# Клеточные автоматы. Реализация и эксперименты

Лев Наумов, Анатолий Шалыто
levnaumov@mail.ru

Настоящая работа призвана привлечь внимание программистов к такой чрезвычайно увлекательной и полезной области дискретной математики, как клеточные автоматы, которые могут обладать весьма сложным поведением, несмотря на простоту описания его клеток.

Один из крупнейших специалистов в области информатики Марвин Минский в работе [1] писал, что самым «важным в этой области, по-видимому, является изучение различных путей возникновения сложного поведения из простых устройств, действий, описаний или концепций». Нам кажется, что не существует ничего более подходящего для этой цели, чем клеточные автоматы.

В последние десятилетия в мире выполнен ряд исследований в этом направлении. Большинство из них было посвящено анализу динамических систем.

К упомянутой области, в частности, относится теория фракталов [2]. Однако используемый в этой теории формализм, основанный на решении рекуррентных нелинейных уравнений, который связан с применением теории функций комплексных переменных, является нетривиальным.

В работе [3] на основе применения рекурсивных процедур при написании программ предложен другой подход к построению фракталов, который, однако, не упрощает понимания процессов вычислений.

К рассматриваемой области принадлежит также и теория хаоса [4], которая признана, наряду с теорией относительности и квантовой теорией [5], одним из основных теоретических достижений XX века. Нельзя сказать, что математический аппарат, применяемый в этой теории, прост.

В рассматриваемой области весьма актуальны исследования клеточных автоматов. Для этих устройств, как отмечалось выше, характерно, что они порождают сложное поведение даже при использовании действительно очень простого и наглядного математического аппарата.

В настоящей статье, применяя клетки, поведение каждой из которых описывается одной булевой формулой, демонстрируется различные формы самовоспроизведения (фрактальный рост, клонирование, перемещение). При этом рассматривается два класса клеток: с памятью, обладающие всего лишь двумя состояниями, и без памяти.

Отметим, что булева алгебра, видимо, является наиболее простой и красивой из наук вообще. Ведь недаром физик-теоретик академик А. Б. Мигдал, которому было с чем сравнивать, в работе [6], посвященной красоте в науке, выделяет именно булеву алгебру, которая всего лишь при двух значениях переменных и

нескольких простейших операциях является основой всех информационных технологий.

В настоящей работе, в отличие от других работ, например [7], кроме теоретических положений и иллюстраций, приводятся также и полные тексты программ.

## Клеточные автоматы

Клеточные автоматы были предложены в работе фон Неймана [8]. Большой интерес к ним связан с тем, что автоматы этого класса являются универсальной моделью параллельных вычислений, также как машины Тьюринга являются аналогичной моделью для последовательных вычислений [9].

Клеточный автомат – дискретная динамическая система, представляющая собой совокупность одинаковых клеток, одинаковым образом соединенных между собой. Все клетки образуют, так называемую, решетку клеточного автомата. Решетки могут быть разных типов, отличаясь как по размерности, так и по форме клеток.

Каждая клетка является конечным автоматом, состояния которого определяются состояниями соседних клеток и, возможно, ее собственным состояниям.

Отметим, что в клеточных автоматах, как моделях вычислений, не рассматриваются входные и выходные воздействия. При аппаратной реализации клеточные автоматы обычно называют однородными структурами [10].

Клеточные автоматы в общем случае характеризуются следующими свойствами.

1. **Изменения значений всех клеток происходят одновременно** после вычисления нового состояния каждой клетки решетки.
2. **Решетка однородна**. Невозможно отличить никакие два места на решетке по ландшафту.
3. **Взаимодействия локальны**. Лишь клетки окрестности (как правило, соседние) способны повлиять на данную клетку.
4. **Множество состояний клетки конечно**.

Клеточные автоматы можно реализовать разными способами. В настоящей работе используется следующий подход.

1. Вводятся два массива для хранения состояний клеток. Первый из них содержит текущее состояние каждой клетки. Второй массив предназначен для хранения нового ее состояния.
2. Определяется функция переходов клетки решетки. Для определения следующего ее состояния в качестве параметров в функцию переходов передаются текущие значения состояний клеток окрестности, возможно, включая ее саму. Эту функцию будем задавать в виде булевой формулы.

