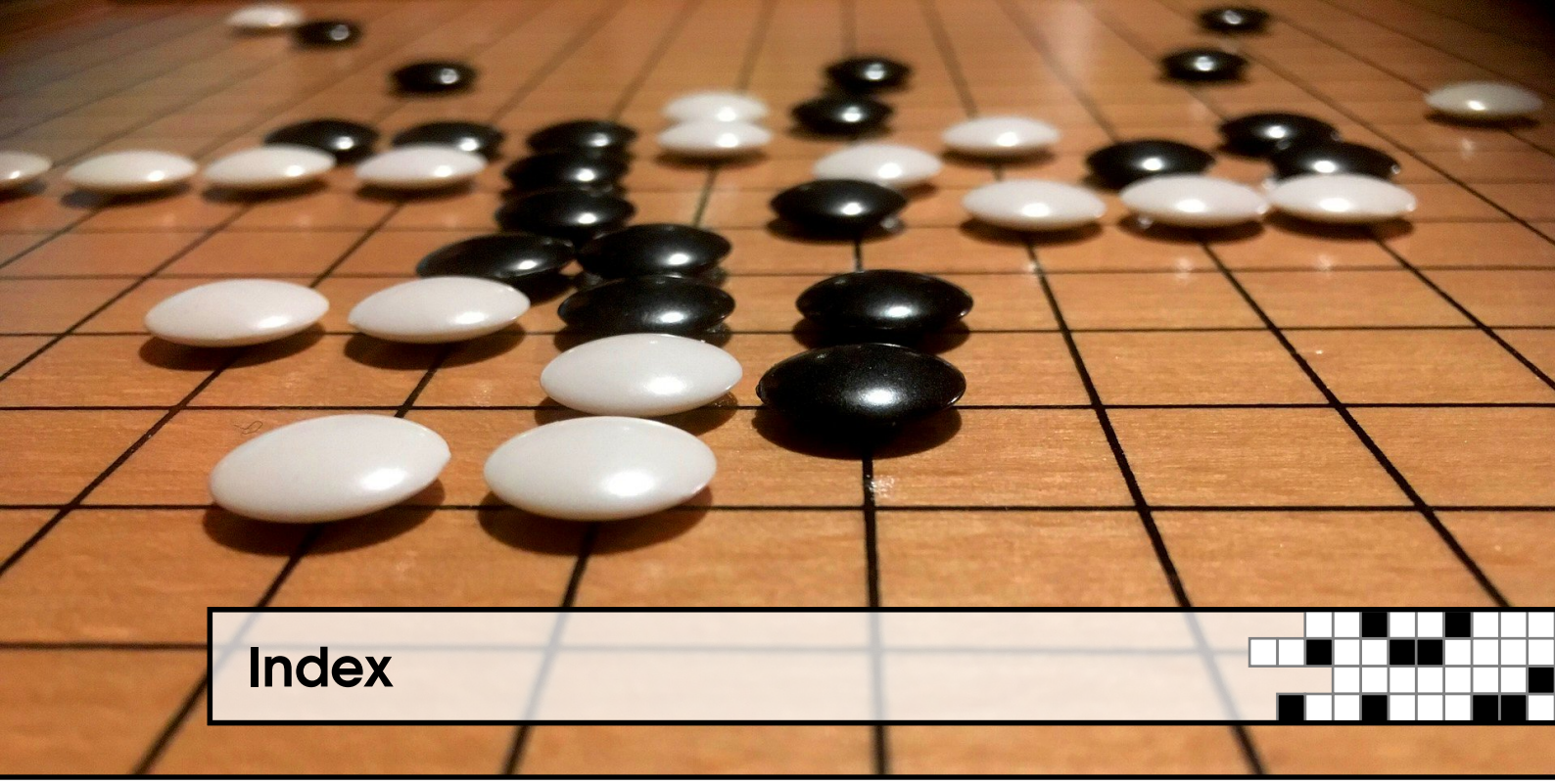
[Okr03] Andrzej Okrasinski. Andrzej Okrasinski's website dedicated to the Game of Life. geocities.ws/conwaylife/, 2003. [Online; accessed May 6, 2016].

[Slo96] N. J. A. Sloane. Sequence A019473 in *The On-Line Encyclopedia of Integer Sequences*. oeis.org/A019473, 1996. Number of stable n-celled patterns ("still lifes") in Conway's Game of Life, up to rotation and reflection. [Online; accessed Jan. 30, 2020].

[Slo99] N. J. A. Sloane. Sequence A046932 in *The On-Line Encyclopedia of Integer Sequences*. oeis.org/A046932, 1999. $a(n)$ = period of $x^n + x + 1$ over $GF(2)$, i.e., the smallest integer $m > 0$ such that $x^n + x + 1$ divides $x^m + 1$ over $GF(2)$. [Online; accessed July 28, 2018].

[Slo00] N. J. A. Sloane. Sequence A056613 in *The On-Line Encyclopedia of Integer Sequences*. oeis.org/A056613, 2000. Number of n-celled pseudo still lifes in Conway's Game of Life, up to rotation and reflection. [Online; accessed Jan. 30, 2020].

[Slo11] N. J. A. Sloane. Sequence A196447 in *The On-Line Encyclopedia of Integer Sequences*. oeis.org/A196447, 2011. The number of parents of successive approximations used in a greedy approach to creating a Garden of Eden in Conway's Game of Life. [Online; accessed May 14, 2016].



Index

I

J

L

M

N

O

P

Q

R

S

T

V