ЗАДАНИЕ 11

Элементарные клеточные автоматы

Построение модели • Выполним реализацию элементарных клеточных автоматов с различными видами граничных условий. Модель должна поддерживать два типа начальных условий: случайное (0 и 1 с вероятностью 1/2); единственная центральная клетка в состоянии 1.

A Создаем новую модель. Устанавливаем ее размер max-pxcor равным 75 и max-pycor равным 50, размер патча — 4 пикселям.

B Добавляем к интерфейсу слайдер (или элемент ввода) number, представляющий номер клеточного автомата (диапазон слайдера от 0 до 255, шаг 1).

C Сразу напишем код процедуры `make-rules`, которая преобразует номер автомата `number` в список `rules` из восьми двоичных цифр — правый столбец соответствующей таблицы правил. Для этого создаем глобальную переменную `rules`. Пишем процедуру `make-rules`, в которой фактически реализуем классический алгоритм перевода числа в десятичной системе счисления в *восьмиразрядное* число в двоичной системе, основанный на последовательном делении исходного числа на 2:

```
to make-rules
  let n number
  set rules (list)
  repeat 8 [
    set rules lput (n mod 2) rules
    set n floor (n / 2)
  ]
end
```

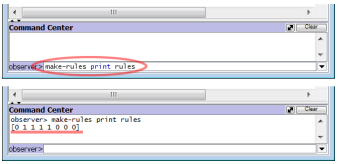

РИС. 11.1 Ввод команд в командном центре

Работу любой процедуры или функции в NetLogo можно проверять, не внося изменений в модель, используя командный центр в нижней части вкладки с интерфейсом (рис. 11.1). Например, наберите в поле ввода команду `make-rules print rules`. Если процедура реализована корректно, то для автомата с номером 30 система должна напечатать строку `[0 1 1 1 1 0 0 0]`.

D Добавляем к интерфейсу кнопку setup, два элемента ввода color-0 и color-1 с типом Color для представления цвета клеток в состояниях 0 и 1, выпадающий список init-state с опциями `"single 1"` и `"random"`, задающий начальное состояние клеточного автомата.

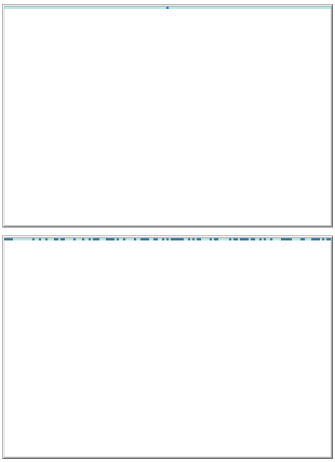

РИС. 11.2 Начальное состояние модели для двух типов начальных условий

E Создаем процедуру `setup`, в которой очищаем модель, с помощью команды `make-rules` строим таблицу правил автомата, просим все патчи выполнить команду настройки `setup-patch`, сбрасываем таймер модели.

F Пишем код процедуры `setup-patch`. Предварительно приписываем всем патчам атрибут `state`. Автомат в начальном состоянии будет изображаться верхней строчкой патчей, после первой итерации — второй строчкой сверху и т.д. Поэтому любой патч в модели может находиться в трех состояниях: 0 и 1 — состояния самого автомата, −1 — состояние еще необработанных патчей. Таким образом, в процедуре `setup-patch` устанавливаем состояние патча следующим образом. Если патч располагается не в верхней строке (т.е. `pycor != max-pycor`), то его состояние полагается равным −1. Иначе состояние патча устанавливается, исходя из значения переменной `init-state`: для начального условия `"single 1"` единичное состояние получает только патч с нулевой координатой `pxcor`, для начального условия `"random"` состояние 0 или 1 выбирается случайным образом с равной вероятностью.

G В процедуре `recolor` устанавливаем цвет патча согласно его состоянию: для состояния −1 выбираем белый цвет, для состояний 0 и 1 — цвета `color-0` и `color-1` соответственно. Проверяем работу кнопки setup для обоих типов начальных условий (рис. 11.2).

H Добавляем к интерфейсу кнопку go, а к коду модели процедуру go. В этой процедуре определяем текущую строку:

```
let c-l max-pycor - ticks - 1
```

и просим все патчи в этой строке (с таким значением координаты pycor) обновить свое состояние командой update-patch. Если текущая строка была самой нижней строкой модели, то останавливаем работу модели командой stop. Иначе обновляем таймер.