3. На нулевом шаге производится заполнение решетки (первого массива) начальными данными. Для выбранных решетки и функции переходов клетки это полностью определяет поведение системы.
4. Для вычисления новых состояний вводится цикл. На каждой итерации для каждой клетки, используя в качестве переменных элементы первого массива, вычисляется ее новое состояние, которое помещается во второй массив. Значения аргументов функции переходов при этом берутся из первого массива.
5. После завершения итерации значения из всех элементов второго массива переносятся в первый массив. Этим обеспечивается псевдопараллельное изменение значений состояний всех клеток решетки.
6. Осуществляется визуализация содержимого решетки. В зависимости от ее размерности (одномерная или двумерная) отображение решетки производится по-разному.
   6.1. Для одномерной (линейной) решетки после *каждой* итерации выводится соответствующая ей строка. Расположение этих строк одна под другой позволяет наблюдать динамику системы во времени (ось времени направлена вертикально вниз).
   6.2. Для двумерной (плоскостной) решетки в каждый момент времени отражается результат лишь *последней* итерации. Последовательный переход от итерации к итерации позволяет наблюдать динамику системы.
7. По нажатию клавиши <Enter> выполняется переход к пункту 4, а нажатие на клавишу <q> приводит к завершению работы программы.

Для облегчения понимания листингов программ применяется не графический (как в настоящей статье), а текстовый вывод информации. Программы, использованные в примерах, можно найти на сайте http://is.ifmo.ru в разделе «Статьи».

## Одномерные клеточные автоматы

В одномерном (линейном) клеточном автомате решетка представляет собой цепочку клеток (одномерный массив), в которой для каждой из них, кроме крайних, имеется по два соседа. Для устранения краевых эффектов решетка «заворачивается» в тор. Это позволяет использовать следующее соотношение для всех клеток автомата

$$y'[i] = f(y[i-1], y[i], y[i+1]),$$

где f – функция переходов клетки;
y'[i] – состояние i-й клетки в следующий момент времени;
y[i-1] – состояние (i-1)-й клетки в данный момент времени;
y[i] – состояние i-й клетки в данный момент времени;
y[i+1] – состояние (i+1)-й клетки в данный момент времени.

Рассмотрим три автомата, отличающихся функцией переходов клетки. Решетка каждого из этих автоматов представляет собой цепочку из 61 клетки, каждая из которых может находиться в одном из двух состояний.

*Пример 1*. Пусть функция переходов клетки имеет следующий вид:

$$y'[i] = y[i-1] \mid y[i] \mid y[i+1],$$

где | – символ логической операции «дизъюнкция» в языке программирования Си.

В качестве исходных значений выберем единицу для центральной клетки и ноль – для всех остальных клеток.

Программа, реализующая одномерный клеточный автомат, структура которой была описана в предыдущем разделе, приведена в листинге 1. В нем жирным начертанием выделены функция переходов клетки и начальное заполнение решетки, так как они специфичны для данного автомата. Остальной код является универсальной оболочкой для произвольных одномерных клеточных автоматов.

*Листинг 1. Одномерный клеточный автомат*

```
#include "stdafx.h"
#include "stdio.h"

// Количество клеток
const int count=61;

// Функция, обеспечивающая заворачивание структуры
// клеточного автомата в тор для избежания краевых
// эффектов
inline int TorIt(int x)
{
      if (x<0) return x+count; else return x%count;
}

// Функция поведения клетки
int f(int y1,int y2, int y3)
{
      return y1|y2|y3;
}

// Главная функция приложения
int main(int argc, char* argv[])
{
      // Массив для хранения текущих состояний клеток
      int y[count];

      // Массив для хранения новых состояний клеток
      int y1[count];

      // Переменная, используемая при обработке нажатия кнопок
      // <ENTER> и <q>
      char c;

      // Начальное заполнение массива клеток
      for (int i=0; i<count; i++)
      {
            y[i]=0;
      }
      y[30]=1;
```

```
	// Итерации автомата
	for (;;)
	{
		// Визуализация
		for (i=0; i<count; i++)
			printf("%d",y[i]);

		// Вычисление новых состояний клеток
		for (i=0; i<count; i++)
		{
			y1[i]=f(y[TorIt(i-1)],y[TorIt(i)],y[TorIt(i+1)]);
		}

		// Перенос новых состояний в массив текущих состояний
		for (i=0; i<count; i++)
			y[i]=y1[i];

		// Обработка нажатия кнопок <ENTER> и <q>
		c=getchar();
		if (c=='q') return 0;
	}

	return 0;
}
```

Эта программа позволяет представить поведение клеточного автомата во времени (рис. 1). Такое поведение может быть названо «пирамида».

