Стивен Вольфрам

Новый вид науки

Глава 2
Решающий эксперимент

Русский перевод для печати

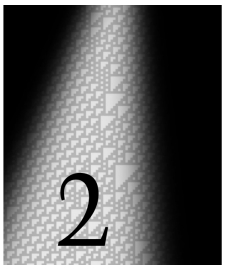
Глава: страницы 23–50 оригинала.
Примечания: страницы 865–882 оригинала.

Включены полный подготовленный перевод и все иллюстрации главы и примечаний. Английские страницы и английские подписи не включены. Программные идентификаторы и математическая запись сохранены. Иллюстрации воспроизведены из предоставленного PDF в оттенках серого.

Неофициальный перевод, подготовленный 18 ноября 2026 года. Исторические сведения и оценки переданы от лица автора. Номера исходных страниц указаны вверху каждого листа.

Оригинал: Stephen Wolfram, A New Kind of Science. Wolfram Media, 2002.
© Stephen Wolfram, LLC. Исходный файл предоставлен пользователем.

Глава 2. Решающий эксперимент
Как ведут себя простые программы?



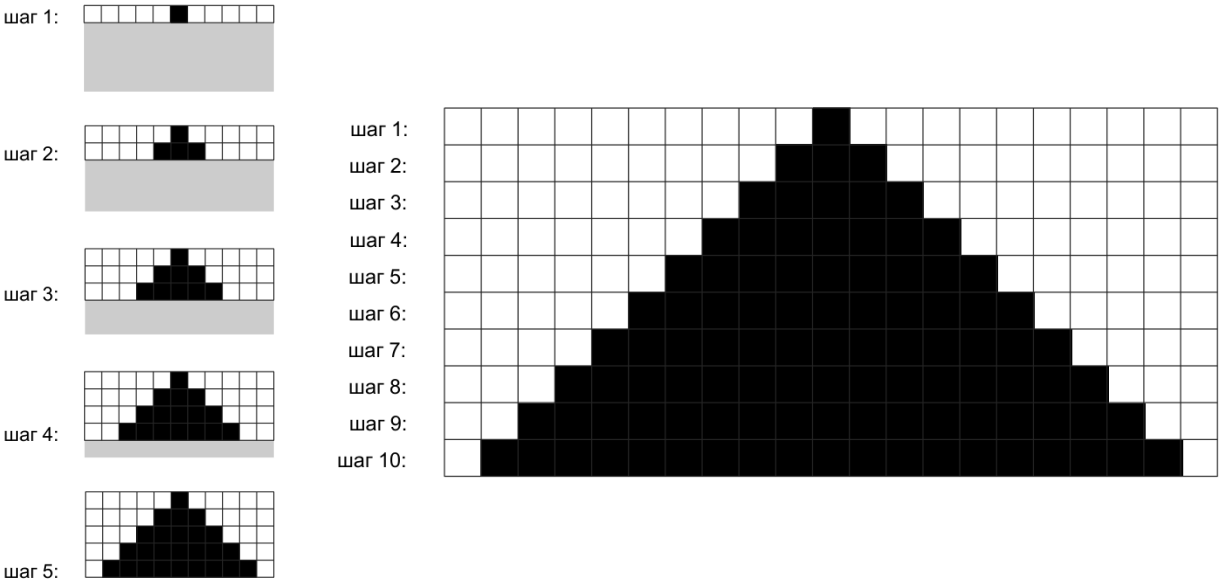
Новые направления науки обычно начинались с некоторых ключевых наблюдений или экспериментов. Для той науки, которую я описываю в этой книге, они были связаны с поведением простых программ.

В повседневной работе с компьютерами мы обычно сталкиваемся с программами, предназначенными для выполнения вполне определённых задач. Но ключевая идея, пришедшая мне почти двадцать лет назад и в конце концов приведшая ко всему тому новому виду науки, который представлен в этой книге, состояла в следующем: что произойдёт, если вместо этого просто рассматривать простые, произвольно выбранные программы, созданные без какой-либо конкретной задачи? Как обычно ведут себя такие программы?

Математические методы, которые прежде господствовали в теоретической науке, мало помогают ответить на этот вопрос. Но с помощью компьютера нетрудно начать экспериментальное исследование. Ведь для этого достаточно задать последовательность возможных простых программ, а затем запускать их и смотреть, как они себя ведут.

Любую программу на некотором уровне можно рассматривать как набор правил, указывающих, что ей следует делать на каждом шаге. Существует много способов задавать такие правила — и в этой книге мы действительно изучим немало из них. Но пока я рассмотрю особый класс примеров, называемых клеточными автоматами: именно они были первыми простыми программами, которые я исследовал в начале 1980-х годов.

Важная особенность клеточных автоматов состоит в том, что их поведение легко представить наглядно. На рисунке ниже показано, что делает один клеточный автомат на протяжении десяти шагов.



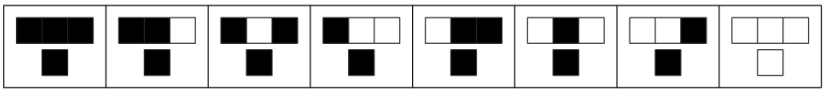
Наглядное представление поведения клеточного автомата: каждый ряд клеток соответствует одному шагу. На первом шаге клетка в центре чёрная, а все остальные клетки белые. Затем на каждом следующем шаге данная клетка становится чёрной, если на предыдущем шаге чёрной была она сама или любой из её соседей. Как видно на рисунке, это приводит к простому расширяющемуся узору, сплошь заполненному чёрным цветом.

Шаг 1; шаг 2; шаг 3; шаг 4; шаг 5; шаг 6; шаг 7; шаг 8; шаг 9; шаг 10.

Шаг 1; шаг 2; шаг 3; шаг 4; шаг 5.

Клеточный автомат состоит из ряда клеток, каждая из которых окрашена в чёрный или белый цвет. На каждом шаге определённое правило устанавливает цвет данной клетки по цветам этой клетки и её ближайших соседей слева и справа на предыдущем шаге.

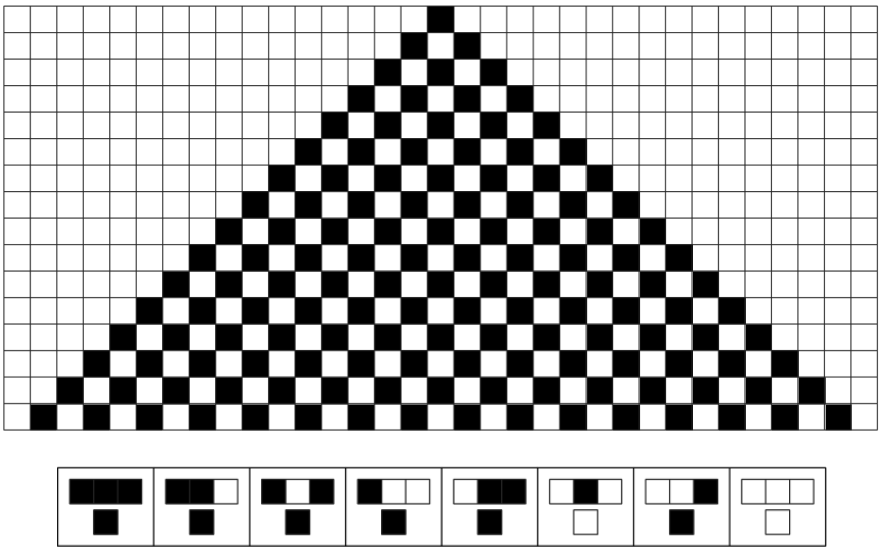
Для показанного здесь клеточного автомата правило, изображённое на рисунке ниже, предписывает окрашивать клетку в чёрный цвет во всех случаях, когда на предыдущем шаге чёрной была сама эта клетка или любой из её соседей.



Представление правила для клеточного автомата, показанного выше. Верхний ряд каждого прямоугольника задаёт одно из возможных сочетаний цветов клетки и её ближайших соседей. Нижний ряд указывает, какого цвета в каждом из этих случаев должна стать центральная клетка на следующем шаге. В системе нумерации, описанной в главе 3, это правило 254 клеточного автомата.

Рисунок вверху страницы показывает, что, если начать с единственной чёрной клетки в центре, это правило приводит к простому растущему узору, сплошь заполненному чёрным цветом. Однако даже слегка изменив правило, можно сразу получить другой узор.

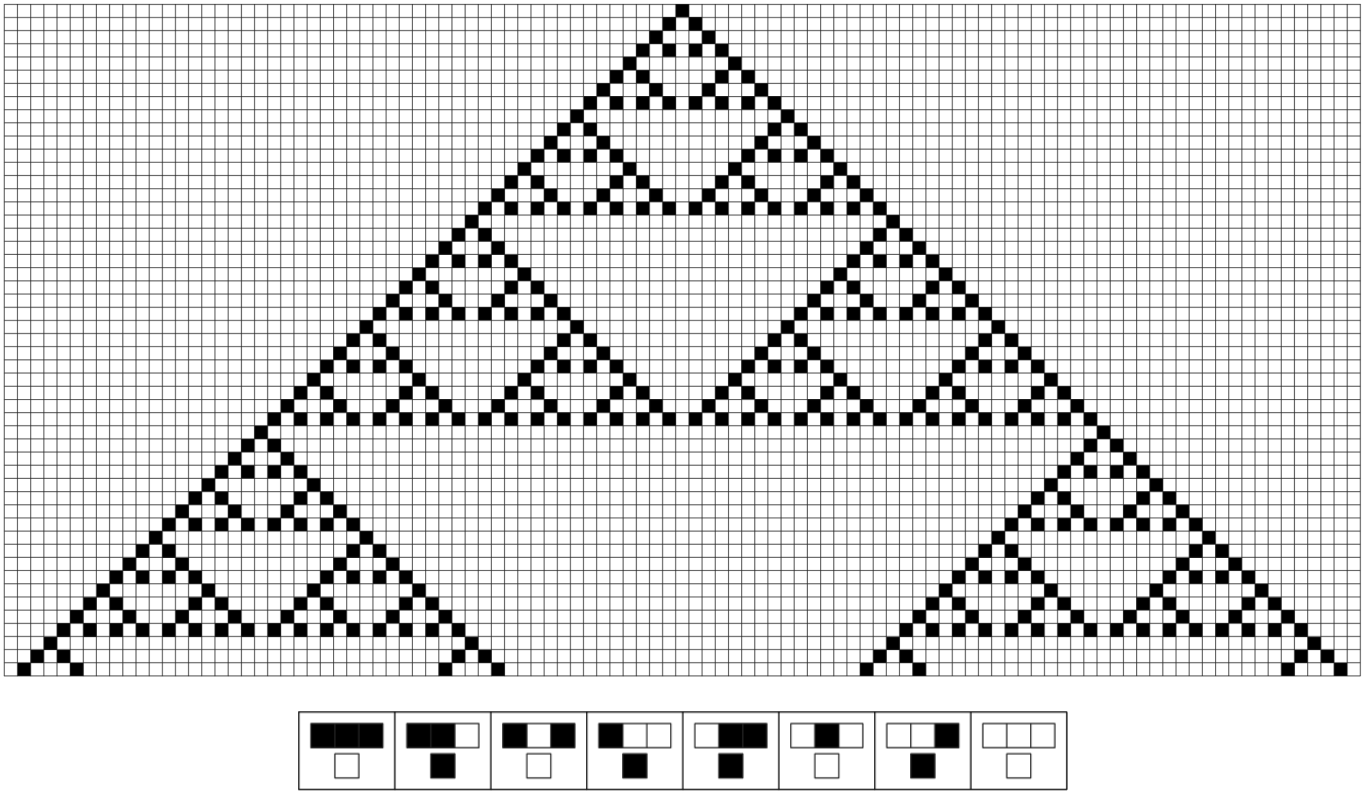
В качестве первого примера на рисунке вверху следующей страницы показано, что происходит с правилом, которое делает клетку белой всякий раз, когда оба её соседа на предыдущем шаге были белыми, — даже если сама клетка прежде была чёрной. Теперь это правило даёт не сплошь чёрный узор, а узор с повторяющимся чередованием чёрного и белого, как на шахматной доске.



Клеточный автомат с немного другим правилом. Правило делает данную клетку чёрной, если любой из её соседей был чёрным на предыдущем шаге, и делает клетку белой, если оба её соседа были белыми. Если начать с единственной чёрной клетки, это правило приводит к шахматному узору. В системе нумерации главы 3 это правило 250 клеточного автомата.

Однако и этот узор довольно прост. Можно было бы предположить, что, по крайней мере для рассматриваемого нами типа клеточных автоматов, любое выбранное правило всегда будет давать достаточно простой узор. Но теперь нас ждёт первая неожиданность.

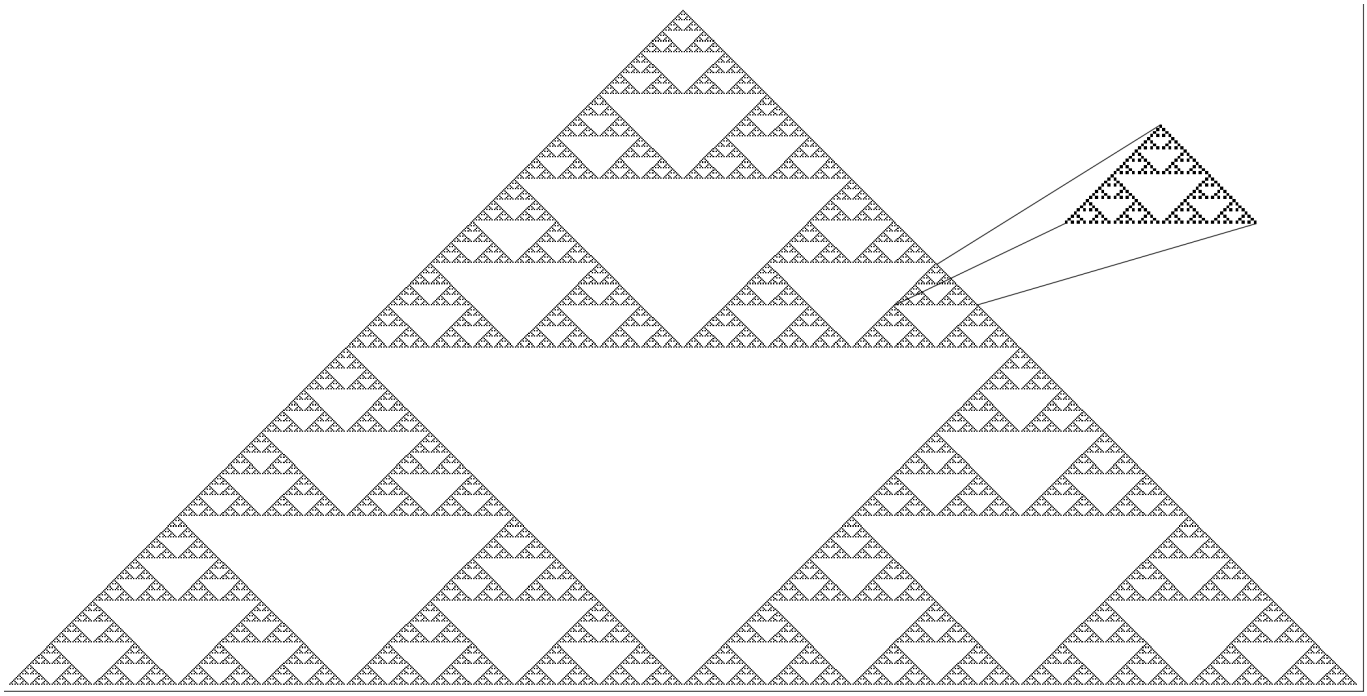
На рисунке ниже показан узор, порождённый клеточным автоматом того же типа, что и прежде, но с немного другим правилом.



Клеточный автомат, порождающий замысловатый вложенный узор. В этом случае правило состоит в том, что клетка должна быть чёрной, если на предыдущем шаге чёрным был один или другой её сосед, но не оба одновременно. Хотя правило очень простое, рисунок показывает, что общий узор, полученный за 50 шагов из единственной чёрной клетки, уже не так прост. Использованное здесь правило можно описать формулой a_i′ = Mod[a_{i−1} + a_{i+1}, 2]. В системе нумерации главы 3 это правило 90 клеточного автомата.

На этот раз правило предписывает делать клетку чёрной, когда на предыдущем шаге чёрным был либо её левый, либо её правый сосед, но не оба одновременно. И снова это правило, бесспорно, совсем простое. Однако теперь рисунок показывает, что порождаемый им узор уже не так прост.

Если запустить клеточный автомат на большее число шагов, как на рисунке ниже, то возникает довольно замысловатый узор. Но теперь можно увидеть, что этому узору присуща совершенно определённая закономерность. Несмотря на всю его сложность, видно, что он действительно состоит из множества вложенных треугольных частей, имеющих в точности одинаковую форму. И, как показывает рисунок, каждая из этих частей по существу представляет собой уменьшенную копию всего узора, внутри которой весьма упорядоченно вложены ещё меньшие копии.



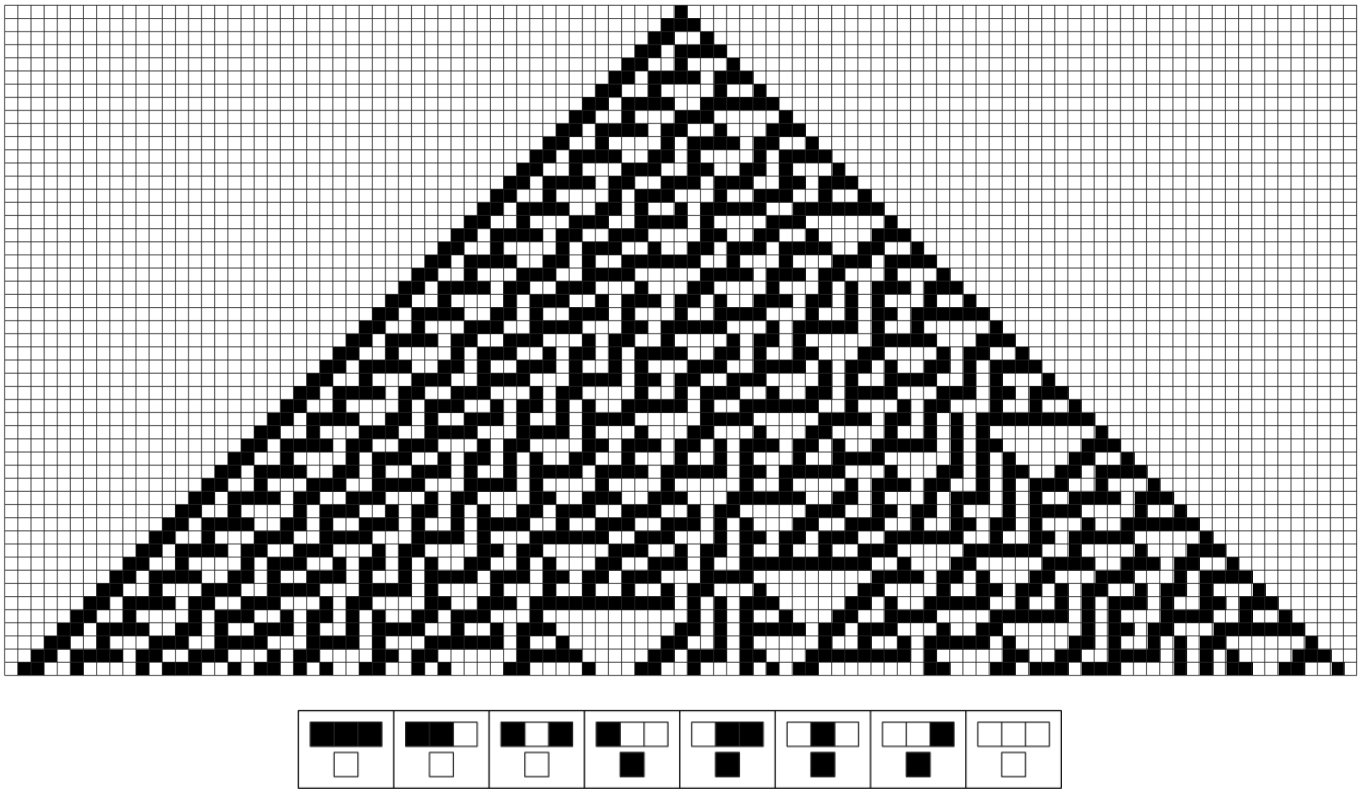
Увеличенная версия узора с предыдущей страницы, теперь показанная без сетки, явно обозначающей каждую клетку. На рисунке представлены пятьсот шагов эволюции клеточного автомата. Полученный узор замысловат, но имеет определённую вложенную структуру. Действительно, как показывает рисунок, каждая треугольная часть по существу является уменьшенной копией всего узора, внутри которой вложены ещё меньшие копии. Узоры с такой вложенной структурой часто называют «фрактальными» или «самоподобными».

Таким образом, все три клеточных автомата, которые мы до сих пор видели, в конечном счёте дают весьма упорядоченные узоры: первый — простой однородный узор, второй — повторяющийся узор, третий — замысловатый, но всё же вложенный узор. И можно было бы предположить, что, по крайней мере для

клеточных автоматов с такими же простыми правилами, как использованные нами, эти три формы поведения исчерпывали бы всё, что вообще можно получить.

Но примечательный факт состоит в том, что это оказывается неверным.

На рисунке ниже показан пример. Используемое правило — я называю его правилом 30 — относится в точности к тому же типу, что и прежде, и его можно описать следующим образом. Сначала рассмотрим каждую клетку и её правого соседа. Если на предыдущем шаге они оба были белыми, то новый цвет клетки берётся таким же, каким прежде был цвет её левого соседа. В противном случае новый цвет берётся противоположным этому.



Клеточный автомат с простым правилом, порождающий узор, который во многих отношениях кажется случайным. Используемое правило относится к тому же типу, что и в предыдущих примерах, и клеточный автомат вновь начинает с единственной чёрной клетки. Но теперь полученный узор очень сложен и почти не обнаруживает общей упорядоченности. Этот рисунок — наш первый пример фундаментального явления: даже простые исходные правила и простые начальные условия способны порождать поведение огромной сложности. В системе нумерации главы 3 показанный здесь клеточный автомат соответствует правилу 30.

Рисунок показывает, что происходит, если начать всего с одной чёрной клетки, а затем снова и снова применять это правило. И то, что мы видим, весьма поразительно — это, вероятно, самое удивительное научное открытие из всех, которые я когда-либо делал. Вместо простого упорядоченного узора, которого можно было бы ожидать, клеточный автомат порождает узор, кажущийся чрезвычайно нерегулярным и сложным.

Но откуда берётся эта сложность? Когда мы задавали систему, мы, несомненно, никак не закладывали её туда непосредственно. Ведь мы использовали лишь простое правило клеточного автомата и начинали лишь с простого начального условия, содержащего единственную чёрную клетку.

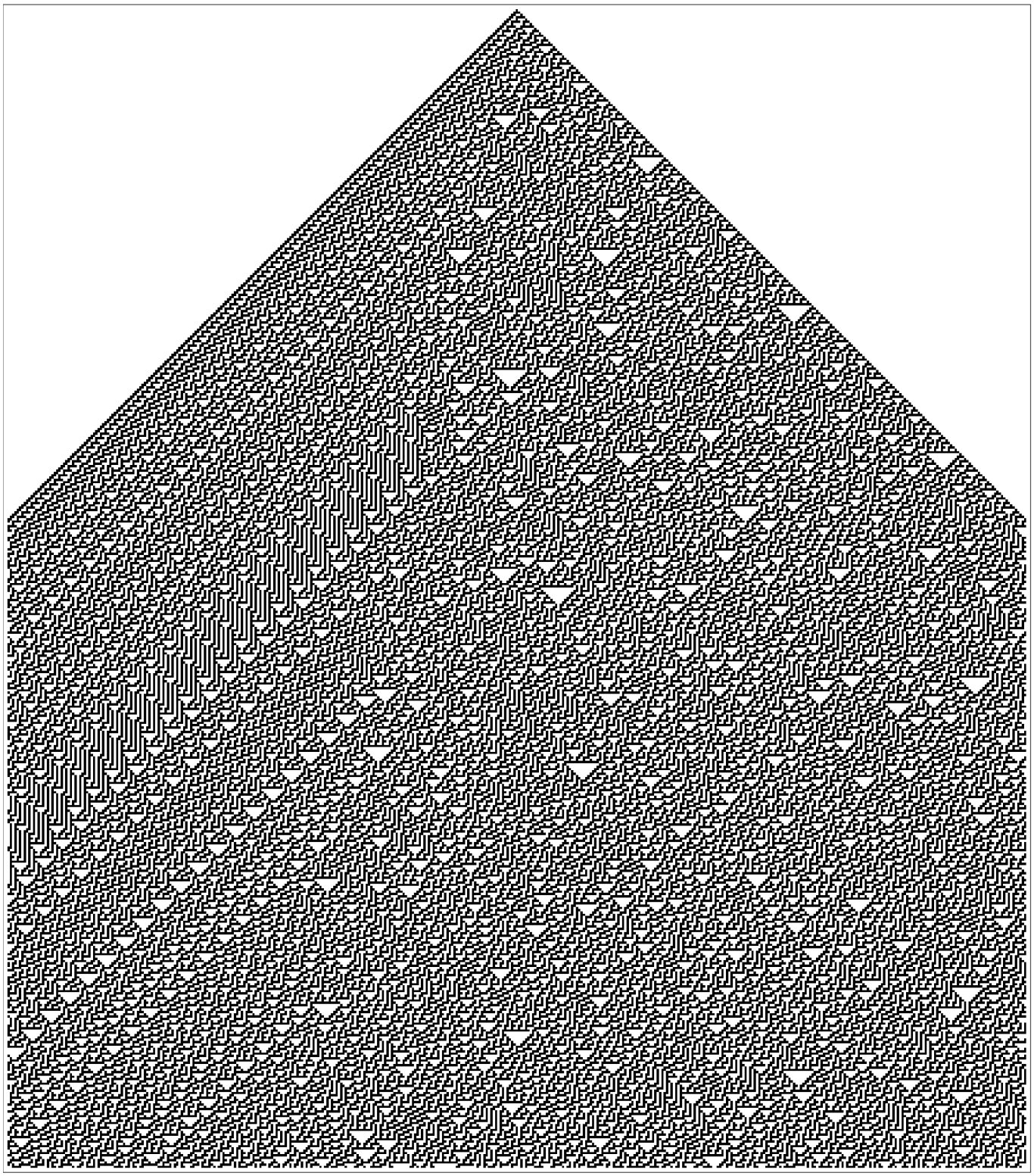
Тем не менее рисунок показывает, что возникающему поведению присуща большая сложность. И действительно, увиденное здесь — первый пример чрезвычайно общего и фундаментального явления, составляющего самую сердцевину нового вида науки, который я развиваю в этой книге. Снова и снова мы будем видеть одно и то же: хотя лежащие в основе системы правила просты и хотя система начинает с простых начальных условий, её поведение всё же может быть очень сложным. И я буду утверждать, что именно это основополагающее явление в конечном счёте ответственно за большую часть сложности, которую мы наблюдаем в природе.