I Добавляем к интерфейсу выпадающий список boundary, соответствующий граничным условиям, с тремя опциями: "cyclic", 0 и 1 (без кавычек).

J Пишем код процедуры update-patch, центральной для данной модели, т.к. именно в ней реализуется основная логика автомата. Если текущий патч является крайним правым или левым и граничное условие не равно "cyclic", то его состояние полагаем равным значению boundary (это будет либо 0, либо 1). В противном случае состояние определяется согласно системе правил автомата. Текущие состояния клетки, соответствующей данному патчу, и двух ее соседей представляются в нашей модели тремя патчами над данным патчем. Вычислим их состояния (a, b и c) с помощью команды patch-at, которая возвращает патч, по его относительным координатам[1]:

```
let a [state] of patch-at -1 1
let b [state] of patch-at 0 1
let c [state] of patch-at 1 1
```

Далее трактуем эти три двоичных значения как цифры числа в двоичной системе счисления (номер строки в таблице правил), которое переводим в число в десятичной системе счисления:

```
let k 4 * a + 2 * b + c.
```

Устанавливаем состояние клетки согласно системе правил rules:

```
set state item k rules.
```

Последней командой обновляем цвет патча (команда recolor).

K Окончательный интерфейс модели показан на рис. 11.3. Проверяем работу модели для разных ав-

[1] Эта команда, в частности, корректно обрабатывает ситуацию с патчами, лежащими на границах модели.

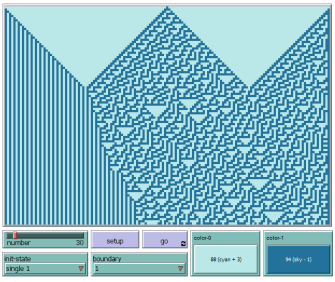

РИС. 11.3 Окончательный интерфейс модели

томатов, используя разные начальные и граничные условия (рис. 11.4).

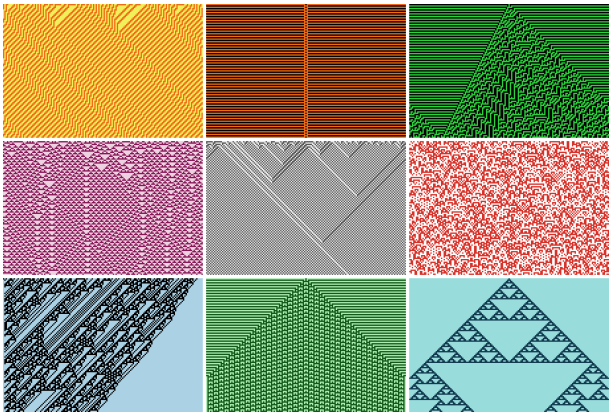

РИС. 11.4 Примеры различных элементарных клеточных автоматов

УПРАЖНЕНИЯ

1 Выполните с помощью построенной модели визуальную классификацию всех 256 элементарных клеточных автоматов по Вольфраму, используя случайные начальные конфигурации. Попытайтесь понять, как визуальные характеристики автомата следуют из его системы правил.

2 Включите в модель поддержку других типов начальных условий (рис. 11.5):

- случайное заполнение клеток автомата нулями и единицами с заданной пользователем плотностью единиц $\rho \in [0 \ldots 1]$;
- случайный кластер в центре автомата заданной ширины с заданной плотностью единиц;
- периодическое заполнение клеток автомата заданным пользователем образцом.

ТАБЛ. 11.1 Система правил стохастического автомата

q_{i-1}^t	q_i^t	q_{i+1}^t	q_i^{t+1}
0	0	0	0.0
0	0	1	1.0
0	1	0	0.9
0	1	1	1.0
1	0	0	1.0
1	0	1	0.0
1	1	0	1.0
1	1	1	0.0

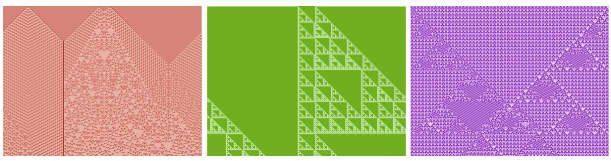

РИС. 11.5 Разные типы начальных условий

3 В *стохастических* клеточных автоматах правила имеют вероятностный характер. Например, такой автомат можно определить с помощью модифицированной таблицы, в которой правый столбец содержит числа в диапазоне от 0 до 1, которые трактуются как вероятности появления в текущей клетке единицы. Реализуйте этот подход. На рис. 11.6 приведен пример работы стохастического автомата с системой правил, показанной в таблице 11.1.