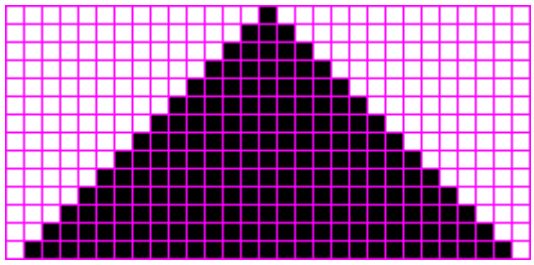

Рис. 1. Поведение первого одномерного клеточного автомата (пример 1)

*Пример 2.* Если, сохраняя функцию переходов, в начальном заполнении решетки изменить состояние еще одной клетки с нуля на единицу, то поведение автомата становится другим (рис. 2). Оно может быть названо «горы».

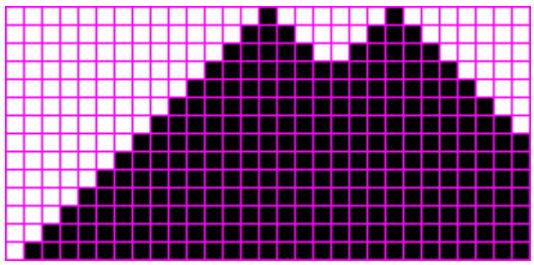

Рис. 2. Поведение первого одномерного клеточного автомата (пример 2)

*Пример 3.* Второй клеточный автомат характеризуется следующей функцией переходов:

$$y'[i] = y[i-1] \mid y[i] \wedge y[i+1],$$

где ^ – символ логической операции «неравнозначность» («сумма по модулю два») в языке программирования Си.

Выбирая в качестве начального тоже заполнение решетки, что и в примере 1, получим поведение, которое может быть названо «пробор».

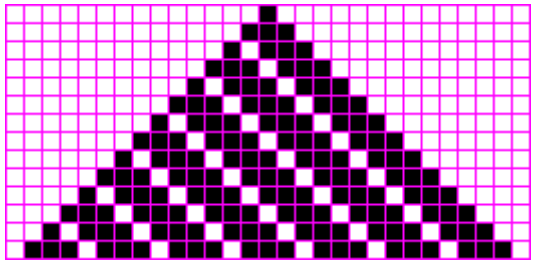

Рис. 3. Поведение второго одномерного
клеточного автомата (пример 3)

*Пример 4.* Выберем для третьего клеточного автомата следующую функцию переходов:

$$y'[i] = y[i-1] \wedge y[i] \wedge y[i+1].$$

Этот автомат, при тех же начальных условиях, что и в предыдущем примере, порождает фрактальное поведение [2]. Он является весьма простой моделью, демонстрирующей самовоспроизведение (рис. 4).

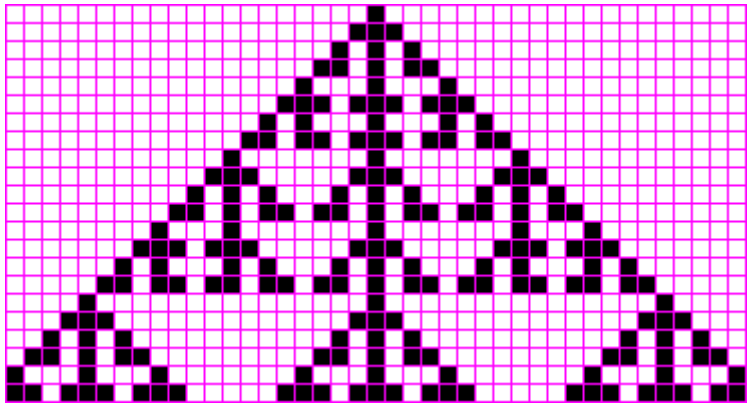

Рис. 4. Поведение третьего одномерного
клеточного автомата (пример 4)