На следующих двух страницах показано всё большее число шагов эволюции клеточного автомата с правилом 30 с предыдущей страницы. Можно было бы подумать, что, возможно, спустя тысячу шагов его поведение в конце концов сведётся к чему-то простому. Но рисунки на следующих двух страницах показывают, что ничего подобного не происходит.

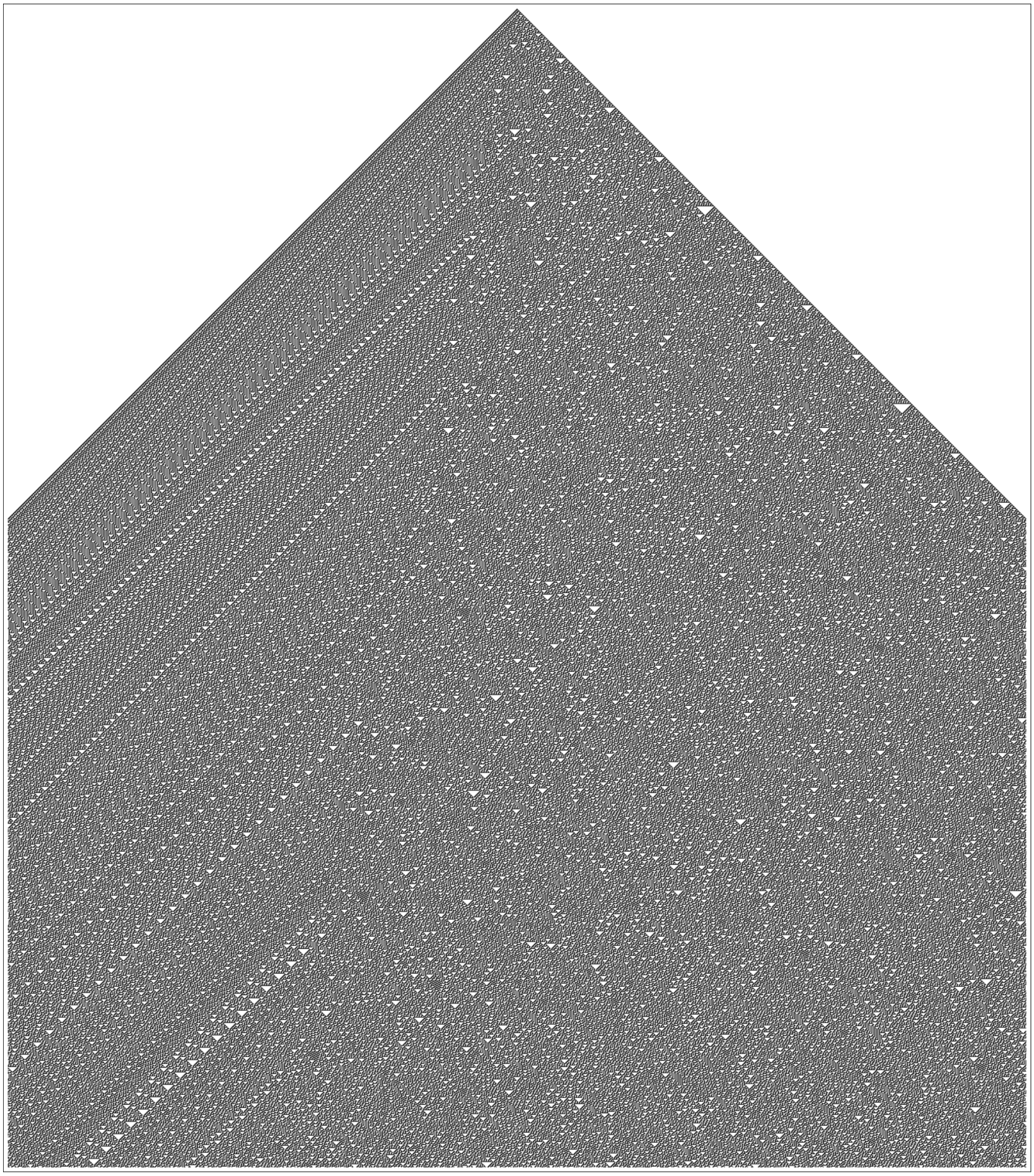
И всё же некоторые закономерности различимы. Например, с левой стороны видны отчётливые диагональные полосы. А по всему узору разбросаны разнообразные белые треугольники и другие небольшие структуры. Однако, учитывая простоту исходного правила, можно было бы ожидать гораздо большего числа закономерностей. И, пожалуй, можно предположить, что наша неспособность увидеть их на рисунках на следующих двух страницах лишь отражает некоторый недостаток человеческого зрения.

Но оказывается, что даже самые изощрённые математические и статистические методы анализа, по-видимому, справляются не лучше. Например, можно рассмотреть последовательность цветов непосредственно под начальной чёрной клеткой. На первом миллионе шагов эта последовательность, к примеру, ни разу не повторяется, и ни один из когда-либо проведённых мною тестов вообще не обнаруживает в ней сколько-нибудь значимого отклонения от полной случайности.

Однако в некотором смысле в такой полной случайности есть определённая простота. Ведь хотя, возможно, и нельзя предсказать, какой



Пятьсот шагов эволюции клеточного автомата с правилом 30 со страницы 27. Возникающий узор продолжает расширяться и влево, и вправо, но здесь показана только та его часть, которая помещается по ширине страницы. Асимметрия между левой и правой сторонами — прямое следствие асимметрии конкретного исходного правила клеточного автомата.



Тысяча пятьсот шагов эволюции по правилу 30. Некоторые закономерности заметны, особенно слева. Но даже после всех этих шагов нет никаких признаков общей упорядоченности; более того, даже при продолжении до миллиона шагов многие стороны полученного узора кажутся совершенно случайными согласно стандартным математическим и статистическим тестам. На рисунке здесь показано в общей сложности чуть менее двух миллионов отдельных клеток.

цвет появится на каком-либо определённом шаге, всё же известно, например, что в среднем чёрный и белый цвета всегда будут встречаться одинаково часто.

Но оказывается, что существуют клеточные автоматы, поведение которых по существу ещё сложнее и у которых даже такие средние величины становится очень трудно предсказать. Рисунки на следующих нескольких страницах дают весьма яркий пример. Основная форма правила остаётся точно такой же, как прежде. Но теперь конкретное используемое правило — я называю его правилом 110 — задаёт новый цвет клетки чёрным во всех случаях, за исключением тех, когда прежние цвета самой клетки и обоих её соседей были одинаковыми, либо когда левый сосед был чёрным, а сама клетка и её правый сосед — белыми.

Узор, полученный с помощью этого правила, обнаруживает удивительное сочетание упорядоченности и нерегулярности. Почти повсюду имеется весьма упорядоченная фоновая текстура, состоящая из множества маленьких белых треугольников, повторяющихся через каждые 7 шагов. А вблизи левого края начинаются диагональные полосы, которые возникают с интервалами ровно в 80 шагов.

Но с правой стороны узор гораздо менее упорядочен. На протяжении первых нескольких сотен шагов там действительно есть область, которая кажется по существу случайной. Однако к низу первой страницы от этой области остаются лишь три экземпляра довольно простой повторяющейся структуры.

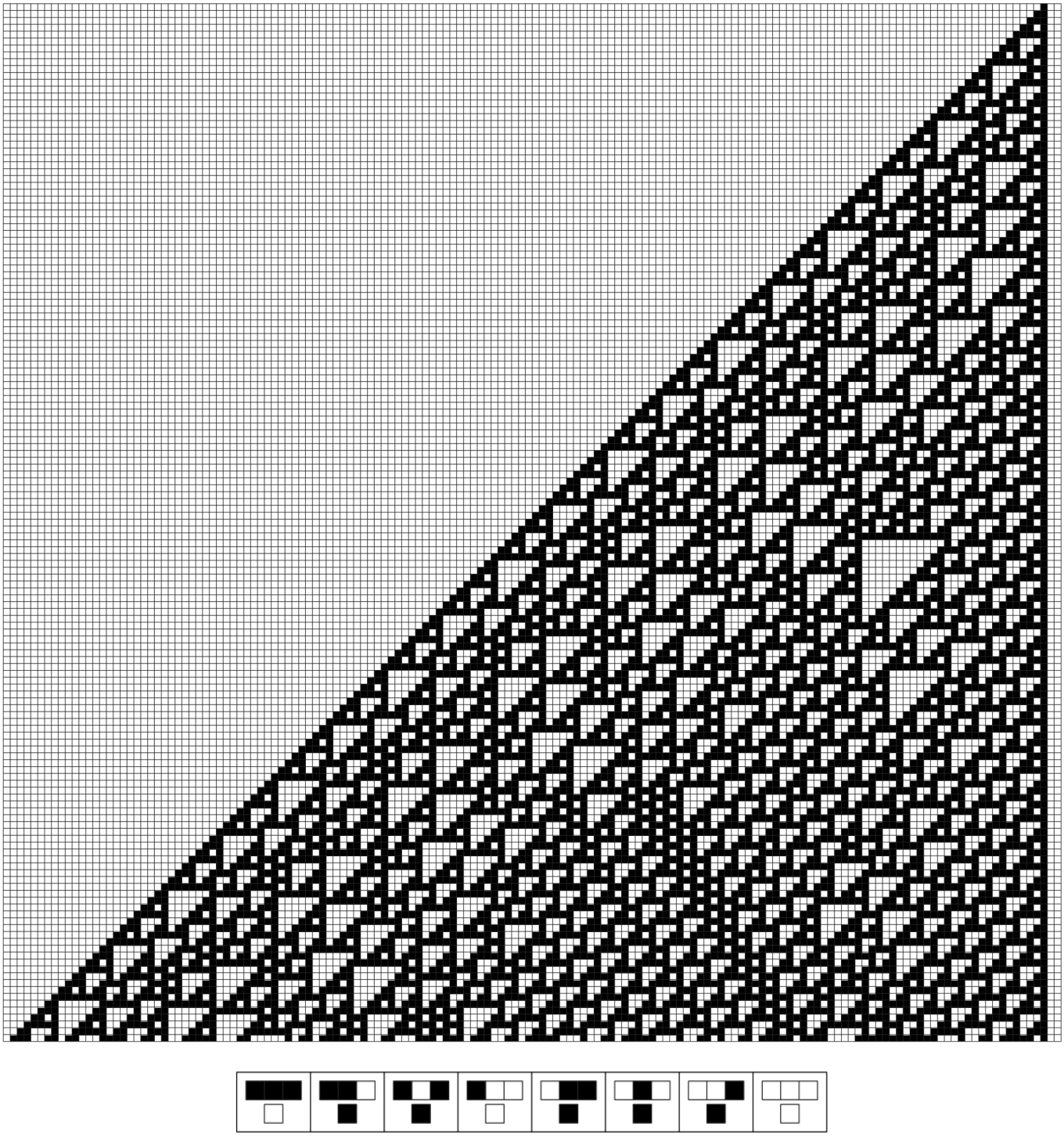
И всё же вверху следующей страницы приход диагональной полосы слева вновь запускает более сложное поведение. По мере развития системы возникают разнообразные чётко выраженные локализованные структуры.

Некоторые из этих структур остаются неподвижными, как те, что находятся внизу первой страницы, тогда как другие равномерно движутся вправо или влево с различными скоростями. Сама по себе каждая из этих структур ведёт себя довольно просто. Но, как показывают рисунки, их разнообразные взаимодействия могут приводить к весьма сложным последствиям.

В результате становится почти невозможно предсказать, даже приблизительно, что будет делать клеточный автомат.

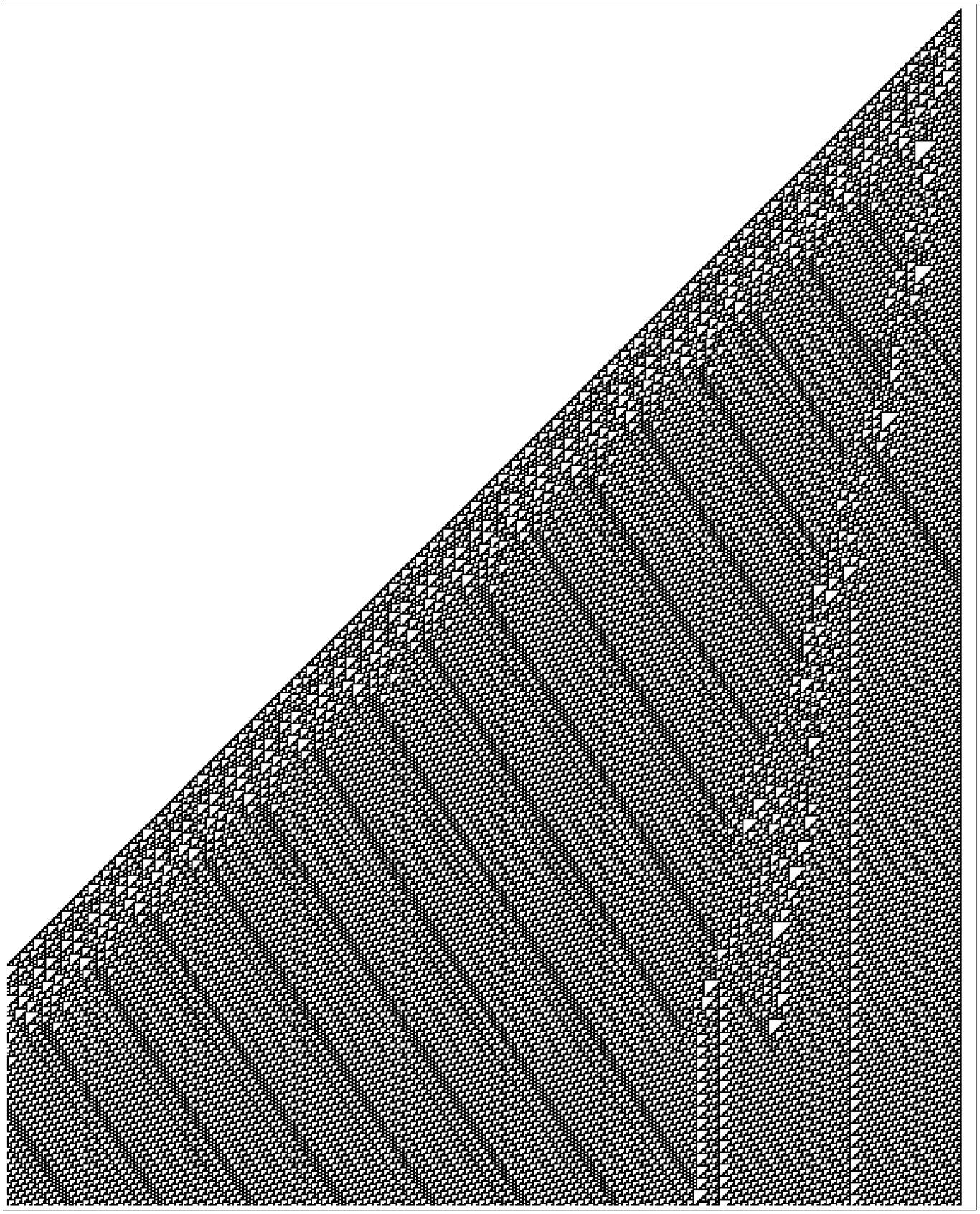
Уничтожат ли все возникающие структуры в конце концов друг друга, оставив лишь весьма упорядоченный узор? Или будет появляться всё больше и больше структур, пока весь узор не станет совершенно случайным?

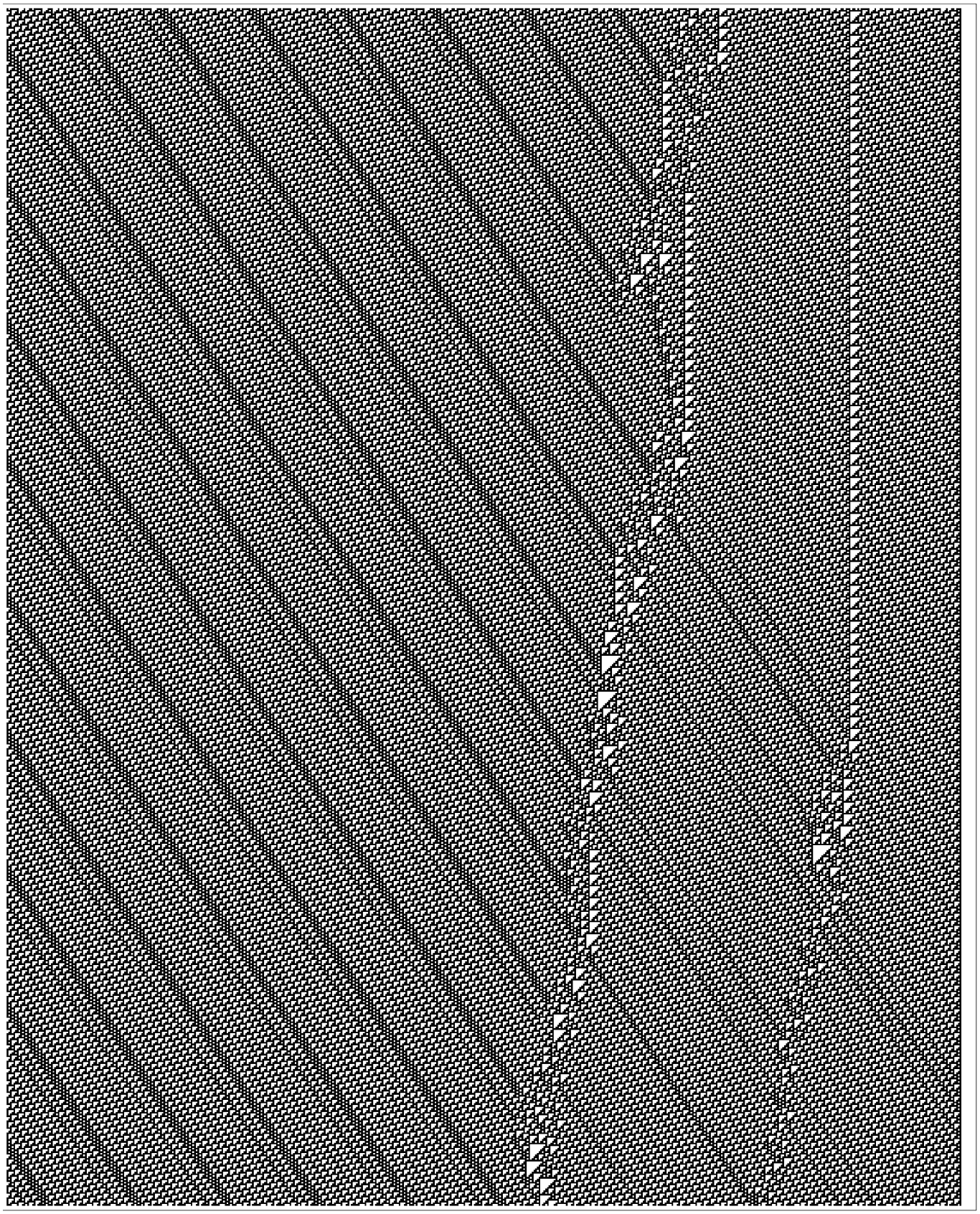
Единственный надёжный способ ответить на эти вопросы, по-видимому, состоит в том, чтобы просто запустить клеточный автомат на столько шагов, сколько потребуется, и

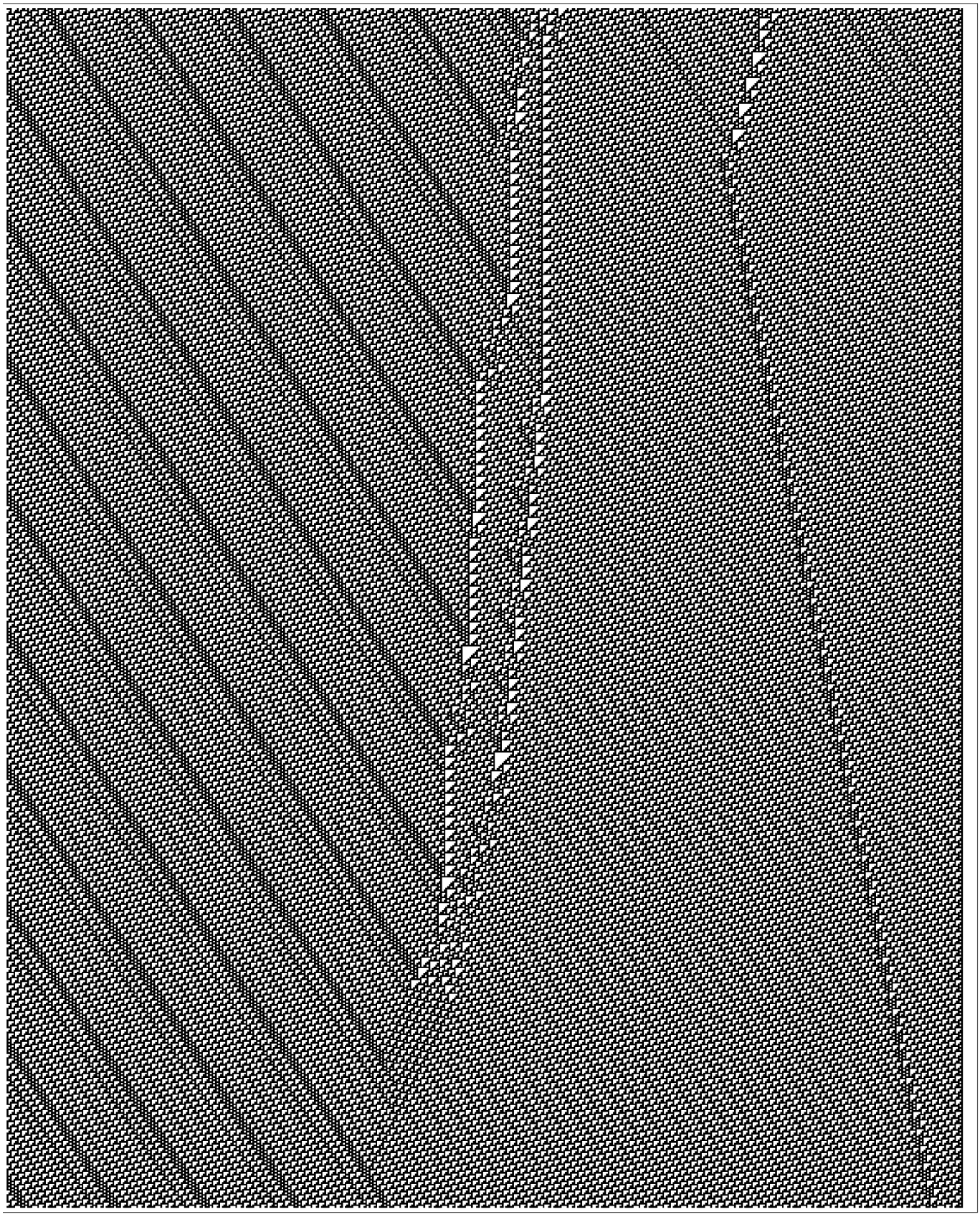


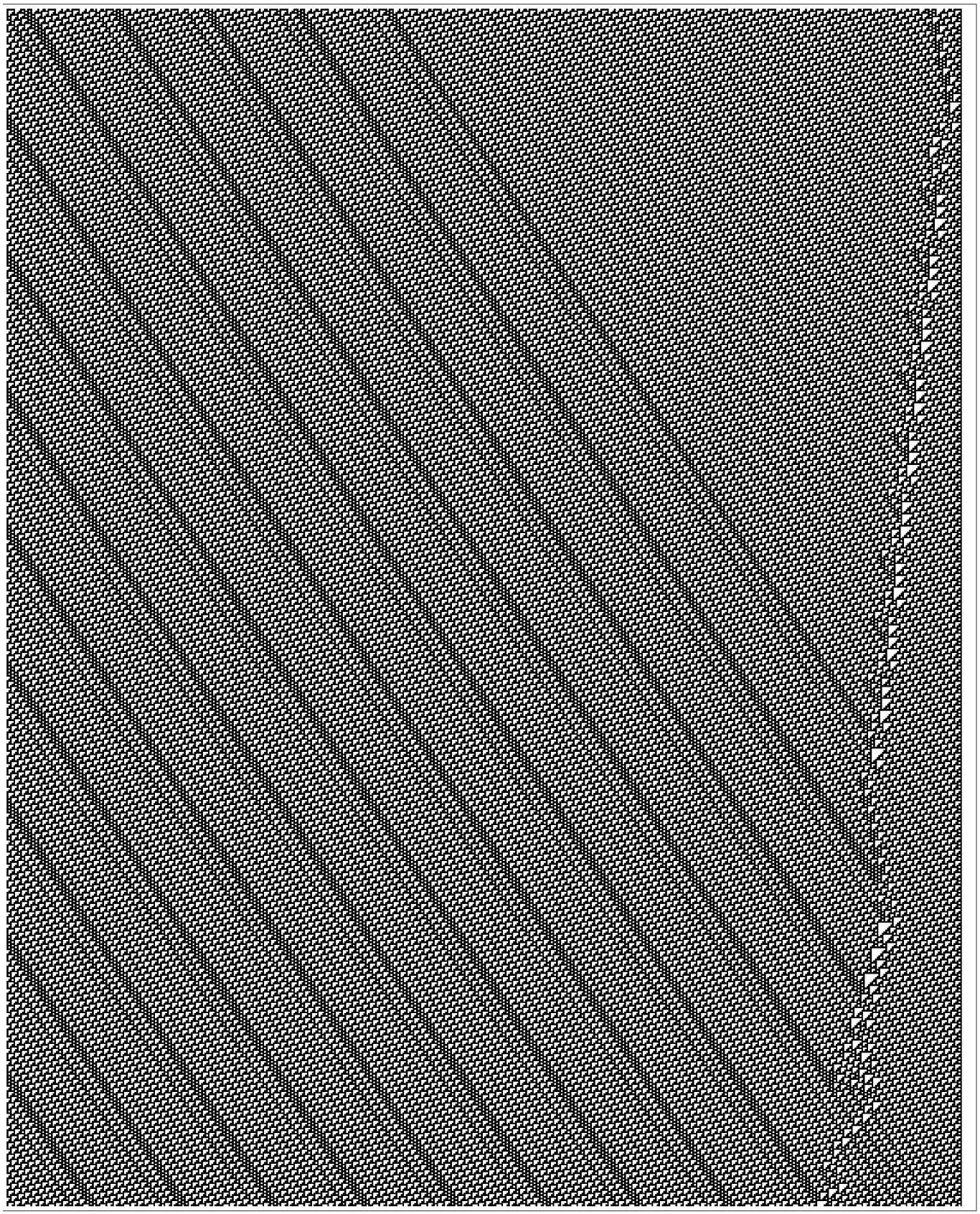
Клеточный автомат, поведение которого не кажется ни весьма упорядоченным, ни полностью случайным. Рисунок получен за 150 шагов применения показанного простого правила, начиная с единственной чёрной клетки. Обратите внимание: конкретное правило, использованное здесь, даёт узор, который расширяется влево, но не вправо. В системе, определённой в главе 3, это правило имеет номер 110.