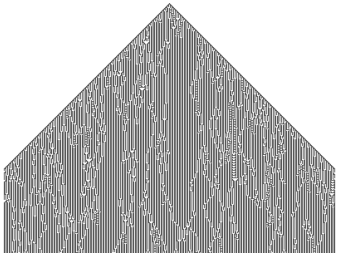

РИС. 11.6 Пример работы стохастического клеточного автомата

4 В *асинхронных* клеточных автоматах клетки обновляют свои состояния в случайном порядке одна за другой. Для вычисления нового состояния клетки используются *текущие* состояния ее соседей, даже если они были обновлены уже на данной итерации. Чтобы из рассмотренной в главе модели сделать асинхронную, нужно брать состояние соседней клетки либо с текущей строки (если это состояние не равно -1, т.е. уже вычислено на данной итерации), либо с предыдущей строки, как это было в исходной версии. Включите в модель поддержку двух режимов работы: синхронного и асинхронного. Сравнение двух режимов для автомата № 22 приведено на рис. 11.7.

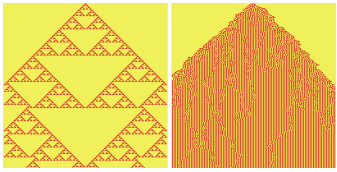

РИС. 11.7 Синхронный и асинхронный режим работы автомата № 22

5 Выполните *анимированную* реализацию элементарных клеточных автоматов, в которой клетки автомата занимают, например, центральную строку патчей (один патч — одна клетка). Цвет патчей этой строки соответствует текущему состоянию автомата (т.е. для визуализации времени в модели используется не ее вертикальная ось, а само время — в виде номера итерации).

6 Элементарные автоматы можно обобщить, например, увеличением их ранга. Если ранг равен r, то число двоичных клеточных автоматов равно $2^{2^{2r+1}}$. Уже для $r = 2$ число таких автоматов превышает 4 миллиарда, т.е. перебрать их все оказывается нереальным. Модифицируйте модель для случая $r = 2$ и проведите выборочное исследование нескольких автоматов со случайными номерами, которые можно генерировать командой `set number random (2 ^ 32)` (рис. 11.8).

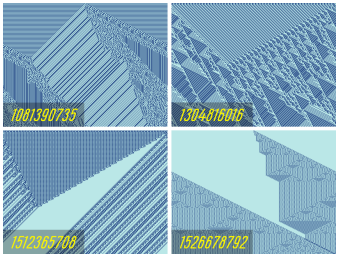

РИС. 11.8 Двоичные автоматы ранга $r = 2$

7 Двоичный клеточный автомат называется *тоталистическим*, если новое состояние клетки зависит от своего текущего состояния и от *суммы* состояний ее соседей. Например, для автомата ранга $r = 1$ имеется три варианта таких сумм, от 0 до 2. Каждой комбинации суммы и текущего состояния можно приписать новое состояние, равное либо 0, либо 1. Следовательно, всего имеется $2 \cdot 2^3 = 16$ тоталистических элементарных автоматов (найдите их номера). Реализуйте и протести-

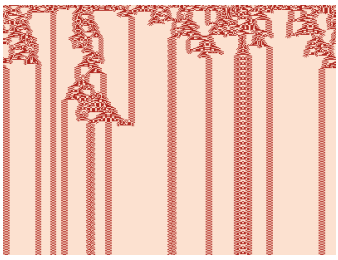

РИС. 11.9 Тоталистический двоичный автомат ранга 4

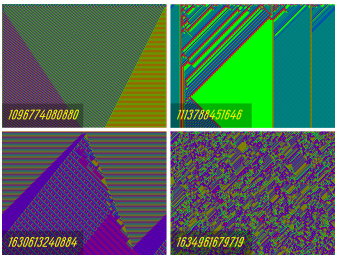

РИС. 11.10 Троичные автоматы ранга $r = 1$

руйте модель тоталистических автоматов произвольного ранга. Например, на рис. 11.9 приведен пример автомата ранга $r = 4$, в котором а) клетка переходит из состояния 0 в состояние 1, если сумма состояний ее соседей равна 3, 4 или 5; б) клетка остается в состоянии 1, если эта сумма равна 4 или 5.

8 Еще одним способом обобщения является увеличения числа m состояний клеток автомата. Количество автоматов ранга 1 с числом состояний m равно m^{m^3}, уже для минимального случая $m = 3$ это число примерно равно 7 триллионам. Реализуйте модель троичных автоматов ранга 1 и выполните выборочное ее исследование (рис. 11.10).

9 Хотя некоторые из элементарных клеточных автоматов и обладают алгоритмической универсальностью, программирование их является весьма нетривиальной задачей[2]. Но если расширить ранг автомата и увеличить число его состояний, то можно создавать автоматы, эмулирующие работу различных менее сложных алгоритмических моделей. Например, с помощью автомата ранга 3 можно создать модель машины Тьюринга. Придумайте схему реализации такого автомата и алгоритм преобразования системы правил машины Тьюринга в систему правил клеточного автомата. На рис. 11.11 приведен пример работы клеточного автомата, который моделирует сразу три машины Тьюринга, выполняющих операцию инкремента числа в двоичной системе счисления (белые клетки — нули, черные — единицы).

[2] Ситуация называется «тьюринговым болотом», когда теоретически мы можем реализовать любой алгоритм, а практически реализация даже простейших алгоритмов оказывается сверхтяжелой задачей.

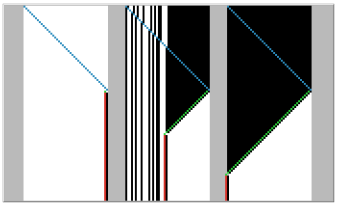

РИС. 11.11 Моделирование работы машины Тьюринга, выполняющей операцию прибавления единицы к числу в двоичной системе счисления

10 *Реверсивным* (обратимым) клеточным автоматом называется автомат, в котором по текущему состоянию можно однозначно восстановить состояние автомата на предыдущем шаге[3]. Среди элементарных клеточных автоматов имеется только 6 реверсивных (автоматы под номерами 15, 51, 85, 170, 204 и 240), причем их поведение является достаточно простым (рис. 11.12). Реверсивность имеет важное значение, если мы собираемся использовать клеточный автомат для моделирования физических процессов, большинство из которых обладает свойством обратимости. Реверсивные автоматы, обладающие нетривиальным поведением, легко строятся на основе клеточных автоматов *второго порядка*, в которых новое состояние клетки определяется состояниями ее соседей на *двух* предыдущих шагах. Например, возьмем за основу некоторое правило элементарного клеточного автомата. Пусть для текущей клетки это правило дает значение p, а состояние

[3] N. Margolus, *Physics-like models of computation*, Physica D: Nonlinear Phenomena, 1984, Vol. 10, Issue 1–2, p. 81–95.

этой клетки на предыдущем шаге было равно q, тогда ее новое состояние определяется операцией строгой дизъюнкции $p \oplus q$. Модифицируйте описанную в главе модель, включив в нее поддержку реверсивного режима работы автомата (рис. 11.13).

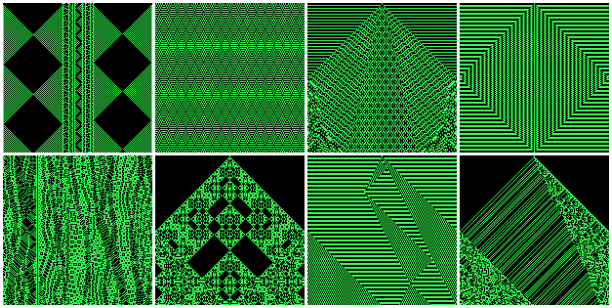

РИС. 11.13 Примеры реверсивных клеточных автоматов второго порядка

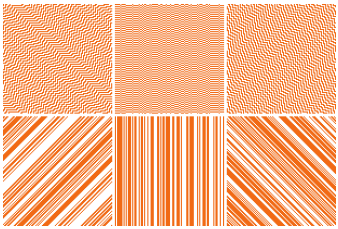

РИС. 11.12 Реверсивные элементарные клеточные автоматы