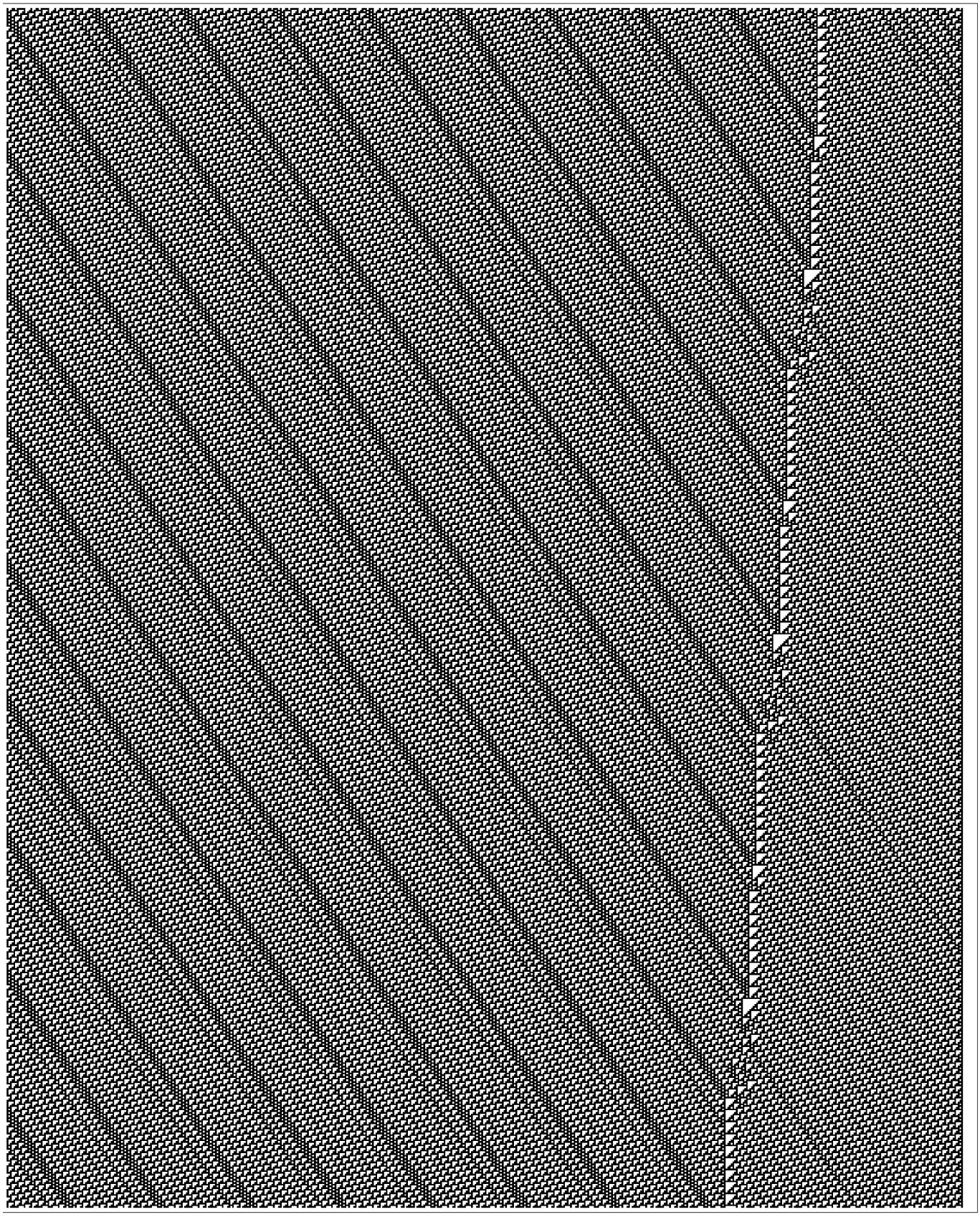
Дальнейшие шаги развития узора, показанного выше. На каждой следующей странице представлены в общей сложности 700 шагов. Узор бесконечно продолжает расширяться влево, но показана лишь та его часть, которая помещается по ширине каждой страницы. Долгое время неясно, как в конечном счёте будет выглядеть правая часть узора. Но после 2780 шагов возникает довольно простая повторяющаяся структура. Обратите внимание: для получения следующих рисунков требуется в общей сложности около 12 миллионов применений исходного правила клеточного автомата к отдельным клеткам.

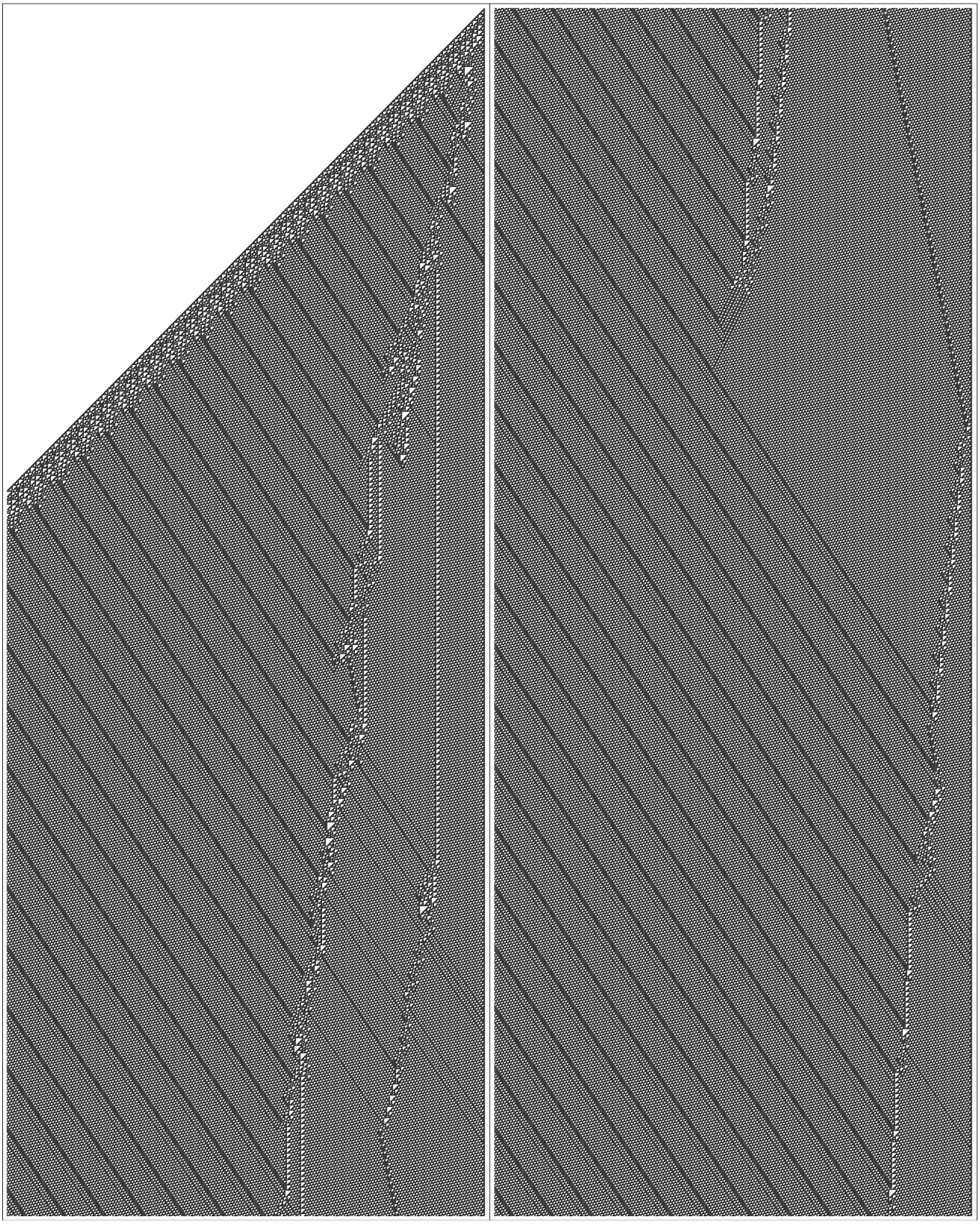












наблюдать за тем, что происходит. Как оказывается, в показанном здесь конкретном случае исход окончательно проясняется примерно через 2780 шагов: одна структура сохраняется, и её взаимодействие с периодическими полосами, приходящими слева, порождает поведение, повторяющееся каждые 240 шагов.

Как бы твёрдо ни был человек убеждён в том, что простые программы способны порождать лишь простое поведение, рисунки на предыдущих нескольких страницах должны навсегда избавить его от этого представления. И, пожалуй, самое удивительное в этих рисунках — то, насколько мало в них в конечном счёте остаётся следов простоты правила клеточного автомата, с помощью которого они были получены.

Можно было бы, например, подумать, что, поскольку все клетки клеточного автомата следуют совершенно одному и тому же правилу, на рисунках вроде тех, что были на последних нескольких страницах, все клетки должны каким-то очевидным образом делать одно и то же. Но вместо этого они, по-видимому, делают совершенно разные вещи. Некоторые из них, например, входят в состав регулярного фона, а другие — в состав той или иной локализованной структуры. Это возможно потому, что, хотя отдельные клетки следуют одному и тому же правилу, различные конфигурации клеток с разными последовательностями цветов вместе могут порождать самые разные виды поведения.

Глядя только на исходное правило клеточного автомата, практически невозможно было бы всё это предвидеть. Но с помощью соответствующих компьютерных экспериментов легко выяснить, что происходит на самом деле, — и тем самым начать исследование целого нового мира замечательных явлений, связанных с простыми программами.

Необходимость новой интуиции

Рисунки в предыдущем разделе ясно показывают, что для возникновения чрезвычайно сложного поведения достаточно очень простых правил. И всё же поначалу в это может быть почти невозможно поверить. Ведь это противоречит некоторым из наших самых основополагающих интуитивных представлений о том, как обычно устроены вещи.

Единое изображение поведения, показанного на предыдущих пяти страницах. Всего представлено 3200 шагов. Обратите внимание: это более чем вдвое больше, чем на рисунке на странице 30.

Повседневный опыт приучил нас ожидать, что предмет, который выглядит сложным, должен был быть создан сложным способом. Поэтому, например, увидев сложное механическое устройство, мы обычно предполагаем, что чертежи, по которым его изготовили, тоже должны быть в некотором смысле соответственно сложными.

Но результаты в конце предыдущего раздела показывают, что по крайней мере иногда такое предположение может быть совершенно неверным. Ведь увиденные нами узоры, по существу, строятся по очень простым планам: достаточно начать с одной чёрной клетки, а затем многократно применять простое правило клеточного автомата. И всё же то, что возникает по этим планам, демонстрирует колоссальную сложность.

Так что же заставляет нашу обычную интуицию давать сбой? По-видимому, важнее всего то, что она в основном сформировалась из опыта создания вещей и инженерной работы, где так уж получается, что систем, подобных рассмотренным в предыдущем разделе, избегают.

Ведь обычно мы начинаем с поведения, которое хотим получить, а затем пытаемся спроектировать систему, способную его обеспечить. Но, чтобы делать это надёжно, приходится ограничиваться системами, поведение которых легко понять и предсказать: если мы не можем предвидеть, как будет вести себя система, то не можем быть уверены, что она сделает то, что нам нужно.

Однако, в отличие от инженерии, природа не связана таким ограничением. Поэтому ничто не мешает появлению систем вроде тех, что были показаны в конце предыдущего раздела. И действительно, один из важных выводов этой книги состоит в том, что такие системы в природе на самом деле весьма распространены.

Но поскольку единственные ситуации, в которых мы регулярно знаем и лежащие в основе правила, и общее поведение, связаны с созданием вещей или инженерной работой, в обычных условиях у нас так и не складывается интуиция относительно систем вроде тех, что были в конце предыдущего раздела.

Есть ли тогда в повседневном опыте хоть что-нибудь, что могло бы подсказать нам, какие явления возникают в этих системах? Вероятно, ближе всего к этому размышления об особенностях практического применения компьютеров.

Ведь мы знаем, что компьютеры способны выполнять множество сложных задач. Однако на уровне базового аппаратного обеспечения обычный компьютер может исполнять всего несколько десятков видов простых логических, арифметических и иных инструкций. И в некоторой степени тот факт, что, исполняя большое количество таких

инструкций, можно получить самые разные виды сложного поведения, сходен с явлением, которое мы наблюдали у клеточных автоматов.

Но есть важное различие. Хотя отдельные машинные инструкции, исполняемые компьютером, могут быть весьма простыми, определяемая программой последовательность этих инструкций может быть длинной и сложной. И действительно, как и в других областях инженерии, обычный опыт разработки программ показывает: чтобы заставить компьютер делать что-то сложное, необходимо составить программу, которая сама будет в некотором смысле соответственно сложной.

В системе наподобие клеточного автомата лежащие в её основе правила можно считать приблизительными аналогами машинных инструкций компьютера, а начальные условия — приблизительными аналогами программы. Однако в предыдущем разделе мы увидели, что у клеточных автоматов простыми могут быть не только лежащие в основе правила, но и начальные условия — состоящие, скажем, всего из одной чёрной клетки, — и всё же возникающее поведение может оказаться чрезвычайно сложным.

Таким образом, хотя практическое применение компьютеров даёт намёк на часть того, что мы увидели в предыдущем разделе, явление в целом гораздо масштабнее и сильнее. И в некотором смысле самая загадочная его сторона состоит в том, что оно, по-видимому, предполагает получение чего-то из ничего.

Ведь клеточные автоматы, которые мы задали, по любым меркам просты в описании. И всё же, запустив их, мы в итоге получили узоры настолько сложные, что они, казалось, вообще не поддавались никакому простому описанию.

Можно было бы надеяться, что для понимания столь фундаментального явления удастся опереться на какой-то уже существующий вид интуиции. Но на деле, по-видимому, ни одна область повседневного опыта не даёт нам того, что требуется. Поэтому у нас нет иного выбора, кроме как попытаться развить совершенно новый вид интуиции.

И единственный разумный способ сделать это — познакомиться с большим количеством примеров. До сих пор мы видели лишь несколько примеров, и все они относились к клеточным автоматам. Но в следующих главах мы увидим гораздо больше примеров — как в клеточных автоматах, так и во всевозможных других системах. Осваивая эти примеры, в конце концов можно развить такую интуицию, при которой обнаруженные мной основные явления начинают казаться почти очевидными и неизбежными.

Почему эти открытия не были сделаны раньше

Главный результат этой главы — то, что программы, основанные на простых правилах, способны порождать поведение огромной сложности, — кажется настолько фундаментальным, что можно предположить: он должен был быть обнаружен давным-давно. Но этого не произошло, и полезно разобраться в некоторых причинах, по которым этого не произошло.

В истории науки довольно часто именно новые технологии в конечном счёте дают начало развитию новых областей фундаментальной науки. Так, например, технология телескопов привела к современной астрономии, а технология микроскопов — к современной биологии. И теперь во многом таким же образом компьютерная технология привела к тому новому виду науки, который я описываю в этой книге.

В самом деле, эту главу и несколько следующих за ней в некотором смысле можно рассматривать как изложение некоторых самых простых экспериментов, которые можно провести с помощью компьютеров. Но почему такие простые эксперименты никогда не проводились раньше?

Одна из причин состоит просто в том, что они не относились к основному направлению ни одной существующей области науки или математики. Но более важная причина в том, что привычная интуиция традиционной науки не давала оснований думать, будто их результаты окажутся интересными.

И действительно, если бы было известно, что эти эксперименты стоят усилий, многие из них можно было бы провести ещё задолго до появления компьютеров. Ведь, хотя это может быть несколько утомительно, проследить поведение чего-то вроде клеточного автомата вручную вполне возможно. Причём для этого совершенно не нужны сложные идеи из математики или какой-либо другой области: достаточно понимать, как многократно применять простые правила.

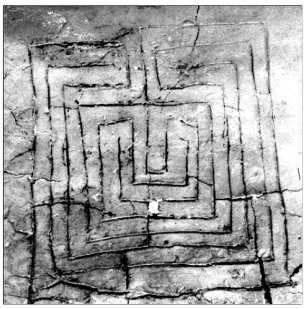
Глядя на исторические образцы орнаментального искусства на странице напротив, трудно найти основания полагать, что поведение многих клеточных автоматов нельзя было бы проследить много веков или даже тысячелетий назад. Возможно, однажды будет найден какой-нибудь вавилонский артефакт, созданный с помощью клеточного автомата с правилом 30 со страницы 27. Но я в этом очень сомневаюсь. Я склонен думать, что, если бы рисунки, подобные изображению на странице 27, действительно увидели ещё в древности, наука пошла бы совсем иным путём, чем тот, по которому она пошла на самом деле.



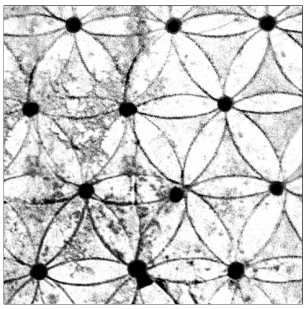
22 000 год до н. э. (палеолит)



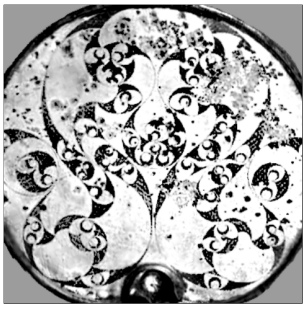
3500 год до н. э. (шумерский)



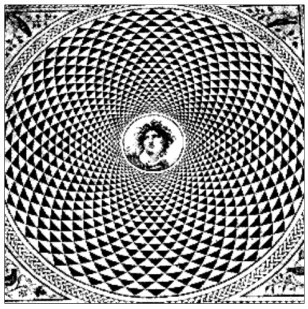
1200 год до н. э. (греческий)



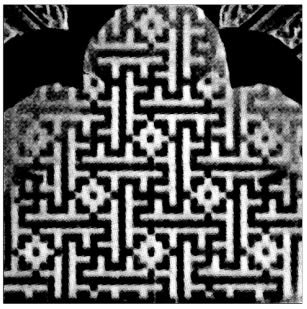
IX век до н. э. (финикийский)



I век до н. э. (кельтский)



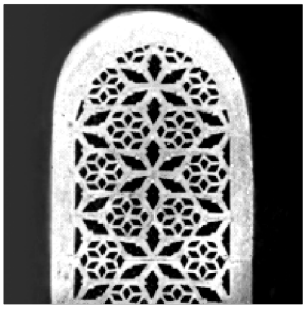
II век (римский)



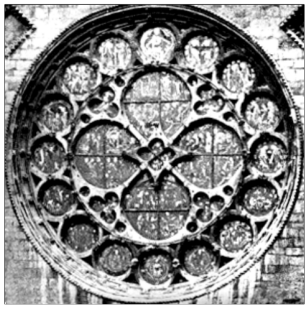
VIII век (исламский)



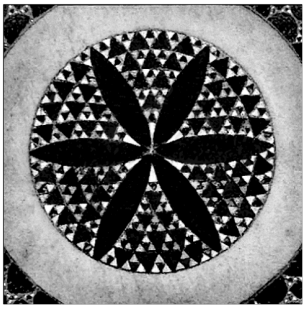
VIII век (кельтский)



XII век (итальянский)



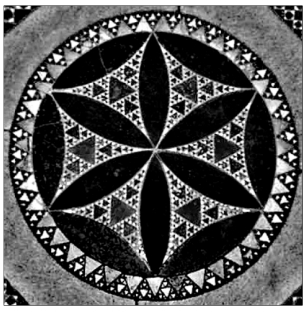
XIII век (английский)



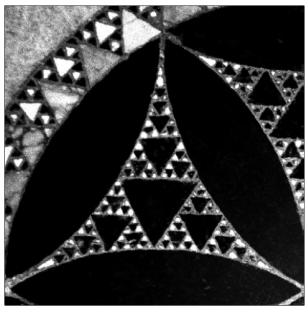
XIII век (итальянский)



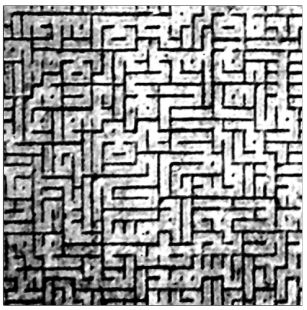
XIII век (итальянский)



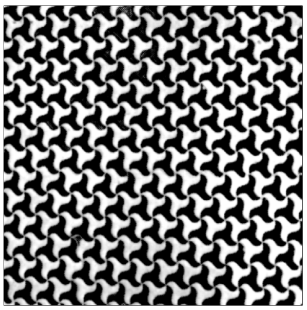
XIII век (итальянский)



XIII век (итальянский)



XIV век (исламский)



XIV век (исламский)

Исторические образцы орнаментального искусства. Повторяющиеся узоры встречаются часто, попадаются и некоторые вложенные узоры, однако более сложные виды узоров, обсуждаемые в этой главе, по-видимому, никогда не использовались. Обратите внимание: предпоследнее изображение — не абстрактный орнамент, а текст, написанный сильно стилизованной формой арабского письма.

Предположительно, ещё в глубокой древности предпринимались попытки выяснить, могут ли простые абстрактные правила воспроизводить поведение природных систем. Но, насколько можно судить, испытывались только правила, связанные с обычной геометрией и арифметикой. Такие правила позволяли получать лишь довольно простое поведение: его хватало для объяснения некоторых закономерностей, наблюдаемых в астрономии, но оно не могло охватить многое из того, что встречается в других областях природы.

Возможно, именно поэтому обычно считалось, что очень многие стороны природного мира просто недоступны человеческому пониманию. Однако успехи, достигнутые благодаря математическому анализу в конце XVII века, наконец начали опровергать это убеждение. С появлением математического анализа удалось добиться настоящего успеха: абстрактные правила, созданные человеческой мыслью, стали использовать для воспроизведения самых разных явлений природного мира.

Но те конкретные правила, которые оказались действенными, были достаточно сложны и основывались на определённых видах математических уравнений. Из наблюдения за сложностью этих правил постепенно стало складываться неявное убеждение, что почти ни в одном важном случае более простые правила не будут полезны для воспроизведения поведения природных систем.

В XVIII и XIX веках применение правил, основанных на математических уравнениях, к анализу физических явлений приносило всё большие успехи. А после впечатляющих результатов, достигнутых с помощью математических уравнений в физике в начале XX века, возникло почти всеобщее убеждение, что в конечном счёте абсолютно каждый аспект природного мира будет объяснён с помощью таких уравнений.

Разумеется, многие явления не поддавались этому подходу с лёгкостью, но обычно предполагалось, что, стоит лишь выполнить необходимые вычисления, объяснение в терминах математических уравнений в конце концов найдётся.

Начиная с 1940-х годов развитие электронных компьютеров значительно расширило круг доступных вычислений. Но, к разочарованию исследователей, большинство реально предпринятых вычислений не дало принципиально нового понимания. В результате многие пришли к убеждению — а некоторые придерживаются его и по сей день, — что компьютеры никогда не смогут внести настоящий вклад в решение вопросов фундаментальной науки.

Однако был упущен решающий момент: возможности компьютеров не ограничиваются лишь вычислением следствий математических уравнений. И действительно, в этой главе мы увидели, что фундаментальные открытия можно совершать, просто непосредственно изучая поведение даже некоторых самых простых компьютерных программ.

Оглядываясь назад, можно усмотреть иронию в том, что идея использовать простые программы в качестве моделей природных систем не возникла на заре вычислительной техники. Ведь с системами вроде клеточных автоматов на ранних компьютерах было бы несравненно проще работать, чем с математическими уравнениями. Но проблема заключалась в том, что компьютерное время было дорогим ресурсом, и поэтому считалось неразумным рисковать, пробуя что-либо, кроме хорошо зарекомендовавших себя математических моделей.

Однако к концу 1970-х годов ситуация изменилась: большие объёмы компьютерного времени стали легко доступны. Именно это позволило мне в 1981 году начать эксперименты с клеточными автоматами.

Как я уже отмечал выше, в принципе ничто не требует использовать компьютер для изучения клеточных автоматов. Но на практике трудно представить, чтобы кому-либо в наше время хватило терпения вручную построить множество изображений клеточных автоматов. Ведь на создание рисунка со страницы 27 вручную уходит примерно час, а на создание рисунка со страницы 29 потребовалось бы несколько недель.

Между тем даже на ранних больших ЭВМ данные для этих рисунков можно было получить соответственно за несколько секунд и несколько минут. Но дело в том, что, рассматривая всего один или два рисунка, крайне маловероятно обнаружить фундаментальные явления того рода, о которых говорится в этой главе. И мне для этого действительно пришлось провести достаточно масштабные компьютерные эксперименты со значительным числом различных клеточных автоматов.

Если уже имеется ясное представление об основных особенностях того или иного явления, то дополнительные подробности часто можно получить с помощью довольно узко направленных экспериментов. Но, по моему опыту, единственный способ обнаружить явления, о существовании которых ещё не известно, — проводить очень систематические и общие эксперименты, а затем рассматривать результаты с как можно меньшим числом предвзятых представлений. И хотя для создания отдельных изображений клеточных автоматов достаточно лишь довольно элементарной

компьютерной технологии, для проведения масштабных систематических экспериментов требуется существенно больше.

Многие мои открытия, касающиеся клеточных автоматов, действительно были прямым следствием применения всё более совершенной компьютерной технологии.

Например, схему классификации клеточных автоматов со случайными начальными условиями, описанную в начале главы 6, я обнаружил, когда впервые одновременно рассмотрел большое количество различных клеточных автоматов на графических дисплеях высокого разрешения. Подобным же образом случайность правила 30 (страница 27) я обнаружил в процессе подготовки крупных моделирований для одного из ранних компьютеров с параллельной обработкой. А в последние годы я открыл огромное множество новых явлений благодаря возможности легко организовывать большое количество компьютерных экспериментов в Mathematica.

Поэтому, несомненно, одна из главных причин, по которым открытия, описанные в этой главе, не были сделаны до 1980-х годов, состоит просто в том, что ещё не существовало компьютерной технологии, достаточно мощной для проведения необходимых исследовательских экспериментов.

Но помимо практической возможности провести такие эксперименты необходимо было ещё и допустить, что их вообще стоит проводить. И здесь компьютерная технология вновь сыграла решающую роль. Ведь значительная часть необходимой интуиции сформировалась у меня из практического опыта работы с компьютерами.

Простой пример: можно было бы предположить, что системы наподобие клеточных автоматов, состоящие из дискретных клеток, никогда не смогут воспроизводить реалистичные природные формы. Но если знать, как устроены компьютерные дисплеи, становится ясно, что это не так. Ведь дисплей компьютера, подобно клеточному автомату, представляет собой регулярный массив дискретных клеток, или пикселей. Однако практический опыт показывает, что такие дисплеи способны создавать вполне реалистичные изображения даже при достаточно небольшом числе пикселей.

Более существенный пример: можно было бы предположить, что простая структура программ клеточных автоматов позволяет легко предвидеть их поведение. Но из практического опыта программирования известно, что предвидеть, что сделает даже простая программа, обычно очень трудно. Именно поэтому ошибки в программах так распространены. Ведь если бы можно было просто взглянуть на программу

и сразу понять, что она будет делать, проверить отсутствие в ней ошибок было бы легко.

Такие представления, как трудность поиска ошибок, не имеют очевидной связи с традиционными научными идеями. Возможно, поэтому даже спустя несколько десятилетий использования компьютеров практически никакая интуиция такого рода, выработанная в практической работе с ними, не проникла в фундаментальную науку. Но в 1981 году так получилось, что я уже несколько лет глубоко занимался и практическими вычислениями, и фундаментальной наукой, а потому находился в почти уникальном положении, позволявшем применять в фундаментальной науке идеи, почерпнутые из практической работы с компьютерами.

И всё же, несмотря на это, в моих открытиях, касавшихся клеточных автоматов, по-прежнему была значительная доля везения. Ведь, как я упоминал на странице 19, самые первые мои эксперименты с клеточными автоматами показывали лишь очень простое поведение, и я продолжал работу только потому, что технически проводить дальнейшие эксперименты мне было очень легко.

Даже после того, как я заметил первые признаки сложности в клеточных автоматах, прошло ещё несколько лет, прежде чем я обнаружил весь набор примеров, приведённых в этой главе, и осознал, насколько легко может возникать сложность в системах наподобие клеточных автоматов.

Отчасти это заняло так много времени потому, что эксперименты требовали всё более совершенной компьютерной технологии. Но более важная причина заключалась в необходимости развить новую интуицию. Почти на каждом этапе интуиция традиционной науки вела меня в неверном направлении. Однако я обнаружил, что интуиция, приобретённая в практической работе с компьютерами, действовала лучше. И хотя порой она тоже вводила в заблуждение, в конечном счёте она сыграла довольно важную роль, помогая мне встать на верный путь.

Таким образом, есть две совершенно разные причины, по которым результаты этой главы трудно было бы открыть намного раньше того момента, когда компьютерная технология достигла уровня 1980-х годов. Во-первых, необходимые компьютерные эксперименты нельзя было проводить настолько легко, чтобы кто-то с большой вероятностью решился их попробовать. Во-вторых, нужные интуитивные представления о вычислениях не могли бы легко сформироваться без обширного опыта практической работы с компьютерами.

Но теперь, когда результаты этой главы известны, можно оглянуться назад и увидеть немало случаев в прошлом, когда к их открытию подходили хотя бы отчасти близко.

Оказывается, двумерные варианты клеточных автоматов рассматривались как возможные идеализированные модели биологических систем ещё в начале 1950-х годов. Но вплоть до моих работ 1980-х реальные исследования клеточных автоматов состояли главным образом в построении довольно сложных наборов правил, для которых можно было показать, что они приводят к определённым видам достаточно простого поведения.

Вопрос о том, может ли в клеточных автоматах возникать сложное поведение, время от времени поднимался, но, исходя из инженерной интуиции, обычно предполагали, что для получения сколько-нибудь существенной сложности нужны очень сложные исходные правила. В результате идея изучать клеточные автоматы с простыми правилами так и не возникла, а потому ничего подобного экспериментам, описанным в этой главе, никогда не проводилось.

Однако в других областях системы, фактически основанные на простых правилах, изучались довольно часто, и иногда сложное поведение действительно наблюдалось. Но в отсутствие концептуальной основы, позволяющей понять его значение, такое поведение обычно либо полностью игнорировали, либо считали своего рода курьёзом, не имеющим особого фундаментального значения.

И действительно, даже на самых ранних этапах истории традиционной математики уже имелись признаки основного явления сложности. Один пример, известный значительно дольше двух тысяч лет, связан с распределением простых чисел (см. страницу 132). Правила получения простых чисел просты, и всё же их распределение во многих отношениях кажется случайным. Однако почти без исключений математические исследования простых чисел были сосредоточены не на этой случайности, а на доказательстве наличия различных закономерностей в их распределении.

Ещё один ранний признак явления сложности можно было бы увидеть в последовательности цифр такого числа, как π ≃ 3.141592653… (см. страницу 136). К XVIII веку было вычислено уже более ста цифр числа π, и они выглядели совершенно случайными. Но этот факт рассматривался главным образом как курьёз; по-видимому, так и не возникла мысль о том,

что может существовать общее явление, при котором простые правила — например, правила вычисления π — способны давать сложные результаты.

В начале XX века в нескольких областях математики были построены различные явные примеры многократного применения простых правил к числам, последовательностям или геометрическим узорам. Иногда при этом наблюдалось вложенное, или фрактальное, поведение. А в нескольких случаях наблюдалось и существенно более сложное поведение. Но сама сложность такого поведения обычно воспринималась как свидетельство того, что оно не может иметь отношения к серьёзной математической работе и представляет лишь развлекательный интерес.

Когда в 1940-х годах начали использовать электронные компьютеры, возможностей заметить явление сложности стало гораздо больше. Оглядываясь назад, можно сказать, что существенная сложность, вероятно, действительно возникала во многих научных вычислениях. Однако эти вычисления почти всегда основывались на традиционных математических моделях, а поскольку предшествующий анализ таких моделей сложности не выявлял, обычно предполагалось, что любая сложность в компьютерных вычислениях — всего лишь ложный эффект используемых приближений.

Один из классов систем, в которых некоторые виды сложности были замечены в 1950-х годах, — так называемые итерируемые отображения. Но, как я расскажу на странице 149, традиционная математика, применявшаяся для анализа таких систем, в конечном счёте сосредоточилась лишь на некоторых отдельных свойствах и полностью упустила главное явление, обнаруженное в этой главе.

В практических вычислениях часто полезно получать последовательности чисел, которые кажутся случайными. Начиная с 1940-х годов было изобретено несколько простых процедур для получения таких последовательностей. Но, возможно, из-за того, что эти процедуры всегда казались довольно частными и специально приспособленными к задаче, из них не было сделано общих выводов о случайности и сложности.

Примерно в том же направлении в конце 1950-х годов изучались системы, похожие на клеточные автоматы, обсуждаемые в этой главе: они предназначались для получения случайных последовательностей, используемых в криптографии. Почти все полученные результаты до сих пор являются военной тайной, но я не думаю, что там были обнаружены какие-либо явления, подобные описанным в этой главе.

Вообще, в рамках основного направления науки сложившаяся привычная интуиция чрезвычайно мешала кому-либо представить, что стоит изучать поведение

очень простых видов компьютерных программ, рассматриваемых в этой главе. Однако за пределами основного научного направления некоторая работа такого рода всё же велась. Например, в 1960-х годах первые компьютерные энтузиасты пробовали запускать различные простые программы и обнаружили, что в некоторых случаях эти программы способны порождать вложенные узоры.

Затем, в начале 1970-х годов, в сфере развлекательных вычислений возник значительный интерес к конкретному двумерному клеточному автомату, известному как игра «Жизнь», поведение которого в некоторых отношениях сходно с поведением клеточного автомата с правилом 110, обсуждаемого в этой главе. Огромные усилия были потрачены на поиск структур, достаточно простых и предсказуемых, чтобы их можно было использовать как идеализированные инженерные компоненты. И хотя сложное поведение наблюдалось, обычно к нему относились как к помехе, которой по возможности следует избегать.

В некотором смысле удивительно, что с игрой «Жизнь» удалось сделать так много, так и не исследовав значительно более простые одномерные клеточные автоматы, рассматриваемые в этой главе. И отсутствие связи с фундаментальной наукой, несомненно, по крайней мере отчасти было тому причиной.

Но каковы бы ни были причины, факт остаётся фактом: несмотря на многочисленные подсказки на протяжении нескольких столетий, основное явление, которое я описал в этой главе, раньше так и не было обнаружено.

В истории науки нередко бывает так, что после обнаружения нового общего явления становится видно: свидетельства его существования имелись уже гораздо раньше. Но дело в том, что без концептуальной основы, возникающей благодаря знанию общего явления, такие свидетельства почти неизбежно остаются незамеченными.

Одна из ироний научного прогресса состоит и в том, что результаты, которые когда-то были настолько неожиданными, что их упускали даже при наличии множества подсказок, со временем начинают казаться почти очевидными. Прожив с результатами этой главы почти два десятилетия, я теперь с трудом представляю, что всё вообще могло бы быть устроено иначе. Но история, которую я изложил в этом разделе, как и история многих других научных открытий, служит отрезвляющим напоминанием о том, насколько легко не заметить то, что впоследствии будет казаться очевидным.

Примечания к главе 2. Решающий эксперимент

Как ведут себя простые программы?

Реализация клеточных автоматов. Состояние клеточного автомата на каждом шаге удобно представлять списком вроде {0, 0, 1, 0, 0}, где 0 соответствует белой клетке, а 1 — чёрной. Тогда начальное условие, состоящее из n белых клеток с одной чёрной клеткой посередине, можно получить с помощью следующей функции (комментарии к этой и другим функциям Mathematica приведены ниже):

```
CenterList[n_Integer] :=
  ReplacePart[Table[0, {n}], 1, Ceiling[n/2]]
```

Для клеточных автоматов того вида, который рассматривается в этой главе, правило тоже можно представить списком. Так, например, правилу 30 со страницы 27 соответствует список {0, 0, 0, 1, 1, 1, 1, 0}. (Нумерация правил обсуждается на странице 53.) В общем случае список для конкретного правила можно получить с помощью функции

```
ElementaryRule[num_Integer] := IntegerDigits[num, 2, 8]
```

Если заданы правило и список, представляющий состояние a клеточного автомата на некотором шаге, следующая простая функция даёт состояние на следующем шаге:

```
CAStep[rule_List, a_List] :=
  rule[[8 - (RotateLeft[a] + 2 (a + 2 RotateRight[a]))]]
```

Список состояний, соответствующих эволюции на протяжении t шагов, тогда можно получить так:

```
CAEvolveList[rule_, init_List, t_Integer] :=
  NestList[CAStep[rule, #] &, init, t]
```

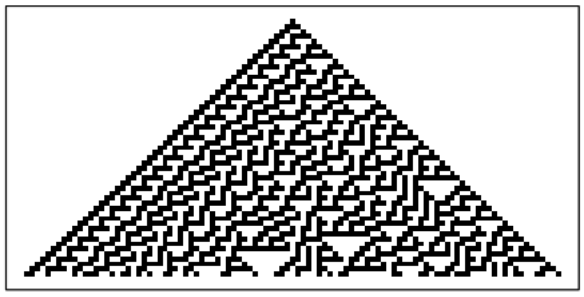
Графическое изображение этой эволюции можно построить с помощью

```
CAGraphics[history_List] := Graphics[
  Raster[1 - Reverse[history]], AspectRatio -> Automatic]
```

После задания приведённых выше определений следующий ввод в Mathematica

```
Show[CAGraphics[CAEvolveList[
  ElementaryRule[30], CenterList[103], 50]]]
```

даст показанное изображение.



Только что приведённого описания должно быть достаточно для большинства задач моделирования клеточных автоматов. В некоторых более ранних версиях Mathematica значительно более быстрый вариант программы можно создать с помощью определения

```
CAStep = Compile[{{rule, _Integer, 1}, {a, _Integer, 1}},
  rule[[8 - (RotateLeft[a] + 2 (a + 2 RotateRight[a]))]]]
```

Кроме того, в Mathematica 4 и более поздних версиях можно использовать

```
CAStep[rule_, a_] := rule[[8 - ListConvolve[{1, 2, 4}, a, 2]]]
```

либо, непосредственно через номер правила num,

```
Sign[BitAnd[2^ListConvolve[{1, 2, 4}, a, 2], num]]
```

(В версиях Mathematica, вышедших после публикации этой книги, можно использовать встроенную функцию CellularAutomaton, как обсуждается на странице 867.) Можно также организовать вызов из CAStep следующей внешней программы на языке C через MathLink — хотя обычно с каждой следующей версией Mathematica получаемое преимущество в скорости будет становиться всё менее значительным:

```
#include "mathlink.h"

main(argc, argv)
int argc; char *argv[];
{
    MLMain(argc, argv);
}

void casteps(revrule, rlen, a, n, steps)
int *revrule, rlen, *a, n, steps;
{
    int i, *ap, t, tp;

    for (i = 0; i < steps; i++)
    {
        a[0] = a[n-2]; /* правая граница */
        a[n-1] = a[1]; /* левая граница */

        t = a[0];
        for (ap = a+1; ap <= a+n-2; ap++)
        {
            tp = ap[0];
            ap[0] = revrule[ap[1]+2*(tp + 2*t)];
            t = tp;
        }
    }

    MLPutIntegerList(stdlink, a, n);
}
```

Связь этой внешней программы с функцией CAStep в Mathematica обеспечивается следующим шаблоном MathLink (обратите внимание на необязательный третий аргумент, который позволяет

CAStep выполнять сразу несколько шагов эволюции клеточного автомата):

```
:Begin:
:Function: casteps
:Pattern: CAStep[rule_List, a_List, steps_Integer:1]
:Arguments: {Reverse[rule], a, steps}
:ArgumentTypes: {IntegerList, IntegerList, Integer}
:ReturnType: Manual
:End:
```

В приведённой выше программе на C есть несколько тонких моментов. Во-первых, правила клеточных автоматов всегда определяются так, чтобы новое значение любой клетки вычислялось по старым значениям её соседей. Но поскольку программа на C явно обновляет значения последовательно слева направо, левому соседу некоторой клетки уже будет присвоено новое значение к моменту, когда мы попытаемся обновить саму эту клетку. Поэтому старое значение левого соседа необходимо сохранять во временной переменной, чтобы его можно было использовать при обновлении самой клетки. (Другой подход к этой проблеме — хранить две копии массива клеток и после каждого шага эволюции клеточного автомата менять местами указатели на них.)

Ещё один тонкий момент в программах клеточных автоматов касается граничных условий. Поскольку в реальном компьютере можно использовать лишь конечный массив клеток, необходимо решить, как применять правило клеточного автомата к клеткам на каждом из концов массива. И в программе на Mathematica, и в программе на C, приведённых выше, мы фактически используем циклический массив: левым соседом самой левой клетки считается самая правая клетка, и наоборот. В программе на C это реализовано явным копированием значения самой левой клетки в крайнюю правую позицию массива и наоборот — до обновления значений в массиве. (В некотором смысле в программе есть ошибка: при обновлении новые значения записываются только в n − 2 из n элементов массива.)

Комментарии к функциям Mathematica. Функция CenterList сначала создаёт список из n нулей, а затем заменяет средний 0 на 1. (В Mathematica 4 и более поздних версиях вместо неё можно использовать PadLeft[{1}, n, 0, Floor[n/2]].) Функция ElementaryRule преобразует num в последовательность двоичных цифр, дополняя её слева нулями до списка длины 8. Схема нумерации правил устроена так: если значение некоторой клетки равно q, значение её левого соседа равно p, а правого — r, то элемент на позиции 8 − (r + 2(q + 2p)) в списке, полученном с помощью ElementaryRule, задаёт новое значение клетки.

CAStep использует тот факт, что Mathematica может обрабатывать сразу все элементы списка. RotateLeft[a] и RotateRight[a] создают сдвинутые варианты исходного списка значений клеток a. При сложении этих списков объединяются соответствующие элементы, как в выражении

```
{p, q, r} + {s, t, u} → {p + s, q + t, r + u}.
```

В результате получается список, который для каждой клетки указывает, какой элемент правила к ней применяется. Сам список новых значений клеток затем строится с использованием того факта, что

```
{i, j, k}[[{2, 1, 1, 3, 2}]] → {j, i, i, k, j}.
```

Обратите внимание: использование RotateLeft и RotateRight автоматически даёт циклические граничные условия.

CAEvolveList применяет CAStep t раз. Многие другие функции эволюции в этих примечаниях используют тот же механизм. В общем случае

```
NestList[s[r, #] &, i, 2] → {i, s[r, i], s[r, s[r, i]]},
```

и так далее.

Побитовые оптимизации. Приведённая выше программа на C хранит значение каждой клетки в отдельном элементе целочисленного массива. Но поскольку каждое значение должно быть либо 0, либо 1, его на самом деле можно закодировать всего одним битом. А так как целочисленные переменные в реальных компьютерах обычно содержат 32 или 64 бита, значения многих клеток можно упаковать в одну целочисленную переменную. Главный смысл этого в том, что обычные машинные инструкции параллельно действуют на все биты такой переменной. Так, например, значения всех клеток, представленных целочисленной переменной a, можно параллельно обновить по правилу 30 одной инструкцией C:

```
a = a>>1 ^ (a | a<<1);
```

Однако эта инструкция обновит только конкретный блок клеток, закодированный в a. Чтобы согласовать обновления последовательности таких блоков, нужен несколько замысловатый код. (В Mathematica это реализовать гораздо легче, как обсуждалось выше, поскольку такие функции, как BitXor, могут работать с целыми числами любой длины.) В общем случае побитовые оптимизации требуют представлять правила клеточного автомата не простыми таблицами соответствий, а булевыми выражениями, которые нужно выводить для каждого правила и которые могут быть довольно сложными (см. страницу 869). Однако применение правил можно ускорить с помощью метода битовых срезов (bitslicing), позволяющего избежать операций сдвига. Идея состоит в том, чтобы хранить конфигурацию клеточного автомата, скажем, в m переменных w[i], биты которых соответствуют значениям клеток $\{a_1, a_{m+1}, a_{2m+1}, \ldots\}$, $\{a_2, a_{m+2}, a_{2m+2}, \ldots\}$, $\{a_3, \ldots\}$ и так далее. Тогда левым и правым соседями j-го бита в w[i] становятся j-е биты в w[i − 1] и w[i + 1], так что, например, шаг эволюции по правилу 30 можно выполнить просто с помощью

```
w[i] = w[i-1] ^ (w[i] | w[i+1])
```

без каких-либо операций сдвига (за исключением граничных условий для w[0] и w[m − 1]). Если требуется много шагов эволюции, достаточно упаковать все значения клеток только в начале, а распаковать их в конце.

Более общие правила. Приведённые до сих пор программы предназначены для клеточных автоматов с правилами конкретного вида, описанного в этой главе. Однако в общем случае правило одномерного клеточного автомата можно задать набором явных замен для всех

возможных блоков клеток в каждой окрестности (см. с. 60). Так, например, правило 30 можно задать в виде

```
{{1, 1, 1} -> 0, {1, 1, 0} -> 0, {1, 0, 1} -> 0, {1, 0, 0} -> 1,
{0, 1, 1} -> 1, {0, 1, 0} -> 1, {0, 0, 1} -> 1, {0, 0, 0} -> 0}
```

Чтобы использовать правила в такой форме, CAStep можно переписать следующим образом:

```
CAStep[rule_, a_List] :=
Transpose[{RotateRight[a], a, RotateLeft[a]}] /. rule
```

или

```
CAStep[rule_, a_List] := Partition[a, 3, 1, 2] /. rule
```

Теперь задаваемые правила могут содержать шаблоны; поэтому, например, правило 90 можно записать так:

```
{{1, _, 1} -> 0, {1, _, 0} -> 1, {0, _, 1} -> 1, {0, _, 0} -> 0}
```

Но как написать программу, которая могла бы работать с правилами в нескольких разных формах? Удобный подход — снабдить каждое правило «обёрткой», указывающей, в какой форме оно представлено. Тогда, например, можно определить

```
CAStep[ElementaryCARule[rule_List], a_List] :=
rule[[8 - (RotateLeft[a] + 2 (a + 2 RotateRight[a]))]]
CAStep[GeneralCARule[rule_, r_Integer:1], a_List] :=
Partition[a, 2 r + 1, 1, r + 1] /. rule
CAStep[FunctionCARule[f_, r_Integer:1], a_List] :=
Map[f, Partition[a, 2 r + 1, 1, r + 1]]
```

Заметим, что вторые два определения обобщены и допускают правила, включающие r соседей с каждой стороны. В каждом случае использование Partition можно заменить на Transpose[Table[RotateLeft[a, i], {i, -r, r}]]. Для повышения эффективности в ранних версиях Mathematica явные списки правил во втором определении можно предварительно обработать с помощью Dispatch[rules], а функции в третьем — с помощью Compile[{{x, _Integer, 1}}, body].

Реализация тоталистических клеточных автоматов обсуждается на с. 886, а клеточных автоматов большей размерности — на с. 927.

Встроенная функция клеточного автомата. Версии Mathematica, выходящие после публикации этой книги, будут включать очень общую функцию эволюции клеточных автоматов. Её описание таково; см. также с. 886.

CellularAutomaton[rnum, init, t] создаёт список, представляющий эволюцию клеточного автомата с правилом rnum из начального условия init в течение t шагов.

CellularAutomaton[rnum, init, t, {offt, offx, ...}] сохраняет только части списка эволюции с указанными смещениями.

Возможные значения rnum:
• n — k = 2, r = 1, элементарное правило;
• {n, k} — общее правило ближайших соседей с k цветами;
• {n, k, r} — общее правило с k цветами и радиусом r;
• {n, k, {r1, r2, ..., rd}} — d-мерное правило с окрестностью (2 r1 + 1) × (2 r2 + 1) × ... × (2 rd + 1);
• {n, k, {{off1}, {off2}, ..., {offs}}} — правило с соседями по заданным смещениям;
• {n, {k, 1}} — тоталистическое правило ближайших соседей с k цветами;
• {n, {k, 1}, r} — тоталистическое правило радиуса r с k цветами;
• {n, {k, {wt1, wt2, ...}}, rspec} — правило, в котором соседу i присвоен вес wti;

• {fun, {}, rspec} — применение функции fun к каждому списку соседей со вторым аргументом, равным номеру шага.

CellularAutomaton[{n, k}, ...] эквивалентно CellularAutomaton[{n, {k, {k^2, k, 1}}}, ...]. Распространённые формы для двумерных клеточных автоматов:
• {n, {k, 1}, {1, 1}} — тоталистическое правило с 9 соседями;
• {n, {k, {{0, 1, 0}, {1, 1, 1}, {0, 1, 0}}, {1, 1}}} — тоталистическое правило с 5 соседями;
• {n, {k, {{0, k, 0}, {k, 1, k}, {0, k, 0}}, {1, 1}}} — внешнетоталистическое правило с 5 соседями;
• {n + k^5 (k - 1), {k, {{0, 1, 0}, {1, 4 k + 1, 1}, {0, 1, 0}}, {1, 1}}} — правило роста с 5 соседями.

Обычно все элементы init и списка эволюции — целые числа от 0 до k - 1. Но при использовании функции общего вида элементы init и списка эволюции не обязаны быть целыми числами. Второй аргумент, передаваемый fun, — номер шага, начиная с 0. Начальные условия строятся из init следующим образом:
• {a1, a2, ...} — явный список значений ai, считающийся циклическим;
• {{a1, a2, ...}, b} — значения ai, наложенные на фон b;
• {{a1, a2, ...}, {b1, b2, ...}} — значения ai, наложенные на фон повторений b1, b2, ...;
• {{{{a11, a12, ...}, off1}, {a21, ...}, off2}, ...}, bspec} — значения aij со смещениями offi на фоне;
• {{a11, a12, ...}, {a21, ...}, ...} — явный список значений в двух измерениях;
• {aspec, bspec} — значения в d измерениях с d-мерным заполнением фона.

Примечание к переводу: в строке init со смещениями off1 и off2 сохранено расположение скобок оригинала. В записи второй пары, по-видимому, имеется типографская опечатка; она здесь не исправлена.

Первый элемент aspec накладывается на фон в первой позиции положительного направления по каждой координате относительно начала координат. Это означает, что bspec[[1, 1, ...]] совмещается с aspec[[1, 1, ...]]. Временные смещения offt задаются так:
• All — все шаги от 0 до t, по умолчанию;
• u — шаги от 0 до u;
• -1 — последний шаг, то есть шаг t;
• {u} — шаг u;
• {u1, u2} — шаги от u1 до u2;
• {u1, u2, du} — шаги u1, u1 + du, ...

CellularAutomaton[rnum, init, t] создаёт список эволюции длины t + 1. Начальному условию приписывается смещение 0. Пространственные смещения offx задаются следующим образом:
• All — все клетки, на которые может повлиять заданное начальное условие;
• Automatic — все клетки в области, отличающейся от фона;
• 0 — клетка, совмещённая с началом aspec;
• x — клетки со смещениями до x вправо;
• -x — клетки со смещениями до x влево;
• {x} — клетка со смещением x вправо;
• {-x} — клетка со смещением x влево;
• {x1, x2} — клетки со смещениями от x1 до x2;
• {x1, x2, dx} — клетки x1, x1 + dx, ...

В одном измерении по умолчанию считается, что первый элемент aspec имеет пространственное смещение 0. В любом числе измерений по умолчанию считается, что aspec[[1, 1, 1, ...]] имеет пространственное смещение {0, 0, 0, ...}. Все элементы списка эволюции, создаваемого CellularAutomaton, всегда имеют один размер. При начальном условии, задаваемом aspec ширины w, область, на которую клеточный автомат с

правилом радиуса r может повлиять через t шагов, имеет ширину w + 2 r t. Если фон bspec не задан, пространственные смещения All и Automatic включают каждую клетку aspec. Пространственное смещение All включает все клетки, на которые может повлиять начальное условие. Смещение Automatic позволяет обрезать фон по бокам узора клеточного автомата. Определяя, какую ширину области сохранить, Automatic рассматривает результаты только на шагах, заданных offt.

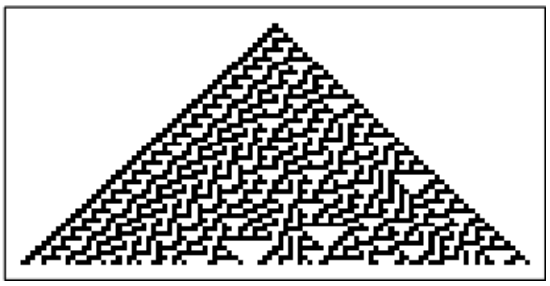
Вот несколько примеров.

Здесь получается массив значений, возникающий при выполнении правила 30 в течение 3 шагов из начального условия, состоящего из единственной единицы, окружённой нулями.

```
In[1]:= CellularAutomaton[30, {{1}, 0}, 3]
Out[1]= {{0, 0, 0, 1, 0, 0, 0}, {0, 0, 1, 1, 1, 0, 0}, {0, 1, 1, 0, 0, 1, 0}, {1, 1, 0, 1, 1, 1, 1}}
```

Здесь правило 30 выполняется в течение 50 шагов, и строится изображение результата.

```
In[2]:= Show[RasterGraphics[CellularAutomaton[30, {{1}, 0}, 50]]]
```

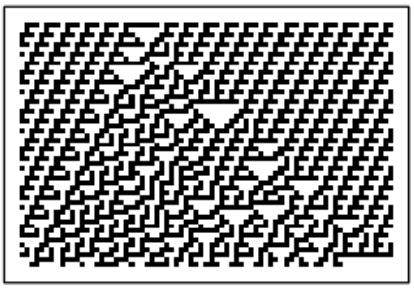


Если все значения начального условия заданы явно, фактически предполагается их циклическое продолжение. Здесь правило 30 выполняется на 5 клетках в течение 3 шагов.

```
In[3]:= CellularAutomaton[30, {1, 0, 0, 1, 0}, 3]
Out[3]= {{1, 0, 0, 1, 0}, {1, 1, 1, 1, 0}, {1, 0, 0, 0, 0}, {1, 1, 0, 0, 1}}
```

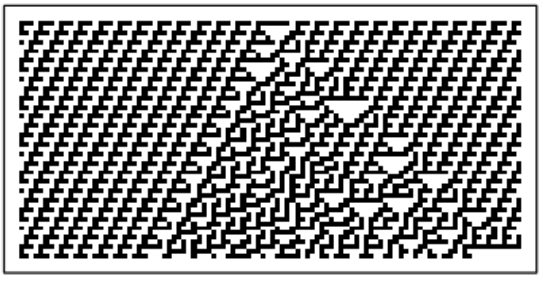
Здесь начальные {1, 1} помещаются на бесконечный фон из повторяющихся блоков {1, 0, 1, 1}. По умолчанию выводится только область узора, затронутая {1, 1}.

```
In[4]:= Show[RasterGraphics[CellularAutomaton[30, {{1, 1}, {1, 0, 1, 1}}, 50]]]
```



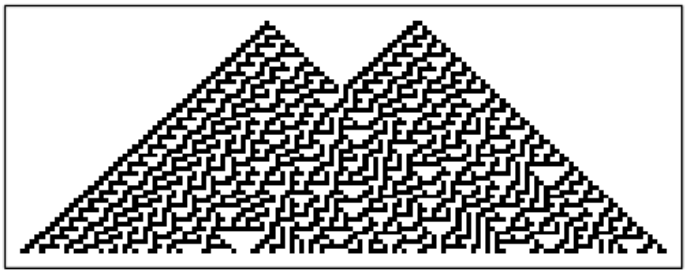
Здесь выдаются все клетки, которые вообще могли быть затронуты, независимо от того, затронуты ли они на самом деле.

```
In[5]:= Show[RasterGraphics[CellularAutomaton[30,
{{1, 1}, {1, 0, 1, 1}}, 50, {All, All}]]]
```



Здесь блоки начального условия помещаются со смещениями -10 и 20.

```
In[6]:= Show[RasterGraphics[CellularAutomaton[30,
{{{{1}, {-10}}, {{1, 1}, {20}}}, 0}, 50]]]
```



Здесь выдаётся только последняя строка после выполнения 10 шагов.

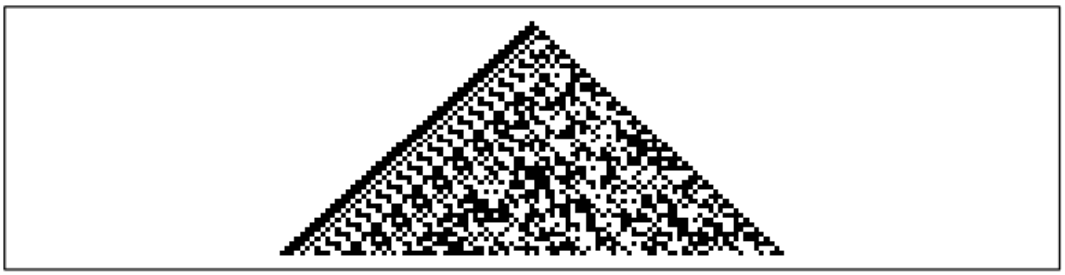
```
In[7]:= CellularAutomaton[30, {{1}, 0}, 10, -1]
Out[7]= {{1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0}}
```

Здесь выполняются 5 шагов, и на каждом шаге выдаются клетки трёх центральных столбцов.

```
In[8]:= CellularAutomaton[30, {{1}, 0}, 5, {All, {-1, 1}}]
Out[8]= {{0, 1, 0}, {1, 1, 1}, {1, 0, 0}, {0, 1, 1}, {0, 1, 0}, {1, 1, 1}}
```

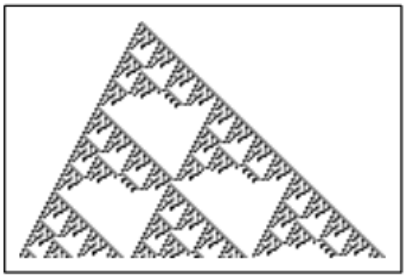
Здесь выбирается каждая вторая клетка в пространстве и времени, начиная с позиции на 200 клеток левее.

```
In[9]:= Show[RasterGraphics[CellularAutomaton[30, {{1}, 0}, 100,
{{1, 100, 2}, {-200, 200, 2}}]]]
```



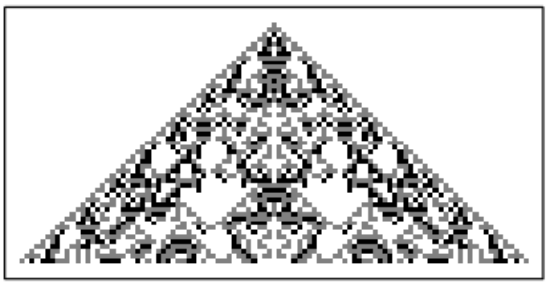
Здесь выполняется общее правило с k = 3, r = 1 и номером 921408.

```
In[10]:= Show[RasterGraphics[CellularAutomaton[{921408, 3, 1}, {{1}, 0}, 100]]]
```



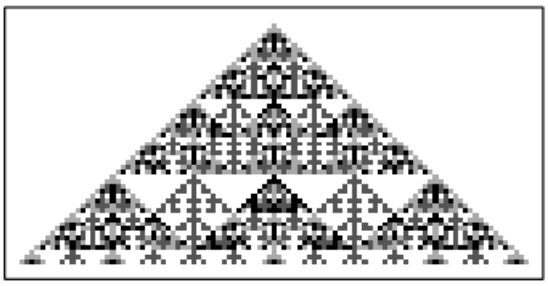
Здесь выполняется тоталистическое правило с k = 3, r = 1 и кодом 867.

```
In[11]:= Show[RasterGraphics[CellularAutomaton[{867, {3, 1}, 1}, {{1}, 0}, 50]]]
```



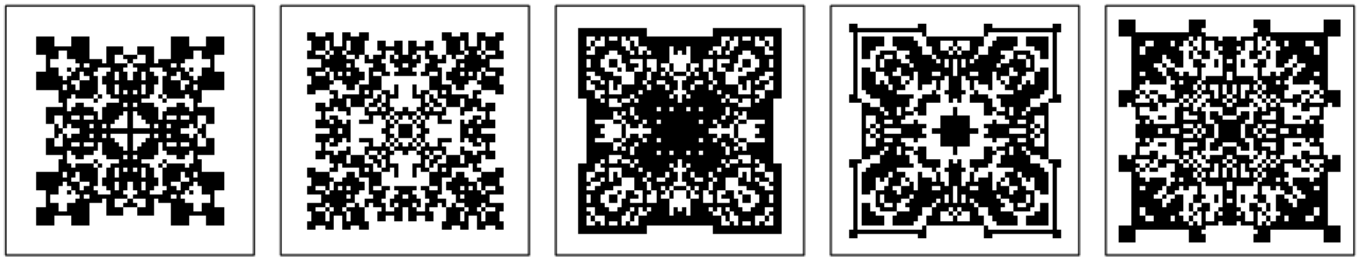
Здесь используется правило, основанное на применении функции к каждой окрестности клеток.

```
In[12]:= Show[RasterGraphics[CellularAutomaton[
{Mod[Apply[Plus, #], 4] &, {}, 1}, {{1}, 0}, 50]]]
```



Здесь двумерное тоталистическое правило с 9 соседями и кодом 3702 выполняется в течение 25 шагов; выдаются результаты последних 5 шагов.

```
In[13]:= Show[GraphicsArray[Map[RasterGraphics,
CellularAutomaton[{3702, {2, 1}, {1, 1}}, {{{1}}, 0}, 25, -5]]]]
```



Специализированная аппаратура. Простая структура клеточных автоматов естественно наводит на мысль об их реализации на специализированной аппаратуре. И действительно, начиная с 1950-х годов был построен ряд специализированных машин для реализации одномерных, двумерных, а иногда и трёхмерных клеточных автоматов. Использовались две основные идеи: параллелизм и конвейерная обработка. Обе опираются на локальный характер правил клеточных автоматов.

При параллельном подходе машина содержит много отдельных процессоров, каждый из которых обслуживает одну клетку или небольшую группу клеток. При конвейерном подходе имеется лишь один процессор, либо несколько, через который последовательно пропускаются данные о разных клетках. Ключевой момент, однако, в том, что на каждом этапе легко определить, какие данные понадобятся, поэтому их можно выбирать заранее, потенциально через специально построенную систему памяти.

В целом достижимое ускорение зависит от многих деталей архитектуры памяти и обмена данными. С годами выигрыш обычно становился менее значительным: встроенная память микропроцессоров росла, а время пересылки данных между микросхемами приобретало относительно большее значение.

В будущем, однако, новые технологии могут изменить соотношение преимуществ и затрат. Клеточные автоматы — очевидные кандидаты на раннюю реализацию и в нанотехнологии, и в оптических вычислениях. См. также с. 841.

Звуковое представление. Шаг эволюции клеточного автомата можно представить звуком, если считать каждую клетку клавишей фортепиано, которая нажата, когда клетка чёрная. Получится аккорд, например такой:

```
Play[Evaluate[Apply[Plus, Flatten[Map[Sin[1000 # t] &,
N[2^(1/12)]^Position[list, 1]]]]], {t, 0, 0.2}]
```

Последовательность таких аккордов иногда может служить полезным представлением эволюции клеточного автомата. См. также с. 1080.

Правила клеточных автоматов как формулы. Значение a[t, i] клетки на шаге t в позиции i для любого клеточного автомата этой главы можно получить из определения

```
a[t_, i_] := f[a[t - 1, i - 1], a[t - 1, i], a[t - 1, i + 1]]
```

Разным правилам соответствуют разные варианты функции f. Например, правилу 90 на с. 25 соответствует

```
f[1, _, 1] = 0; f[0, _, 1] = 1; f[1, _, 0] = 1; f[0, _, 0] = 0
```

Начальные условия можно задать, например, так:

```
a[0, 0] = 1; a[0, _] = 0
```

Клетка на шаге 0 в позиции 0 имеет значение 1, а все остальные клетки на этом шаге — значение 0. Тогда, просто запросив a[4, 0], мы сразу получим значение клетки в позиции 0 через 4 шага. На практике для эффективности основное определение следует записывать так:

```
a[t_, i_] := a[t, i] = f[a[t - 1, i - 1], a[t - 1, i], a[t - 1, i + 1]]
```

Тогда все вычисленные промежуточные значения автоматически сохраняются.

Приведённое выше определение функции f для правила 90 — по существу, просто таблица соответствий. Но эту функцию можно определить и алгебраически:

```
f[p_, q_, r_] := Mod[p + r, 2]
```

Алгебраические определения можно дать и для других правил:
• Правило 254 (с. 24): 1 - (1 - p) (1 - q) (1 - r).
• Правило 250 (с. 25): p + r - p r.
• Правило 30 (с. 27): Mod[p + q + r + q r, 2].
• Правило 110 (с. 32): Mod[(1 + p) q r + q + r, 2].

В этих определениях значения клеток представляются числами 1 или 0. Если вместо этого использовать +1 и -1, получатся другие формулы: например, правилу 90 будет соответствовать p r. Можно также представлять значения клеток как True и False. В этом случае правила клеточных автоматов превращаются в логические выражения:
• Правило 254: Or[p, q, r].
• Правило 250: Or[p, r].
• Правило 90: Xor[p, r].

• Правило 30: Xor[p, Or[q, r]].
• Правило 110: Xor[Or[p, q], And[p, q, r]].

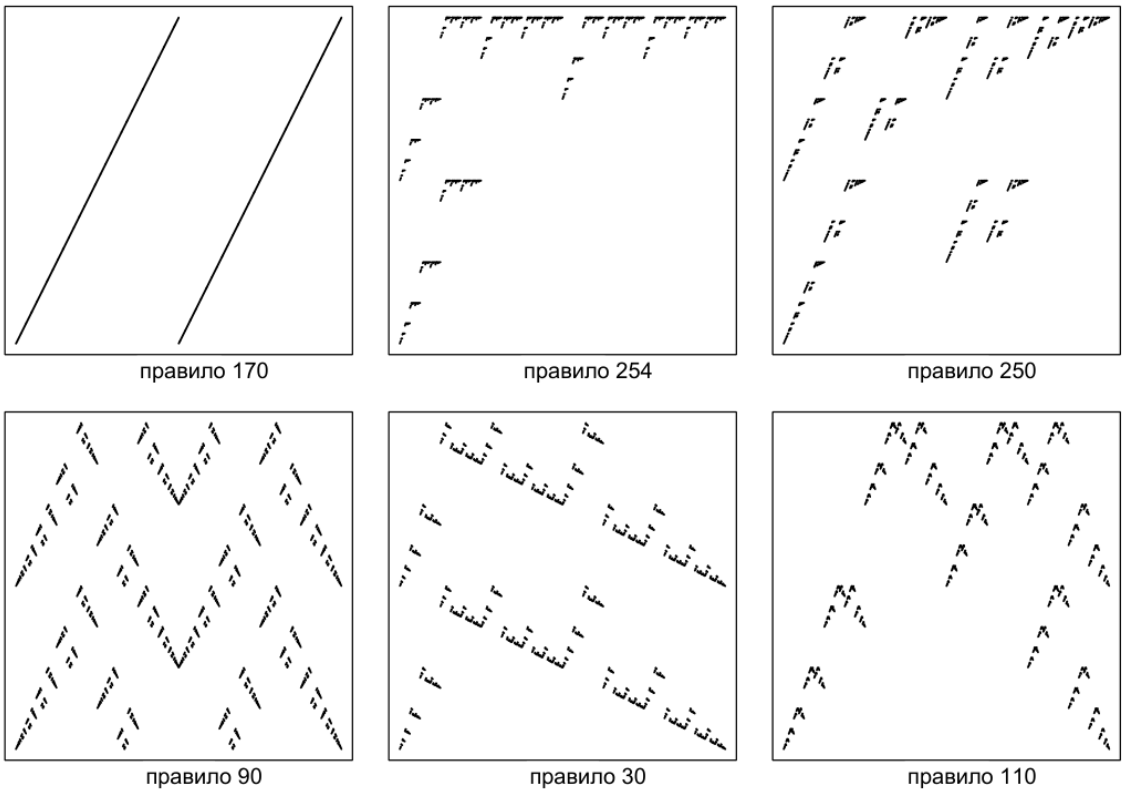
Заметим, что Not[p] соответствует 1 - p; And[p, q] — p q; Xor[p, q] — Mod[p + q, 2]; Or[p, q] — Mod[p q + p + q, 2].

Имея алгебраическую либо логическую форму правила клеточного автомата, можно, по крайней мере в принципе, получать символьные формулы для результатов его эволюции. Так, например, можно использовать начальные условия

```
a[0, -1] = p; a[0, 0] = q; a[0, 1] = r; a[0, _] = 0
```

и получить формулу значения клетки, справедливую при любом выборе значений трёх исходных центральных клеток. Однако на практике большинство таких формул быстро становится очень сложными, как обсуждается на с. 618.

Математическая интерпретация клеточных автоматов. В контексте чистой математики пространство состояний одномерного клеточного автомата с бесконечным числом клеток можно рассматривать как канторово множество. Тогда правило клеточного автомата соответствует непрерывному отображению этого канторова множества в себя; непрерывность следует из локальности правила. Ср. с. 959.



Верхний ряд: правило 170; правило 254; правило 250. Нижний ряд: правило 90; правило 30; правило 110.

Изображения выше представляют отображения, соответствующие различным правилам. Они получены построением графика Sum[a[t + 1, i] 2^(-i), {i, -n, n}] в зависимости от Sum[a[t, i] 2^(-i), {i, -n, n}]

для всех возможных вариантов a[t, i]. Используются периодические граничные условия, так что a[t, i] можно считать в точности соответствующими цифрам рациональных чисел. Правило 170 — классическое отображение сдвига: оно сдвигает значения всех клеток на одну позицию влево, не меняя их. На рисунках ниже это отображение имеет вид Mod[2 x, 1]; ср. с. 153.

С. 26 · Треугольник Паскаля и правило 90. Как показано на с. 611, узор, получаемый по правилу 90, в точности является треугольником Паскаля из биномиальных коэффициентов, взятых по модулю 2: чёрным клеткам соответствуют нечётные биномиальные коэффициенты.

Количество чёрных клеток в строке t равно 2^DigitCount[t, 2, 1]; график DigitCount[t, 2, 1] приведён на с. 902. Позиции чёрных клеток задаются следующим выражением, что устанавливает связь с изображением на с. 117:

```
Fold[Flatten[{#1 - #2, #1 + #2}] &, 0, 2^DigitPositions[t]]
DigitPositions[n_] :=
Flatten[Position[Reverse[IntegerDigits[n, 2]], 1]] - 1
```

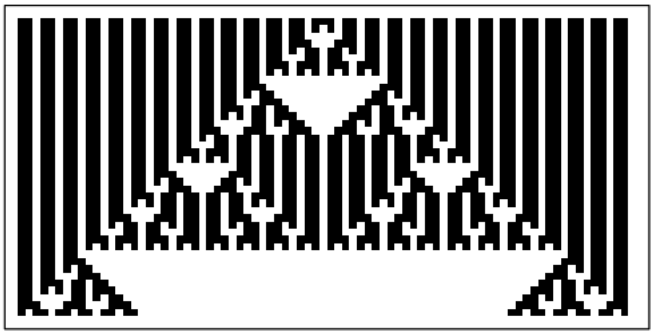
Сам узор, порождаемый правилом 90, соответствует коэффициентам в PolynomialMod[Expand[(1/x + x)^t], 2], см. с. 1091; поэтому цвет конкретной клетки задаётся выражением Mod[Binomial[t, (n + t)/2], 2] /; EvenQ[n + t].

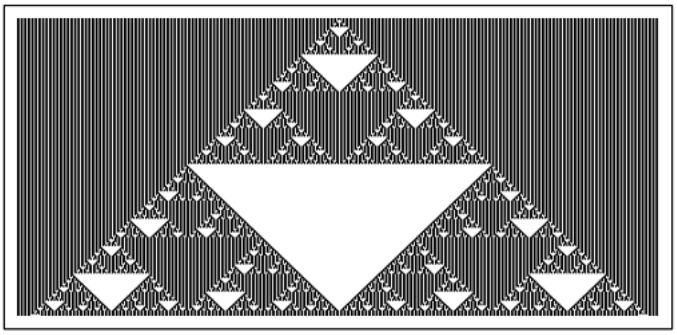
Mod[Binomial[t, n], 2] даёт искажённый узор, порождаемый правилом 60, см. с. 58. В этом узоре цвет конкретной клетки можно получить непосредственно из последовательностей цифр t и n с помощью 1 - Sign[BitAnd[-t, n]] или, см. с. 583,

```
With[{d = Ceiling[Log[2, Max[t, n] + 1]]}, If[FreeQ[
IntegerDigits[t, 2, d] - IntegerDigits[n, 2, d], -1], 1, 0]]
```

Самоподобие. Узор, порождаемый правилом 90 за заданное число шагов, имеет следующее свойство: он совпадает с тем, что получилось бы при выполнении вдвое большего числа шагов с последующим сохранением лишь каждой второй строки и каждого второго столбца. Через 2^m шагов очерченная узором треугольная область содержит в общей сложности 4^m клеток, но только 3^m из них чёрные. В пределе бесконечного числа шагов получается фрактал, известный как узор Серпинского (см. с. 934), с фрактальной размерностью Log[2, 3] ≈ 1,59, см. с. 933. Вложенность возникает во всех клеточных автоматах с аддитивными правилами, см. с. 955.

Другое начальное условие. Вставка одиночной чёрной клетки в фон из повторяющихся чёрно-белых блоков для правила 90 даёт показанный ниже узор, в котором и белые, и полосатые области имеют фрактальную размерность 2.





Больше цветов. Изображения ниже показывают обобщения правила 90 на k возможных цветов с использованием правила

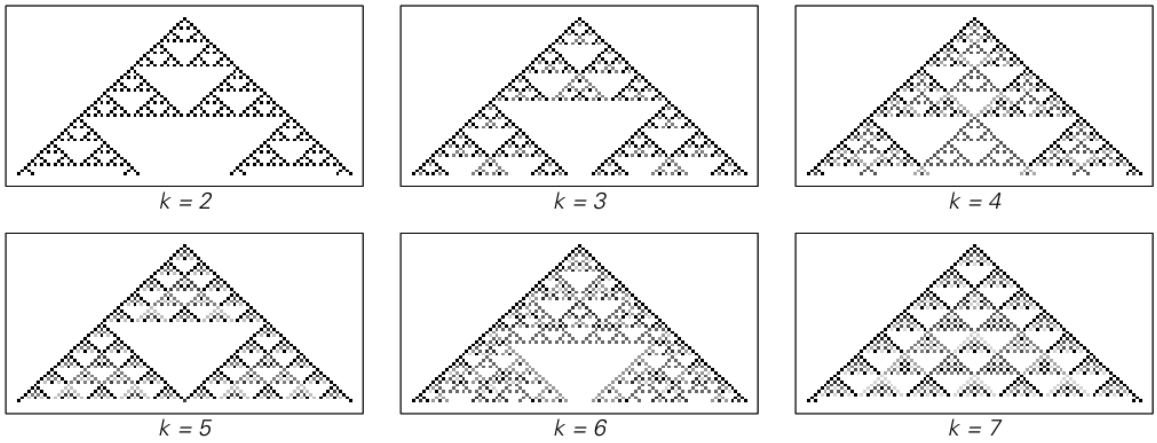
```
CAStep[k_Integer, a_List] :=
Mod[RotateLeft[a] + RotateRight[a], k]
```

или эквивалентно Mod[ListCorrelate[{1, 0, 1}, a, 2], k]. Количество небелых клеток в строке t в этом случае задаётся выражением Apply[Times, 1 + IntegerDigits[t, k]]. Для непростых k узоры получаются

наложением узоров, соответствующих множителям k. Связанный результат: IntegerExponent[Binomial[t, n], k] равно количеству заимствований при вычитании n из t в системе счисления с основанием k. Для простого k значение Mod[Binomial[t, n], k] задаётся так:

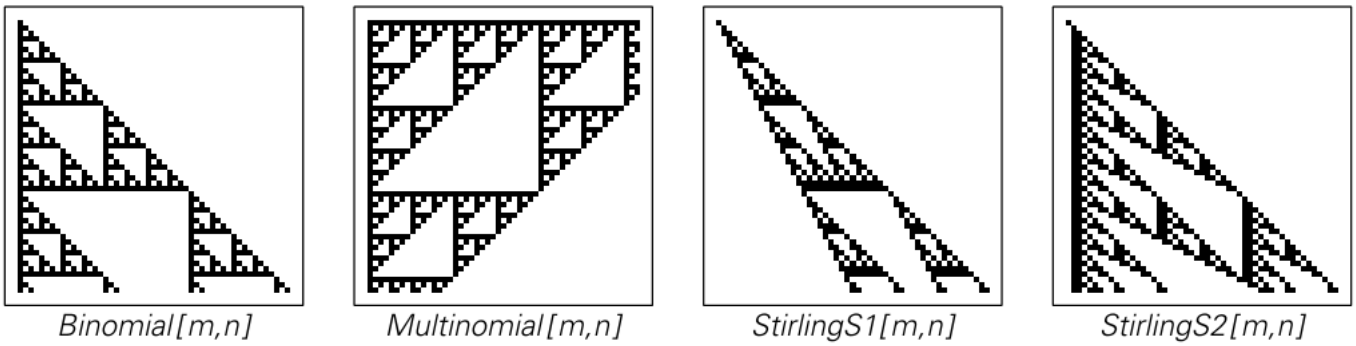
```
With[{d = Ceiling[Log[k, Max[t, n] + 1]]},
Mod[Apply[Times, Apply[Binomial, Transpose[
{IntegerDigits[t, k, d], IntegerDigits[n, k, d]}], {1}]], k]]
```

Узоры, получаемые при любом k, являются вложенными. Для простого k общее число небелых клеток до шага k^m равно ((1/2) k (k + 1))^m, а фрактальная размерность узоров равна 1 + Log[k, (k + 1)/2], см. с. 955. Это примеры аддитивных правил, подробнее обсуждаемых на с. 952. См. также с. 922 о непрерывном случае.



k = 2; k = 3; k = 4; k = 5; k = 6; k = 7.

История. Треугольник Паскаля, вероятно, восходит к античности; он был известен в Китае в XIII веке и довольно подробно обсуждался Блезом Паскалем в 1654 году, в частности в связи с теорией вероятностей. Подход к нахождению биномиальных коэффициентов по модулю k на основе цифр независимо изобретался много раз с середины XIX века, в частности Эдуаром Люка в 1877 году и Джеймсом Глейшером в 1899 году. Тот факт, что нечётные биномиальные коэффициенты образуют вложенный геометрический узор, по-видимому, не был широко замечен, пока я не подчеркнул это в 1982 году.

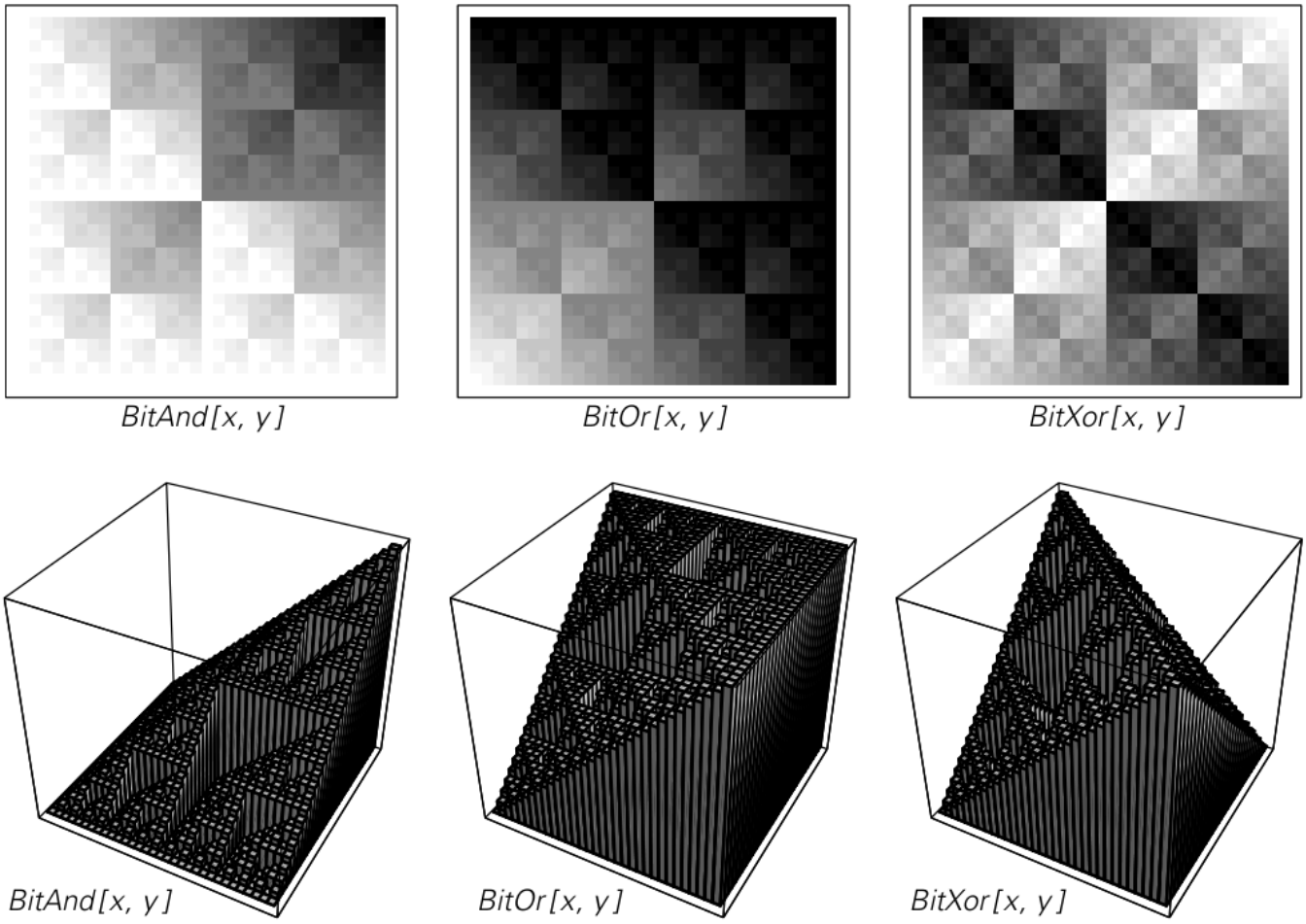


Binomial[m, n]; Multinomial[m, n]; StirlingS1[m, n]; StirlingS2[m, n].

Другие целочисленные функции. Изображения выше показывают узоры, получаемые взятием нескольких целочисленных функций по модулю 2. С d аргументами Multinomial даёт вложенный узор в d измерениях. Заметим, что GCD[m, n] даёт более сложный

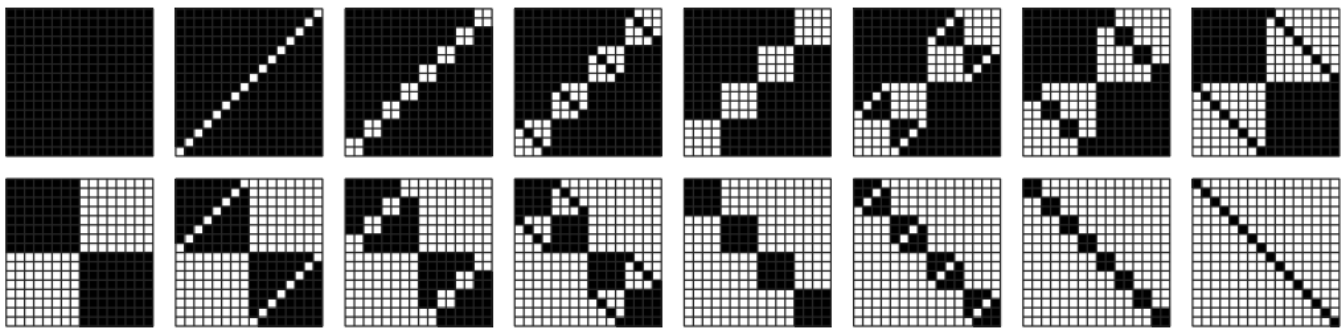
узор (см. с. 613), как и JacobiSymbol[m, 2 n - 1] (см. с. 1081) и разные сочетания функций (см. с. 747).

Побитовые функции. Побитовые функции обычно дают вложенные узоры. Как обсуждалось выше, любое правило клеточного автомата можно представить как подходящее сочетание побитовых функций. Заметим, что BitOr[x, y] + BitAnd[x, y] == x + y и BitOr[x, y] - BitAnd[x, y] == BitXor[x, y].

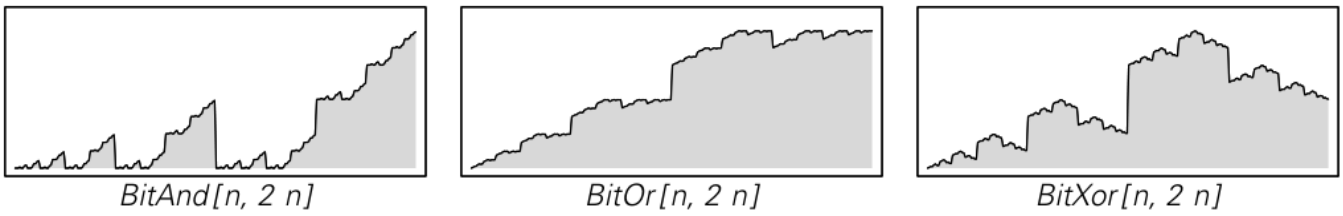


Изображения и поверхности: BitAnd[x, y]; BitOr[x, y]; BitXor[x, y].

Узоры ниже показывают, где BitXor[x, y] == t для последовательных t; они соответствуют шагам программы «munching squares», исследовавшейся на компьютере PDP-1 в 1962 году.



Вложенность наблюдается также в кривых, полученных применением побитовых функций к n и 2 n для последовательных n. Заметим, что число 2 n содержит те же цифры, что и n, но сдвинутые на одну позицию влево.



BitAnd[n, 2 n]; BitOr[n, 2 n]; BitXor[n, 2 n].

С. 28 · Проверки случайности. Выполненные мной статистические проверки включают восемь проверок, перечисленных на с. 1084.

С. 29 · Правило 30. Левая сторона показанного узора явно носит повторяющийся характер. Вообще, если смотреть вдоль диагонали на глубине n клеток от любого края узора, период повторения может составлять не более 2^n. На правом краю первые наблюдаемые периоды таковы: {1, 2, 2, 4, 8, 8, 16, 32, 32, 64, 64, 64, 64, 64, 128, 256}; в целом период, по-видимому, растёт с глубиной экспоненциально. На левом краю период растёт чрезвычайно медленно: период 2 впервые достигается на глубине 3, период 4 — на глубине 8, 8 — на 29, 16 — на 400, 32 — на 87 867, 64 — на 2 107 985 255 или больше, и так далее. Оказывается, каждое удвоение периода происходит в точности тогда, когда диагональ узора в конце концов становится белой полосой, а диагональ слева от неё содержит нечётное число чёрных клеток в каждом повторяющемся блоке. Граница, отделяющая повторение слева от случайности справа, на каждом шаге сдвигается в среднем приблизительно на 0,252 клетки влево; ср. с. 949. Изображение ниже показывает колебания относительно этого среднего.

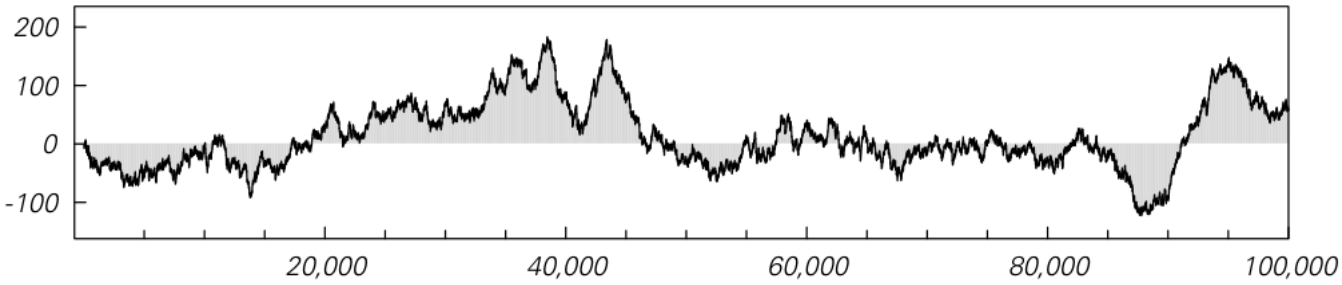
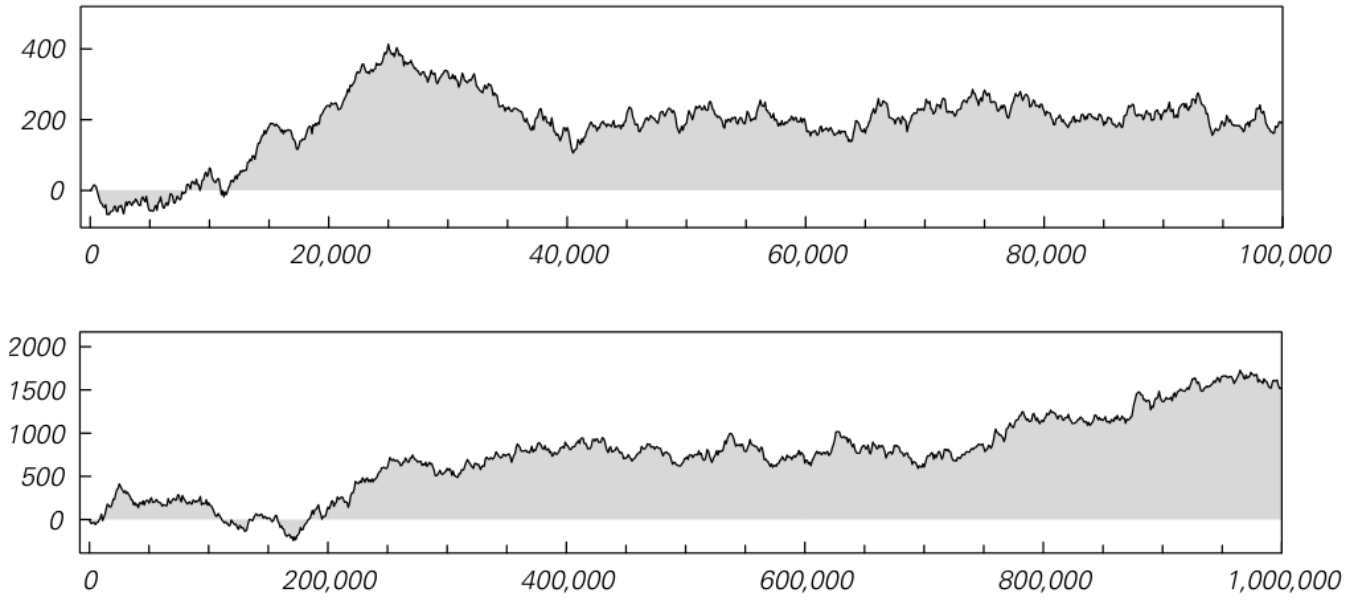


График колебаний: шаги 20 000, 40 000, 60 000, 80 000, 100 000; отклонения -100, 0, 100, 200.

Полный узор. По-видимому, со временем встречаются все возможные блоки, см. с. 725. Вероятность блока из n соседних белых клеток, соответствующего строке белого треугольника, по-видимому, довольно точно приближается к 2^(-n); первый такой блок длины 10 появляется на шаге 67, а первый блок длины 20 — на шаге 515.

Центральный столбец. Изображения ниже показывают превышение числа чёрных клеток над числом белых в центральном столбце. Из первых 100 000 клеток в общей сложности 50 098 чёрные, а из первого миллиона — 500 768. Самая длинная серия одинаковых цветов среди первых 100 000 клеток состоит из 21 чёрной клетки, а среди первого миллиона элементов — из 22 чёрных клеток. Первые n элементов можно эффективно найти так:

```
Module[{a = 1}, Table[First[IntegerDigits[
a, a = BitXor[a, BitOr[2 a, 4 a]]; 2, i]], {i, n}]]
```



Верхний график: шаги 0, 20 000, 40 000, 60 000, 80 000, 100 000; превышение 0, 200, 400. Нижний график: шаги 0, 200 000, 400 000, 600 000, 800 000, 1 000 000; превышение 0, 500, 1000, 1500, 2000.

Последовательность не повторяется по крайней мере на первом миллионе шагов. Я был бы поражён, если бы она когда-нибудь повторилась, но пока не знаю строгого доказательства этого. В 1986 году Эрика Джен показала, что никакая пара столбцов не может повторяться; а рассуждения на с. 1087 указывают, что не может повторяться и центральный столбец вместе с отдельными соседними клетками.

С. 32 · Правило 110. Значительно подробнее правило 110 обсуждается на с. 229 и 675. Возникающие в нём локализованные структуры показаны на с. 292. Заметим, что из восьми случаев

В основном правиле для правила 110 лишь один из восьми случаев отличается от правила 102 — простого аддитивного правила, получаемого отражением правила 60.

Необходимость новой интуиции

Реакции учёных. Сложность изображений в этой главе настолько удивляет многих учёных, что поначалу они считают её ненастоящей. Обычно они полагают, что, хотя изображения и выглядят сложными, на деле они оказались бы простыми, если бы их подвергли подходящему анализу. В главе 10 я приведу обширные свидетельства того, что это не так. Здесь же достаточно сказать: в поиске закономерностей даже самые передовые методы математики и статистики обычно оказываются не сильнее наших глаз. И какое бы формальное определение сложности мы ни использовали (см. с. 557), уже сам факт того, что наши глаза замечают её в системах, рассмотренных в этой главе, чрезвычайно существенен.

Интуиция из практической работы с компьютерами. Повседневный опыт работы с компьютерами и программирования приводит к следующим наблюдениям:
• Можно создавать универсальные компьютеры и универсальные языки программирования.
• Можно составлять разные программы для выполнения самых разнообразных задач.
• Каждую конкретную программу можно реализовать множеством способов.
• Программы могут вести себя сложно и, по-видимому, случайно — особенно когда работают неправильно.
• Отладка программы может быть трудной.
• Читая код программы, часто бывает трудно предвидеть, что она способна делать.
• Чем ниже уровень представления программного кода, тем труднее, как правило, его понять.
• Некоторые вычислительные задачи легко сформулировать, но трудно решить.
• Программы, моделирующие природные системы, относятся к самым затратным по объёму вычислений.
• Люди способны создавать большие программы — по крайней мере по частям.
• Почти всегда программу можно оптимизировать ещё сильнее, но оптимизированную версию может быть труднее понять.
• Более короткие программы иногда эффективнее, но оптимизация часто требует отдельно обрабатывать много случаев, делая программы длиннее.
• Если в программу вносят слишком много заплаток, она обычно вообще перестаёт работать.

Применение в дизайне. Многие изображения в этой книге поразительно похожи на художественные орнаменты разных стилей. Вероятно, это отражает не столько сходство лежащих в их основе правил, сколько сходство признаков, наиболее заметных человеческому зрению. Заметим, что квадратные сетки цветных клеток, подобные клеточным автоматам в этой главе, можно совершенно непосредственно использовать как схемы ткачества. См. также с. 929.

Почему эти открытия не были сделаны раньше

С. 43 · Орнаментальное искусство. Почти со всеми основными культурными эпохами связаны определённые характерные формы орнамента. Вероятно, орнаменты на предметах определённого рода часто возникали как идеализированные подражания более ранним или более естественным формам этих предметов: так, распространены имитации плетения, кирпичной кладки и разнообразных растительных форм. Крупномасштабные, чисто абстрактные узоры занимали центральное место и в таких культурных традициях, как исламская, где природные формы считались творениями Бога, которые нельзя изображать непосредственно. Сложившиеся стили орнамента обычно многократно воспроизводят, создавая ощущение уюта и привычности, особенно в архитектуре. По-видимому, подавляющее большинство сложных орнаментов создавалось ремесленниками почти или совсем без формального теоретического обсуждения. Однако, особенно с XIX века, предпринимались различные попытки найти систематические способы классификации орнаментальных форм, иногда на основе аналогий с грамматикой. Вопросы пропорций, впрочем,

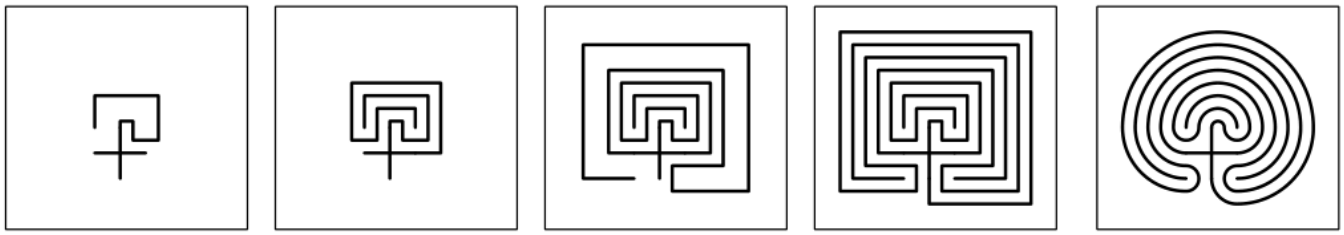
издавна были предметом обстоятельных теоретических обсуждений. Примечательно, что, хотя повторяющиеся узоры в орнаментах используются очень широко, даже вложенность встречается довольно редко. И хотя, например, разрабатывались сложные правила симметрии, ничего похожего на правила клеточных автоматов, по-видимому, никогда не возникало. Результаты этой книги теперь показывают, что такие правила способны передавать сущность многих сложных процессов в природе. Поэтому, даже не имея исторического контекста, они потенциально могут стать основой орнаментальных форм, воспринимаемых как знакомые идеализации природы. Ср. с. 929.

Изображения в основном тексте показывают последовательность ранних примеров разных характерных форм орнамента.

22 000 год до н. э. (палеолит). Браслет из бивня мамонта из Мезина, Украина. Подобные зигзагообразные рисунки встречаются и на других предметах того же периода. В показанном примере примечательно, что угол зигзагов сопоставим с углом линий Шрегера, естественно возникающих в дентине мамонта.

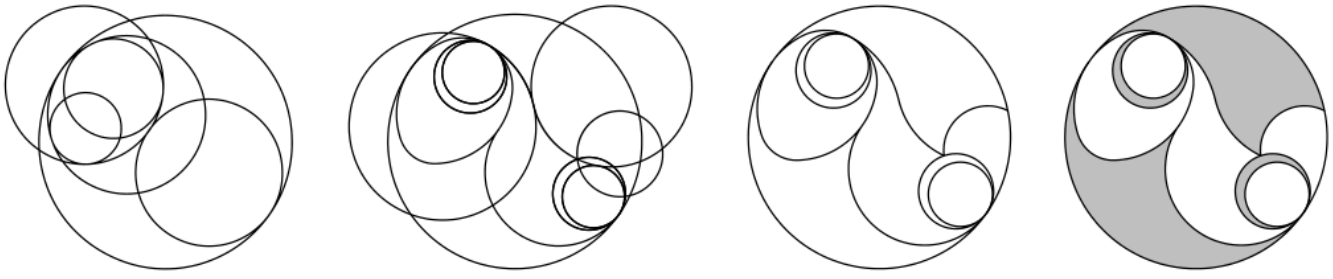
3000 год до н. э. (Шумер). Колонны с глиняными конусами трёх цветов, вставленными в глину стены храма Эанна в Уруке, Месопотамия (Варка, Ирак), — возможно, упомянутого в «Эпосе о Гильгамеше». Сейчас находятся в Государственном музее, Берлин. Это самый ранний известный явный пример мозаики.

1200 год до н. э. (Греция). Оборотная сторона глиняной учётной таблички из Пилоса, Греция. Предположительно, узор был построен способом, показанным ниже. Согласно преданию, это был план лабиринта, в котором обитал Минотавр, во дворце в Кноссе на Крите; его создателем считался Дедал. Говорят также, что это была эмблема Трои — или, возможно, план некоторых её стен. Этот узор — квадратный или округлый — с удивительно небольшими изменениями встречался в самых разных местах по всему миру: на критских монетах, в граффити Помпей, на полу Шартрского собора, в резьбе в Перу, в эмблемах коренных племён. На протяжении, вероятно, трёх тысяч лет он оставался самым распространённым рисунком лабиринта.

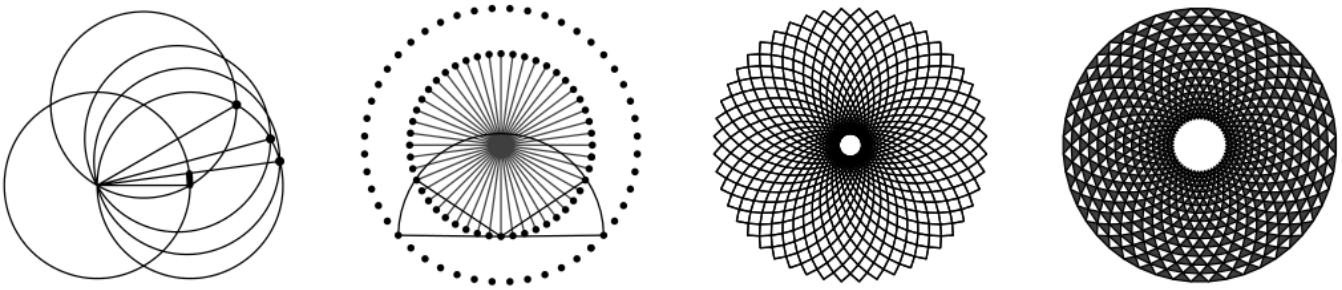


900 год до н. э. (Финикия). Резьба по слоновой кости, предположительно из Средиземноморья. Сейчас находится в Британском музее. Этот распространённый декоративный узор получали, рисуя окружности с центрами в отверстиях, расположенных треугольной сеткой. Он встречается также в египетском и другом искусстве. Подобные узоры рассматривали Евклид, а позднее Леонардо да Винчи в связи с теорией луночек.

I век до н. э. (кельты). Оборотная сторона так называемого зеркала из Десборо — бронзового зеркала из Десборо, Англия, изготовленного в железном веке между 50 годом до н. э. и 50 годом н. э. Сейчас находится в Британском музее. Гравированный узор состоит из частей окружностей, лишь касающихся друг друга, как на рисунке ниже.



II век н. э. (Рим). Мозаика из архитектурного комплекса в Риме, Италия. Сейчас находится в Национальном музее, Рим. Предположительно, этот геометрический узор создавался так: сначала повторным делением углов пополам строили 48 равноотстоящих лучей, как на первом рисунке ниже; затем проводили полуокружности с центрами на концах каждого луча и, наконец, добавляли концентрические окружности, проходящие через точки пересечения. Подобные розеточные узоры, возможно, использовались в Греции около 350 года до н. э.; в XVI веке они стали популярны в церквях.



VIII век (исламское искусство). Деталь наружной стены Большой мечети в Кордове, Испания, построенной около 785 года н. э.

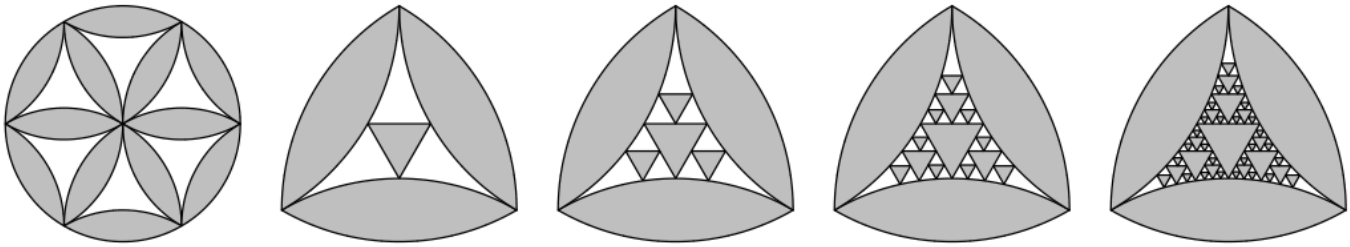
VIII век (кельты). Участок площадью менее двух дюймов на два дюйма внутри буквы P на чрезвычайно богато украшенной странице «хи-ро» Келлской книги — иллюминированной рукописи Евангелия, создававшейся в течение ряда лет в разных монастырях, вероятно начиная примерно с 800 года н. э. в ирландском монастыре на острове Айона, Шотландия. Даже на одной этой странице имеется, пожалуй, ещё с десяток очень похожих вложенных структур.

XII век (Италия). Окно Палатинской капеллы в Палермо, Сицилия, предположительно построенной около 1140 года н. э. Капелла характерна для так называемого арабо-норманнского стиля.

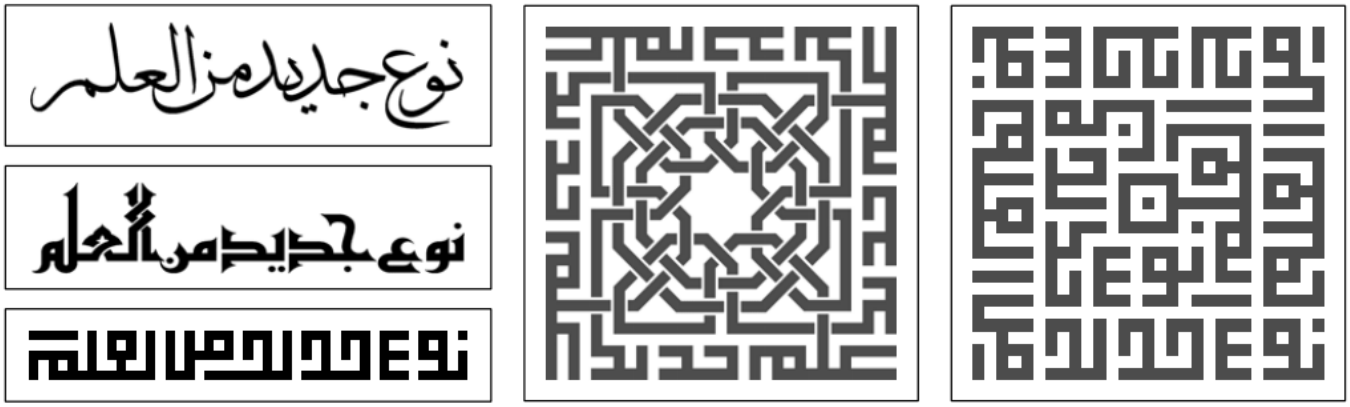
XIII век (Англия). Окно-роза «Око настоятеля» Линкольнского собора в Англии, построенное около 1225 года н. э. Подобные древовидные узоры встречаются во многих готических окнах примерно того же периода.

XIII век (Италия), четыре изображения. Мраморные мозаики на полу собора в Ананьи, Италия, выполненные около 1226 года н. э. Космой из группы Космати. Четвёртое изображение — увеличенный фрагмент третьего. Третье изображение, особенно часть, увеличенная на четвёртом, показывает приблизительную вложенную структуру, предположительно созданную так, как показано ниже. Все треугольники равносторонние, поэтому на одном шаге появляются треугольники нескольких размеров, хотя основная структура узора остаётся такой же, как у клеточного автомата по правилу 90. Ср. с аполлониевой упаковкой на с. 986. Группа Космати — в основном четыре поколения одной семьи — примерно с 1190 по 1300 год создавала сложные геометрические и другие мозаики, сочетавшие византийское, исламское и иные влияния. Преимущественно они работали в Риме и его окрестностях, но также, например, в Вестминстерском аббатстве в Англии. Треугольные формы с одним уровнем вложенности в их работах весьма распространены; три уровня вложенности, как здесь, —

редкость. Примечательно, что в более поздних подражаниях мозаикам Космати такие узоры почти никогда не использовались.



XIV век (исламское искусство). Стенное украшение мавзолея Пир-и-Бакран в Линджане, Иран, построенного около 1299–1312 годов. Этот узор — квадратная куфическая каллиграфическая запись широко цитируемого стиха Корана. Исходя из традиционного арабского письма насх, показанного ниже, куфический стиль начал развиваться около 900 года н. э.; примерно к 1100 году н. э. квадратное куфическое письмо уже использовалось в архитектурном орнаменте.



XIV век (исламское искусство). Облицованная плиткой стена в Севильском Алькасаре, Испания, построенном в 1364 году. Примерно тогда же тот же узор использовался в Альгамбре в Гранаде, Испания. Его можно получить, взяв сетку треугольников, а затем согласованно вдавливая внутрь или выдвигая наружу стороны каждого из них. Начиная с 1930-х годов такие узоры замечательным образом использовал Мауриц Эшер.

Другие случаи. Среди известных примеров неизбежно преобладают созданные из сохранившихся камня или керамики; несомненно, существовали и другие, например деревянные или тканевые. Пример работы с деревом — китайские решётки. В сохранившихся образцах преобладают повторяющиеся узоры, но стиль «ледяные трещины», вероятно восходящий к 100 году н. э., содержит приблизительную вложенность, хотя и с множеством случайных элементов. Все показанные узоры по существу двумерны. Пример одномерных орнаментальных узоров — профили архитектурных тяг. С античности они часто бывали весьма сложными, и мыслимо, что иногда их можно истолковать как проявляющие вложенность.

Распознавание искусства. Одна странная возможность состоит в том, что формы, подобные получаемым по правилу 30, могли быть созданы как произведения искусства давным-давно, но теперь не распознаваться в этом качестве. Ведь хотя в пещерном изображении животного легко узнать намеренно созданное искусство, точки, выбитые на скале в приблизительном соответствии с узором правила 30, можно даже не заметить как нечто человеческого происхождения. Однако, хотя в пещерах, возможно ещё с 30 000 года до н. э., встречается множество нарисованных узоров, кажущихся случайными, я был бы поражён, если бы какой-либо из них действительно был создан по определённым простым правилам. См. с. 839.

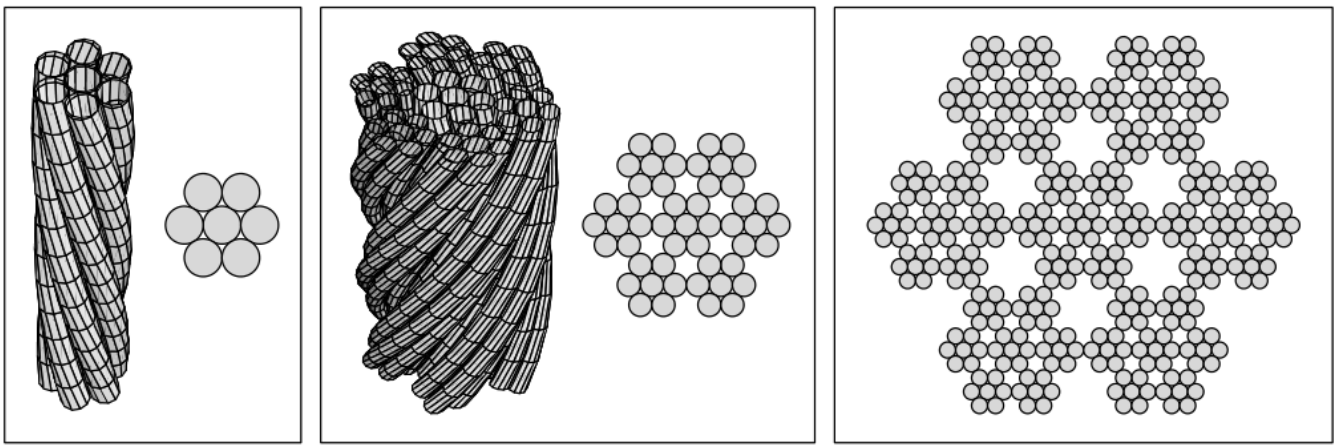
Понятие правил. Процессы, основанные на правилах, встречаются во множестве областей человеческой деятельности. Иногда правила служат главным образом ограничениями. Но нередко

их используют — как в клеточном автомате — для задания того, как должны строиться структуры. Однако почти без исключения в прошлом правила подбирались так, чтобы давать лишь вполне определённые и простые результаты. Помимо орнаментального искусства, к примерам с долгой историей относятся следующие.

Архитектура. Такие сооружения, как зиккураты и пирамиды, предположительно возводились путём соединения множества камней по простым правилам. Великая пирамида в Египте была построена около 2500 года до н. э. и содержит около двух миллионов больших камней. Для сравнения: изображения правила 30 на с. 29 и 30 содержат в общей сложности около миллиона клеток. Возможно, уже с 1000 года до н. э. индуистские храмы строили из сходных элементов разных масштабов, получая форму приблизительной вложенности. В римской и более поздней архитектуре помещения зданий довольно часто располагались в приблизительно вложенных узорах; крайний пример — Кастель-дель-Монте XIII века. Начиная со Средних веков многие персидские сады, например сады Тадж-Махала примерно 1650 года, имели довольно правильные вложенные структуры, получаемые несколькими последовательными делениями на четыре части. А с начала XIII века готические окна часто строились с несколькими уровнями приблизительно древовидных вложенных форм, как отмечено выше. В планировке городов вложенность, по-видимому, не использовалась, если не считать её слабого проявления в звёздчатых укреплениях Вобана, хотя в организационных структурах она обычна. Как указано выше, архитектурный орнамент также нередко фактически строился по определённым правилам.

Изготовление тканей. По-видимому, определённые правила применялись в ткачестве с ранних этапов человеческой истории. Но поскольку цель состоит в получении ткани, основное расположение нитей обычно всегда повторяется.

Верёвки. По крайней мере с 3000 года до н. э. верёвки изготавливали скручиванием прядей, которые сами были получены скручиванием; это даёт поперечные сечения с некоторой вложенностью, как на втором рисунке ниже. После появления стальных канатов в 1870-х годах использовались точные схемы конструкции, в том числе, по крайней мере в последнее время, схема 7/Cross7/Cross7, показанная последней ниже.



Узлы и фигуры из нитей. В течение многих тысяч лет определённые правила использовались для завязывания узлов и, предположительно, для изготовления фигур из нитей. Но когда правила включают больше нескольких шагов, они обычно становятся повторяющимися.

Складывание бумаги. Хотя складыванием бумаги, предположительно, занимаются по крайней мере 2000 лет, даже вложенная форма на с. 892, по-видимому, была замечена лишь совсем недавно.

Математика. Начиная с вавилонских времён арифметические действия выполнялись повторным применением простых правил к цифрам чисел. А с древнегреческих времён для построения геометрических фигур использовались итерационные методы. В конце XVII века возникла также идея, что математические доказательства можно считать состоящими из повторных применений определённых правил. Но идея исследовать возможные простые правила независимо от их назначения — получения результатов — по-видимому, так и не появилась. И по мере того как математика сосредоточивалась на непрерывных системах, понятие перечисления возможных правил становилось всё труднее применить.

Логика. Правила логики применяются примерно с 400 года до н. э. Но, кроме форм вроде силлогизмов, по-видимому, почти не изучалось порождение с их помощью узнаваемых узоров. См. с. 1099.

Грамматика. Идея, что человеческая речь строится из слов по определённым грамматическим правилам, существует по крайней мере приблизительно с 500 года до н. э., когда Панини дал грамматику санскрита. Менее формальные варианты этой идеи были распространены и в Древней Греции. Однако в основном лишь примерно в 1950-х годах грамматические правила начали рассматривать как предписания для порождения структур, а не просто как ограничения. См. с. 1103.

Поэзия. Определённые правила стихотворного ритма были хорошо разработаны уже в античности; а примерно к 200 году до н. э. индийские работы по перечислению их возможных форм, по-видимому, привели и к треугольнику Паскаля, и к числам Фибоначчи. Схемы рифмовки, включающие повторные перестановки длины 6 (секстина) и переплетённые повторяющиеся последовательности (терцина), использовались уже к XIV веку, в частности Данте.

Музыка. Простые последовательности и разные формы повторения, предположительно, используются в музыке по крайней мере со времён Пифагора. Начиная с XIII века иногда применялись более сложные формы переплетения, например каноны. А за последнее столетие несколько композиторов неявно или явно использовали структуры, основанные на простых системах подстановок Фибоначчи и других подобных системах. Заметим, что правила, например контрапункта, используются прежде всего как ограничения, а не как способы порождения структуры.

Военная муштра. Представление об использовании определённых правил для организации и маневрирования солдатских построений, по-видимому, существовало уже в вавилонские и ассирийские времена и было хорошо кодифицировано к римской эпохе. Довольно сложные случаи описывал, например, Никколо Макиавелли в 1521 году; но все они предназначались лишь для довольно простого поведения, например для перестроения колонны солдат в шеренги. См. задачу о синхронизации стрелков на с. 1035.

Игры. Игры обычно основаны на определённых правилах, но устроены так, что на каждом шаге приходится выбирать одну из множества возможностей — благодаря умению либо случайности. Игра го, возникшая до 500 года до н. э., а возможно ещё около 2300 года до н. э., — пример того, как особенно простые правила допускают поразительно сложные игровые узоры. В го чёрные и белые камни ставят на сетку, отчего внешне игра похожа на клеточный автомат.

Головоломки. Геометрические и арифметические головоломки, удивительно близкие к распространённым сегодня, по-видимому, существовали уже около 2000 года до н. э. Обычно они

основаны на ограничениях, и иногда их можно рассматривать как свидетельство того, что простые ограничения могут иметь сложные решения.

Криптография. Правила шифрования сообщений используются, возможно, с 2000 года до н. э.; нетривиальные повторяющиеся схемы стали обычными в XVI веке, но более сложные появились лишь значительно позже начала XX века. См. с. 1085.

Проектирование лабиринтов. От античности примерно до XVI века большинство лабиринтов следовало небольшому числу схем — чаще всего непосредственно основываясь на показанной на с. 873 или используя подобные ей составные части. Теперь известно, что возможно множество других схем.

Рисунки по правилам. Геометрические каракули довольно часто основаны на определённых правилах, но редко эти правила применяются долго, а сами рисунки сохраняются. Тем не менее часть задуманной Леонардо да Винчи в начале XVI века книги «Геометрические игры» сохранилась; в ней показаны сложные узоры, удовлетворяющие определённым ограничениям. Предпринимались разные попытки перечислить все возможные узоры некоторых простых типов. Примечательный пример — Себастьен Трюше, который в 1704 году рисовал двумерные узоры, по-разному сочетая четыре квадратные плитки с диагональной границей, повёрнутые в четыре возможные стороны.

С. 44 · Понимание природы. В греческую эпоху было замечено, что простые геометрические правила могут объяснить многие астрономические явления, в особенности видимое вращение звёзд и круглую форму Солнца и Луны. Однако было отмечено, что, за немногими исключениями вроде пчелиных сот, природные объекты на Земле, по-видимому, не подчиняются простым геометрическим правилам. Самыми сложными кривыми греческой геометрии были такие кривые, как циссоиды и конхоиды. Отсюда заключили, что лишь некоторые предположительно совершенные объекты, например небесные тела, можно

считать полностью доступными человеческому пониманию. Какие правила для природных объектов могли фактически испытываться в иудео-христианской традиции, менее ясно, хотя, например, в Книге Иова говорится о трудности «исчислить облака мудростью». И на протяжении Средних веков, за заметным исключением алхимиков, продолжали считать, что чудеса природы находятся за пределами человеческого понимания.

Атомизм. Представление, что всё может состоять из большого числа дискретных элементов, обсуждалось примерно около 450 года до н. э. Левкиппом и Демокритом. Несколько позже эпикурейцы предположили, что может быть достаточно немногих типов элементов; при этом проводилась аналогия, в частности Лукрецием около 100 года н. э., с тем, что разные сочетания букв могут образовать все слова языка. Но атомизм поддерживали лишь некоторые школы греческой философии, и вскоре он утратил популярность. Он возродился в конце XVII века, когда широко обсуждались корпускулярные теории света и материи. В начале XIX века рассуждения на основе атомов принесли успех в химии, а в конце XIX века для объяснения свойств вещества использовали статистическую механику больших совокупностей атомов. См. с. 1019. С возникновением квантовой теории в начале XX века было твёрдо установлено, что физические системы содержат дискретные частицы. Однако обычно предполагалось, что следует рассматривать лишь конкретные частицы с реалистическими механическими свойствами; поэтому абстрактные идеализации вроде клеточных автоматов не возникали. См. также с. 1027 и 1043.

История клеточных автоматов. Несмотря на чрезвычайную простоту их устройства, до приблизительно 1950-х годов, по-видимому, никто не рассматривал ничего похожего на клеточные автоматы общего вида. Но в 1950-х годах под разным влиянием появления электронных компьютеров были независимо введены несколько разных типов систем, эквивалентных клеточным автоматам. Можно указать ряд предшественников. Операции над последовательностями цифр с античности использовались в арифметике. Конечно-разностные приближения дифференциальных уравнений начали появляться в начале XX века и к 1930-м годам были довольно хорошо известны. А изобретённые в 1936 году машины Тьюринга основывались на рассмотрении произвольных операций над последовательностями дискретных элементов. Физические идеи вроде модели Изинга, по-видимому, прямого влияния не оказали.

Самый известный путь введения клеточных автоматов, в конечном счёте приведший и к их названию, связан с работой Джона фон Неймана над абстрактной моделью самовоспроизведения в биологии — темой, выросшей из кибернетических исследований. Примерно в 1947 году, возможно отталкиваясь от химической технологии, фон Нейман начал рассматривать модели трёхмерных фабрик, описываемых дифференциальными уравнениями в частных производных. Вскоре он переключился на робототехнику и воображал реализацию примера с помощью игрушечного конструктора. Однако по аналогии со схемами электронных цепей он понял, что двух измерений должно быть достаточно. И, последовав предложению Станислава Улама 1951 года, который, возможно, уже независимо рассматривал эту задачу, упростил модель и пришёл к двумерному клеточному автомату. По-видимому, он надеялся затем перенести результаты обратно на дифференциальные уравнения. Конкретный клеточный автомат, построенный им в 1952–1953 годах, имел 29 возможных цветов каждой клетки и сложные правила, специально устроенные для имитации работы компонентов электронного компьютера и разнообразных механических устройств. Чтобы математически доказать возможность самовоспроизведения, фон Нейман затем наметил конструкцию конфигурации из 200 000 клеток, способной воспроизвести себя; подробности разработал Артур Бёркс в начале 1960-х годов. По-видимому, фон Нейман считал — вероятно, отчасти под впечатлением сложности реальных биологических организмов и электронных компьютеров, — что примерно такой уровень сложности неизбежно необходим системе для проявления развитых возможностей вроде самовоспроизведения. В этой книге я показываю, что это совершенно не так; но, исходя из интуиции, сформированной существовавшими математикой и инженерией, фон Нейман, вероятно, никогда этого не представлял.

Из работы фон Неймана сразу выросли два направления. Первое, главным образом в 1960-х годах, состояло во всё более причудливых обсуждениях создания реальных самовоспроизводящихся автоматов, часто в форме космических аппаратов. Второе было попыткой глубже схватить сущность самовоспроизведения посредством математического исследования детальных свойств клеточных автоматов. В течение 1960-х годов находились конструкции всё более простых клеточных автоматов, способных к самовоспроизведению (см. с. 1179) и универсальным вычислениям (см. с. 1115). С начала 1960-х годов были замечены некоторые довольно простые общие свойства клеточных автоматов, считавшиеся важными для самовоспроизведения; их изучали со всё более сложным техническим формализмом. Пример — так называемый результат о «райском саде»: в клеточных автоматах могут существовать конфигурации, возникающие лишь как начальные условия; см. с. 961. Были также явно построены различные клеточные автоматы с поведением, проявляющим отдельные простые черты, возможно существенные для самовоспроизведения, например так называемая синхронизация стрелков на с. 1035.

К концу 1950-х годов было отмечено, что клеточные автоматы можно рассматривать как параллельные компьютеры. И особенно в 1960-х годах был доказан ряд всё более подробных и технических теорем — часто аналогичных теоремам о

машинах Тьюринга — об их формальных вычислительных возможностях. В конце 1960-х годов начались попытки связать клеточные автоматы с математическими исследованиями динамических систем, хотя, как обсуждается ниже, фактически это уже было сделано на десятилетие раньше, с другой терминологией. А к середине 1970-х годов работы по клеточным автоматам большей частью стали весьма специальными, и интерес к ним существенно угас. Тем не менее кое-какие работы продолжались, особенно в России и Японии. Заметим, что даже в информатике для клеточных автоматов использовали разные названия, включая мозаичные автоматы, клеточные пространства, итеративные автоматы, однородные структуры и универсальные пространства.

Как упоминалось в основном тексте, уже к концу 1950-х годов существовали всевозможные универсальные компьютеры, на которых легко можно было моделировать клеточные автоматы. Однако главным образом эти компьютеры применяли для изучения традиционных, гораздо более сложных систем, например дифференциальных уравнений в частных производных. И всё же около 1960 года было выполнено несколько моделирований, связанных с двумерными клеточными автоматами. Станислав Улам и другие использовали компьютеры Лос-Аламоса, чтобы получить несколько примеров того, что они называли рекурсивно определёнными геометрическими объектами: по существу, результатов эволюции обобщённых двумерных клеточных автоматов из отдельных чёрных клеток; см. с. 928. Особенно после получения более крупных изображений в 1967 году Улам отметил, что по крайней мере в одном случае довольно простые правила роста порождали сложный узор, и упомянул возможное отношение этого к биологии. Но, возможно потому, что традиционные математические методы почти не позволяли продвинуться в этом вопросе, результат не получил широкой известности и не был развит. Улам пытался построить одномерный аналог, но в итоге получил не клеточный автомат, а числовые последовательности, обсуждаемые на с. 908. Примерно в 1961 году Эдвард Фредкин моделировал двумерный аналог правила 90 на компьютере PDP-1 и отметил его свойства самовоспроизведения, см. с. 1179, но в целом был больше заинтересован в поиске простых физических черт.

Несмотря на отсутствие научных исследований, в начале 1970-х годов один пример клеточного автомата прочно вошёл в развлекательные вычисления. По-видимому, отчасти под влиянием вопросов математической логики, а отчасти работ Улама и других по «имитационным играм», Джон Конвей в 1968 году начал эксперименты с различными правилами двумерных клеточных автоматов: преимущественно вручную, но позднее и на компьютере PDP-7. К 1970 году он нашёл простой набор правил, названный им игрой «Жизнь», который демонстрирует разнообразное сложное поведение; см. с. 249. Во многом благодаря популяризации Мартином Гарднером в Scientific American «Жизнь» стала широко известна. Огромные усилия были потрачены на поиск специальных начальных условий, дающих определённые формы повторяющегося или иного поведения, но практически никаких систематических научных исследований не проводилось — возможно, отчасти потому, что даже Конвей относился к системе главным образом как к развлечению. И почти без исключения изучались лишь совершенно конкретные правила «Жизни». Впрочем, в 1978 году Джонатан Миллен, рассматривая возможный одномерный аналог «Жизни», более простой для реализации на ранних персональных компьютерах, ненадолго обратился к тому, что оказывается тоталистическим правилом с k = 2, r = 2 и кодом 20 на с. 283.

Совершенно независимо от всего этого уже в 1950-х годах конкретные типы двумерных и одномерных клеточных автоматов применялись в различных электронных устройствах и специализированных компьютерах. Когда в середине 1950-х годов началась цифровая обработка изображений, например для оптического распознавания знаков и подсчёта микроскопических частиц, для устранения шума обычно использовались именно правила двумерных клеточных автоматов. А начиная с 1960 года на протяжении нескольких десятилетий была создана длинная серия так называемых систем клеточной логики, реализующих двумерные клеточные автоматы преимущественно для обработки изображений. Большинство применявшихся правил специально подбиралось ради простого поведения; однако иногда, в основном ради развлечения, отмечали, что можно, например, получать узоры из чередующихся полос — «custering».

В конце 1950-х и начале 1960-х годов схемы миниатюризации электроники и ранние интегральные схемы часто основывались на одинаковых логических элементах, расположенных вдоль линий или на сетках и образующих так называемые клеточные массивы. В начале 1960-х годов некоторое время существовал интерес к итеративным массивам, в которых данные многократно пропускались через подобные системы. Но принципов проектирования было найдено мало, а технология изготовления микросхем с более сложными и менее однородными схемами быстро развивалась. Тем не менее начиная с 1960-х годов идея матричных или параллельных компьютеров неоднократно возвращалась — особенно в таких системах, как ILLIAC IV 1960–1970-х годов, систолические массивы и различные массово-параллельные компьютеры 1980-х годов. Однако предполагаемые правила работы каждого элемента таких систем обычно несравненно сложнее, чем правила любого из рассматриваемых мной простых клеточных автоматов.

По крайней мере с начала 1940-х годов электронные и иные цифровые линии задержки или сдвиговые регистры были обычным средством хранения данных, например цифр чисел. К концу 1940-х годов было отмечено, что так называемые сдвиговые регистры с линейной обратной связью (см. с. 974) способны порождать сложные выходные последовательности. Эти системы оказываются, по существу, одномерными аддитивными клеточными автоматами, подобными правилу 90, с ограниченным числом клеток; ср. с. 259. С середины 1950-х годов проводился обширный алгебраический анализ их поведения, однако в основном он сосредоточивался на таких вопросах, как периоды повторения, и даже не выявил явно

вложенные узоры. Связанный с этим анализ линейных рекуррентностей над конечными полями в отдельных случаях проводился ещё в XIX веке, а в некоторой подробности — в 1930-х годах. Одномерные клеточные автоматы общего вида связаны со сдвиговыми регистрами с нелинейной обратной связью. Некоторые исследования последних, в том числе удивительно близких к правилу 30 (см. с. 1088), Соломон Голомб проводил в 1956–1959 годах на специализированной аппаратуре для применения в помехоустойчивом радиоуправлении, но опять-таки сосредоточиваясь на вопросах вроде периодов повторения. Сдвиговые регистры с линейной обратной связью быстро получили широкое применение в связи. Регистры с нелинейной обратной связью, по-видимому, широко использовались в военной криптографии, но, несмотря на постоянные слухи, подробности сделанного остаются секретными.

В чистой математике бесконечные последовательности нулей и единиц в разных формах рассматривались по крайней мере с конца XIX века. С 1930-х годов развитие символической динамики (см. с. 960) привело к исследованию отображений таких последовательностей в самих себя. А к середине 1950-х годов, особенно Густавом Хедлундом, изучались так называемые блочные отображения, коммутирующие со сдвигом; они оказываются в точности одномерными клеточными автоматами, см. с. 961. В 1950-х и начале 1960-х годов в этой области, по крайней мере в США, работали многие выдающиеся чистые математики; но, поскольку значительная часть исследований предназначалась для криптографии, многое держалось в секрете. А публиковались преимущественно абстрактные теоремы о слишком глобальных свойствах, чтобы они могли выявить сложность того рода, которую я обсуждаю.

Конкретные типы клеточных автоматов возникали также, обычно под другими названиями, в чрезвычайно разнообразных обстоятельствах. В конце 1950-х и начале 1960-х годов системы, по существу являвшиеся одномерными клеточными автоматами, исследовались как способ оптимизировать схемы для арифметических и иных операций. С 1960-х годов при моделировании идеализированных нейронных сетей нейроны иногда соединяли с соседями на сетке, получая двумерный клеточный автомат. Сходным образом различные модели активных сред, особенно сердечной и других мышц, а также реакционно-диффузионных процессов использовали дискретную сетку и дискретные состояния возбуждения, что соответствует двумерному клеточному автомату. В физике дискретные идеализации статистической механики и динамические варианты систем вроде модели Изинга иногда были близки к клеточным автоматам — за исключением существенного отличия: случайности, встроенной в их основные правила. Аддитивные клеточные автоматы, например правило 90, неявно возникали при исследованиях биномиальных коэффициентов по модулю простых чисел в XIX веке (см. с. 870), но появлялись и в других контекстах, например в «лесах низкорослых деревьев», изучавшихся около 1970 года.

И всё же к концу 1970-х годов, несмотря на все эти направления, исследования систем, эквивалентных клеточным автоматам, в основном угасли. Иронично, что это произошло как раз тогда, когда компьютеры впервые стали широко доступны для поисковых исследований. Но в некотором смысле это было удачей: когда в 1981 году я начал работать над клеточными автоматами, это позволило мне определить область по-новому, хотя, к некоторому моему позднейшему сожалению, я решил, стремясь отдать должное истории, использовать название «клеточные автоматы» для изучавшихся мной систем. Публикация моей первой статьи о клеточных автоматах в 1983 году (см. с. 881) привела к быстрому росту интереса к этой области. С тех пор на протяжении лет о клеточных автоматах публиковалось всё больше статей, как показывает приведённое ниже количество исходных документов в Science Citation Index; почти все они следовали направлениям, определённым мной.

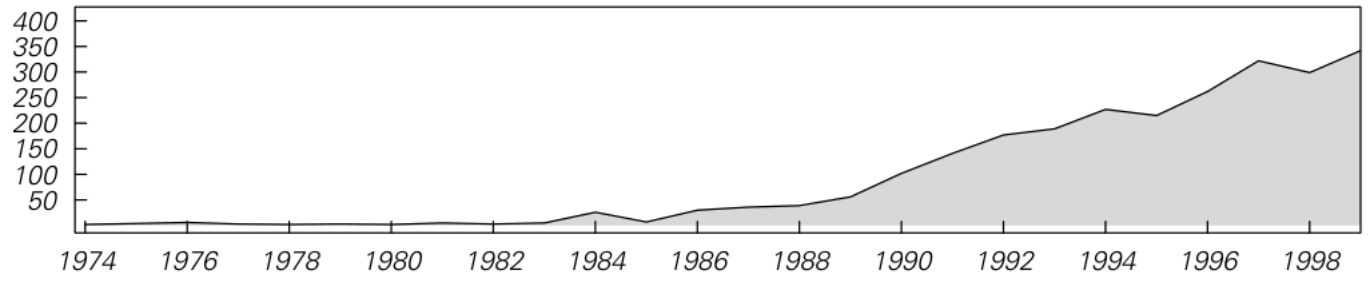


График числа документов: годы 1974, 1976, 1978, 1980, 1982, 1984, 1986, 1988, 1990, 1992, 1994, 1996, 1998; значения 50, 100, 150, 200, 250, 300, 350, 400.

Близкие подходы. Основные явления этой главы много раз в прошлом были более или менее близки к открытию. Историческая последовательность главных примеров, по-видимому, такова:
• VI–III века до н. э.: предположительно замечено, что просто сформулированные задачи, например поиск простых или совершенных чисел, имеют сложные решения; но этому не придают общего значения, см. с. 132 и 910.
• XIII век: изучаются последовательности Фибоначчи, треугольник Паскаля и другие числовые построения по правилам, однако обнаруживается лишь простое поведение.
• XVI век: Леонардо да Винчи экспериментирует с правилами, соответствующими простым геометрическим ограничениям (см. с. 875), но находит только простые формы, удовлетворяющие этим ограничениям.
• XVIII век: Леонард Эйлер и другие вычисляют представления чисел, задаваемых простыми формулами, в виде цепных дробей (см. с. 143 и 915), в одних случаях отмечая регулярность, а в других не делая никаких замечаний.
• XVIII и XIX века: замечено, что цифры π и других трансцендентных чисел проявляют кажущуюся случайность (см. с. 136), но идея рассматривать её как результат процесса вычисления не возникает.
• XIX век: распределение простых чисел подробно исследуется, но в основном рассматривают его закономерности, а не нерегулярности. См. с. 132.

• XIX век: в задаче трёх тел обнаруживается сложное поведение, но предполагается, что с усовершенствованием математических методов оно в конце концов будет объяснено. См. с. 972.
• 1880-е годы: Джон Венн и другие отмечают кажущуюся случайность цифр π, но почему-то принимают её как данность.
• 1906 год: Аксель Туэ исследует простые системы подстановок (см. с. 893) и находит поведение, которое кажется сложным, хотя оказывается вложенным.
• 1910-е годы: Гастон Жюлиа и другие изучают итерируемые отображения, но сосредоточиваются на свойствах, допускающих простое описание.
• 1920 год: Моисей Шейнфинкель вводит комбинаторы (см. с. 1121), но рассматривает в основном случаи, специально построенные так, чтобы соответствовать обычным логическим функциям.
• 1921 год: Эмиль Пост рассматривает простую тег-систему (см. с. 894), поведение которой трудно предсказать, но, не сумев ничего о ней доказать, переходит к другим задачам.
• 1920 год: вводится модель Изинга, но изучается только статистика конфигураций, а не какая-либо динамика.
• 1931 год: Курт Гёдель доказывает теорему Гёделя (см. с. 782), но используемые им конструкции настолько сложны, что он и другие предполагают: простые системы никогда не смогут проявлять подобные явления.
• Середина 1930-х годов: Алан Тьюринг, Алонзо Чёрч, Эмиль Пост и другие вводят разные модели вычислений, но используют их для построения доказательств и не исследуют действительное поведение простых примеров.
• 1930-е годы: ставится задача 3n + 1 (см. с. 904), и обнаруживается непредсказуемое поведение, но главное внимание сосредоточено на доказательстве простого результата о ней.
• Конец 1940-х и 1950-е годы: разрабатываются генераторы псевдослучайных чисел (см. с. 974), но их рассматривают как уловки, поведение которых не имеет особого научного значения.
• Конец 1940-х и начало 1950-х годов: в довольно простых электронных устройствах, построенных для иллюстрации кибернетических идей, иногда наблюдается сложное поведение, но его обычно считают тем, чего следует избегать.
• 1952 год: Алан Тьюринг применяет компьютеры для изучения биологических систем, но использует традиционные математические модели, а не, скажем, машины Тьюринга.
• 1952–1953 годы: Джон фон Нейман теоретически исследует сложные клеточные автоматы, но не пытается рассматривать более простые случаи или моделировать системы на компьютере.
• Середина 1950-х годов: Энрико Ферми с сотрудниками моделирует простые системы нелинейных пружин на компьютере, но не замечает, что простые начальные условия могут привести к сложному поведению.
• С середины 1950-х до середины 1960-х годов: конкретные двумерные клеточные автоматы применяются для обработки изображений; замечают несколько правил, проявляющих несколько сложное поведение, но считают их интересными лишь для развлечения.
• Конец 1950-х годов: выполняется компьютерное моделирование итерируемых отображений, но внимание сосредоточено главным образом на повторяющемся поведении. См. с. 918.
• Конец 1950-х годов: идеи теории динамических систем начинают применять к системам, эквивалентным одномерным клеточным автоматам, но подробности конкретного поведения не изучаются, кроме тривиальных случаев.
• Конец 1950-х годов: идеализированные нейронные сети моделируются на цифровых компьютерах, но наблюдаемое несколько сложное поведение считается главным образом помехой исследованию интересующих явлений и не изучается. См. с. 1099.
• Конец 1950-х годов: Берни Олдер и Томас Уэйнрайт выполняют компьютерное моделирование динамики идеализированных молекул — твёрдых сфер, но сосредоточиваются на крупномасштабных свойствах, не проявляющих сложности. См. с. 999.
• 1956–1959 годы: Соломон Голомб моделирует сдвиговые регистры с нелинейной обратной связью, некоторые с правилами, близкими к правилу 30, но исследует в основном периоды повторения, а не их подробное сложное поведение. См. с. 1088.

• 1960, 1967 годы: Станислав Улам с сотрудниками моделирует системы, близкие к двумерным клеточным автоматам, и отмечает появление сложных узоров, как описано выше.
• 1961 год: Эдвард Фредкин моделирует двумерный аналог правила 90 и отмечает свойства, сводящиеся к вложенности, как описано выше.
• Начало 1960-х годов: студенты MIT пробуют запускать множество небольших компьютерных программ, а в некоторых случаях визуализировать их результат. Они находят разные примеры, например «munching foos», дающие вложенное поведение (см. с. 871), но не идут дальше.
• 1962 год: Марвин Минский и другие изучают множество простых машин Тьюринга, но не заходят достаточно далеко, чтобы обнаружить сложное поведение, показанное на с. 81.
• 1963 год: Эдвард Лоренц моделирует дифференциальное уравнение, проявляющее сложное поведение (см. с. 971), но сосредоточивается на отсутствии периодичности и чувствительной зависимости от начальных условий.
• Середина 1960-х годов: моделируются случайные булевы сети (см. с. 936), но внимание сосредоточено на простых усреднённых свойствах.

• 1970 год: Джон Конвей вводит двумерный клеточный автомат — игру «Жизнь», как описано выше.
• 1971 год: Майкл Патерсон рассматривает класс простых двумерных машин Тьюринга, которые называет «червями» и которые проявляют сложное поведение. См. с. 930.
• 1973 год: я рассматриваю некоторые двумерные клеточные автоматы, но требую от правил свойств, препятствующих сложному поведению. См. с. 864.
• Середина 1970-х годов: Бенуа Мандельброт развивает идею фракталов (см. с. 934) и подчёркивает важность компьютерной графики в исследовании сложных форм.
• Середина 1970-х годов: Томмазо Тоффоли моделирует все 4096 двумерных клеточных автоматов простейшего типа, но исследует главным образом лишь их стабилизацию из случайных начальных условий.
• Конец 1970-х годов: Дуглас Хофштадтер изучает рекурсивную последовательность со сложным поведением (см. с. 907), но не продвигается достаточно далеко, чтобы сделать существенные выводы.
• 1979 год: Бенуа Мандельброт открывает множество Мандельброта (см. с. 934), но сосредоточивается на его вложенной структуре, а не на его общей сложности.
• 1981 год: я начинаю изучать одномерные клеточные автоматы и получаю небольшое изображение, аналогичное изображению правила 30 на с. 27, но не исследую его.
• 1984 год: я подробно исследую правило 30 и начинаю понимать значение его самого и подобных систем.

Важность явного представления. Если пролистать эту книгу, одним из её разительных отличий от большинства прежних научных изложений окажется наличие столь многочисленных явных изображений, показывающих поведение каждого элемента системы. В прошлом считалось более научным приводить лишь числовые сводки таких данных. Но большинство явлений, обсуждаемых мной в этой книге, невозможно было бы обнаружить без таких явных изображений. См. также с. 108.

Моя работа над клеточными автоматами. Я начал всерьёз работать над клеточными автоматами в середине 1981 года. До этого я некоторое время размышлял о том, как в природных системах могут возникать сложные узоры, по видимости нарушая второй закон термодинамики. Меня особенно интересовали самогравитирующие газы, у которых основная физика казалась ясной, но в которых, по-видимому, возникали сложные явления вроде образования галактик. Меня также интересовали нейронные сети: довольно простые их модели были разработаны Уорреном Маккаллоком и Уолтером Питтсом в 1940-х годах. Я пришёл к клеточным автоматам, пытаясь уловить существенные свойства целого ряда систем — от самогравитирующих газов до нейронных сетей. Я хотел найти модели, имеющие простую структуру, подобную модели Изинга в статистической механике, изучаемой с 1920-х годов, но при этом обладающие определёнными правилами эволюции во времени и легко моделируемые на компьютере. По иронии, хотя клеточные автоматы пригодны для многого, они оказались довольно неподходящими для моделирования и самогравитирующих газов, и нейронных сетей. См. с. 1021. Но к тому времени, когда я это понял, уже было ясно, что клеточные автоматы чрезвычайно интересны для множества других целей.

Свои первые крупные компьютерные эксперименты с клеточными автоматами я провёл в конце 1981 года, см. с. 19. Сначала меня больше всего поразили две особенности. Во-первых, начиная со случайных начальных условий, клеточные автоматы могли самоорганизовываться и давать сложные узоры. Во-вторых, в случаях вроде правила 90 простые начальные условия приводили к вложенным, или фрактальным, узорам. В первой половине 1982 года я напряжённо анализировал поведение клеточных автоматов, используя идеи статистической механики, теории динамических систем и дискретной математики. А в июне 1982 года закончил первую статью о клеточных автоматах — «Статистическая механика клеточных автоматов». Опубликованная в журнале Reviews of Modern Physics в июле 1983 года, эта статья уже в первоначальной форме содержала многие ключевые идеи, приведшие к развитию науки, описанной в этой книге. В ней обсуждается, что, отказавшись от традиционных математических уравнений, можно потенциально построить простые модели, воспроизводящие сложные явления; упоминаются и некоторые следствия рассмотрения моделей

вроде клеточных автоматов как вычислительных систем. Статья также содержала небольшое изображение правила 30, запущенного из одной чёрной клетки. Но тогда я не исследовал это изображение подробно и молчаливо предполагал, что всякая наблюдаемая мной случайность должна происходить из использованных случайных начальных условий. См. с. 112.

Где-то осенью 1981 года, за ужином с тогда ещё молодыми специалистами по информатике из MIT, я впервые узнал, что один из вариантов изобретённых мной систем ранее уже исследовался под названием «клеточные автоматы». Об игре «Жизнь» я знал, но её развлекательная направленность отбивала желание её изучать. Узнав название «клеточные автоматы», я смог разыскать довольно много относящихся к ним статей 1950–1960-х годов. Но обнаружил, что активные исследования того, что называли клеточными автоматами, более или менее угасли, с небольшим исключением группы в MIT, тогда главным образом занятой созданием специализированной аппаратуры для двумерных клеточных автоматов. К концу 1982 года препринты моей статьи о клеточных автоматах вызвали значительный отклик, и я включился в организацию конференции, прошедшей в марте 1983 года в Лос-Аламосе, чтобы свести вместе многочисленных новых энтузиастов клеточных автоматов и прежних исследователей этой области.

Готовясь к этой конференции, я решил использовать графические возможности только что полученной мной новой рабочей станции — одного из самых ранних компьютеров Sun Microsystems — для систематического исследования поведения большого набора разных клеточных автоматов. И, проведя несколько недель за просмотром экрана за экраном с узорами и попытками анализировать их свойства, пришёл к выводу, что в поведении клеточных автоматов со случайными начальными условиями можно выделить всего четыре основных класса, каждый со своими характерными особенностями. См. с. 231.

В 1982 и начале 1983 года мои попытки анализировать клеточные автоматы основывались главным образом на идеях дискретной математики и теории динамических систем. В течение 1983 года я также начал серьёзно использовать теорию формальных языков и теорию вычислений. Но в основном я сосредоточивался на характеристике поведения, получаемого из всех возможных начальных условий. И фактически всё ещё смутно предполагал, что простые начальные условия дадут лишь довольно простое поведение. В нескольких моих статьях в действительности были достаточно подробные изображения, опровергающие это. Я их замечал, но они никогда не входили в число глубоко изученных мной примеров — отчасти по поверхностной причине: соответствующие правила были несимметричны либо неизбежно приводили к узорам, неудобным для показа по другим причинам. Я не знаю точно, что заставило меня пристальнее взглянуть на простые начальные условия, хотя, кажется, впервые систематически получил изображения высокого разрешения всех клеточных автоматов с k = 2, r = 1 как пробное задание для раннего лазерного принтера — вероятно, в начале 1984 года. И я точно знаю, например, что 1 июня 1984 года распечатал изображения правила 30, правила 110 и тоталистического правила с k = 2, r = 2 и кодом 10, см. примечание ниже, взял их с собой в самолёт из Нью-Йорка в Лондон и через несколько дней уже рассказывал в Швеции о случайности в правиле 30 и её потенциальном значении.

Примерно через месяц написание статьи для Scientific American, формально посвящённой программному обеспечению в науке и математике, заставило меня тщательнее обдумать основные вопросы вычислений и моделирования и впервые описать идею вычислительной несводимости, см. с. 737. Осенью 1984 года я начал исследовать некоторые следствия своих открытий о клеточных автоматах для фундаментальных вопросов науки. И к началу 1985 года написал две, как я считаю, самые фундаментальные, хотя и чрезмерно короткие, статьи того периода: одну — о неразрешимости и вычислительной трудности в теоретической физике, другую — о внутреннем порождении случайности и происхождении случайности в физических системах.

В начале лета 1985 года я консультировал молодую компанию Thinking Machines Corporation, разработавшую массово-параллельный компьютер Connection Machine, который неплохо подходил для моделирования клеточных автоматов. Отчасти ради приложения для этого компьютера я в итоге подробно исследовал правило 30 и его случайность, среди прочего предложив его как практический генератор случайных последовательностей и криптосистему.

Я всегда считал, что клеточные автоматы могут помочь подступиться к фундаментальным вопросам термодинамики и гидродинамики. И в середине 1985 года, отчасти пытаясь найти применения Connection Machine, разработал практическую схему вычислений в механике жидкости с помощью клеточных автоматов, см. с. 378. Затем в течение той зимы и следующей весны я анализировал эту схему и устанавливал её соответствие традиционному континуальному подходу.

Однако к 1986 году я почувствовал, что ответил по крайней мере на первый круг очевидных вопросов о клеточных автоматах. Всё сильнее казалось, что с имеющимися вычислительными средствами продвинуться дальше будет нелегко. В июне 1986 года я организовал последнюю конференцию по клеточным автоматам, а затем, в августе 1986 года, по существу оставил эту область, чтобы приступить к разработке Mathematica.

За прошедшие годы я снова и снова возвращался к клеточным автоматам и каждый раз удивлялся и восхищался богатством проявляемых ими явлений. Как я доказываю в этой книге, чрезвычайно

широкий круг систем в конечном счёте должен проявлять одни и те же основные явления. Но клеточные автоматы, особенно одномерные, делают их особенно ясными. Именно поэтому, даже исследовав самые разнообразные другие системы, в качестве примеров в этой книге я по-прежнему чаще всего использую одномерные клеточные автоматы.

Мои статьи. Основные опубликованные мной статьи о клеточных автоматах и других вопросах, связанных с этой книгой, были следующими. Даты указывают, когда я закончил работу над каждой статьёй; фактически статьи выходили через 6–12 месяцев.
• «Статистическая механика клеточных автоматов» (июнь 1982 года): введение одномерных клеточных автоматов и исследование многих их свойств.
• «Алгебраические свойства клеточных автоматов» (совместно с Оливье Мартеном и Эндрю Одлыжко; февраль 1983 года): анализ аддитивных клеточных автоматов, например правила 90.
• «Универсальность и сложность в клеточных автоматах» (апрель 1983 года): классификация поведения клеточных автоматов.
• «Теория вычислений клеточных автоматов» (ноябрь 1983 года): описание поведения с помощью теории формальных языков.

• «Двумерные клеточные автоматы» (совместно с Норманом Паккардом; октябрь 1984 года): распространение результатов на два измерения.
• «Неразрешимость и вычислительная трудность в теоретической физике» (октябрь 1984 года): введение вычислительной несводимости.
• «Происхождение случайности в физических системах» (февраль 1985 года): введение внутреннего порождения случайности.
• «Порождение случайных последовательностей клеточными автоматами» (июль 1985 года): подробное исследование правила 30.
• «Термодинамика и гидродинамика клеточных автоматов» (совместно с Джеймсом Сейлемом; ноябрь 1985 года): континуальное поведение из клеточных автоматов.
• «Подходы к инженерии сложности» (декабрь 1985 года): поиск систем, достигающих заданных целей.
• «Клеточно-автоматные жидкости: основная теория» (март 1986 года): вывод уравнений Навье—Стокса из клеточных автоматов.

Идеи первых пяти и самой последней из этих статей были за последние примерно пятнадцать лет достаточно хорошо усвоены. Но идеи остальных пяти — нет; и, по-видимому, чтобы представить их надлежащим образом, действительно требуется всё построение этой книги.

Другие мои значительные публикации, содержащие относящиеся к делу обобщения, были следующими. Здесь даты относятся к фактической публикации: иногда почти одновременно с написанием, но иногда со значительной задержкой.
• «Компьютеры в науке и математике» (сентябрь 1984 года): статья в Scientific American об основах вычислительного подхода к науке и математике.
• «Клеточные автоматы как модели сложности» (октябрь 1984 года): вводная статья о клеточных автоматах в Nature.
• «Геометрия биномиальных коэффициентов» (ноябрь 1984 года): аддитивные клеточные автоматы и вложенные узоры.
• «Двадцать задач теории клеточных автоматов» (1985 год): список нерешённых задач для исследования, большинство которых теперь наконец решены в этой книге.
• «Таблицы свойств клеточных автоматов» (июнь 1986 года): свойства элементарных клеточных автоматов.
• «Криптография с клеточными автоматами» (1986 год): использование правила 30 как криптосистемы.
• «Теория сложных систем» (1988 год): речь 1984 года, предлагающая направление исследований для нового Института Санта-Фе.

Код 10. По многим меркам правило 30 — самый простой клеточный автомат, порождающий случайность из единственной исходной чёрной клетки. Но есть и другие простые примеры, которые исторически я заметил немного раньше правила 30, хотя и не исследовал: они встречаются среди тоталистических правил с k = 2, r = 2. И действительно, из 64 таких правил 13 проявляют случайность. Показанный ниже пример — код 10. Он задаёт, что, если чёрными являются 1 или 3 клетки из 5, следующая клетка будет чёрной; иначе она будет белой.