## Двумерные клеточные автоматы

### *Окрестность из восьми клеток*

В двумерном (плоскостном) клеточном автомате решетка реализуется двумерным массивом. В ней каждая клетка имеет восемь соседей. Для устранения краевых эффектов решетка так же, как и в предыдущем случае, «заворачивается» в тор. Это позволяет использовать следующее соотношение для всех клеток автомата:

$$y'[i][j] = f(y[i][j], y[i-1][j], y[i-1][j+1], y[i][j+1], y[i+1][j+1], y[i+1][j], y[i+1][j-1], y[i][j-1], y[i-1][j-1]).$$

*Пример 5.* Наиболее известным из двумерных клеточных автоматов является автомат, моделирующий игру «Жизнь» [13]. В этом автомате, как и во всех, рассмотренных выше, клетки могут находиться в двух состояниях. Функция переходов клеток реализует следующие условия:

- если данная клетка мертва (находится в состоянии «ноль»), то она оживет (перейдет в состояние «единица») при условии, что у нее имеется ровно три живых соседа;
- если данная клетка жива, то она останется живой только при условии, что у нее есть два или три живых соседа и умрет в противном случае.

В этой игре интерес представляет наблюдение за развитием популяции клеток при различных начальных условиях.

Программа, реализующая игру «Жизнь», структура которой была описана выше, приведена в листинге 2. В этой программе решетка имеет размеры 23 на 23, а в качестве начальных условий выбран «планер» [13].

*Листинг 2. Двумерный клеточный автомат*

```
#include "stdafx.h"
#include "stdio.h"

// Количество клеток
const int count=23;

// Функция, обеспечивающая заворачивание структуры
// клеточного автомата в тор для избежания краевых
// эффектов
inline int TorIt(int x)
{
      if (x<0) return x+count; else return x%count;
}

// Функция поведения клетки
// U - Верх; UR - Верх-Право; R - Право; DR - Низ-Право;
// D - Низ; DL - Низ-Лево; L - Лево; UL - Верх-Лево
int f(int y, int yU, int yUR, int yR, int yDR, int yD, int yDL, int
yL, int yUL)
{
```

```
	// Вычисление количества живых соседей
	int i=yU+yUR+yR+yDR+yD+yDL+yL+yUL;

	// Мертвая клетка оживет, если у нее имеется 3 живых соседа
	if (y==0 && i==3) return 1;

	// Живая клетка останется живой, если у нее имеется 2 или 3
	// живых соседа
	if (y==1 && (i==2 || i==3)) return 1;

	// В остальных случаях клетка будет мертвой
	return 0;
}

// Главная функция приложения
int main(int argc, char* argv[])
{
	// Массив для хранения текущих состояний клеток
	int y[count][count];

	// Массив для хранения новых состояний клеток
	int y1[count][count];

	// Переменная, используемая при обработке нажатия кнопок
	// <ENTER> и <q>
	char c;

	// Счетчик итераций
	long iter=0;

	// Начальное заполнение массива клеток
	for (int i=0; i<count; i++) for (int j=0; j<count; j++)
	{
		y[i][j]=0;
	}
	y[11][11]=1;
	y[10][11]=1;
	y[ 9][11]=1;
	y[11][12]=1;
	y[10][13]=1;

	// Итерации автомата
	for (;;)
	{
		// Визуализация
		for (i=0; i<count; i++)
		{
			for (int j=0; j<count; j++)   printf("%d",y[i][j]);
			printf("\n");
		}
		printf("#%d",iter);

		// Вычисление новых состояний клеток
		for (i=0; i<count; i++) for (int j=0; j<count; j++)
		{
			y1[i][j]=f(y[TorIt(i)][TorIt(j)],
			y[TorIt(i-1)][TorIt(j)],y[TorIt(i-1)][TorIt(j+1)],
			y[TorIt(i)][TorIt(j+1)],y[TorIt(i+1)][TorIt(j+1)],
			y[TorIt(i+1)][TorIt(j)],y[TorIt(i+1)][TorIt(j-1)],
			y[TorIt(i)][TorIt(j-1)],y[TorIt(i-1)][TorIt(j-1)]);
		}
```

```
            // Перенос новых состояний в массив текущих состояний
            for (i=0; i<count; i++) for (int j=0; j<count; j++)
                  y[i][j]=y1[i][j];

            // Обработка нажатия кнопок <ENTER> и <q>
            c=getchar();
            if (c=='q') return 0;

            iter++;
      }

      return 0;
}
```

«Планер» был выбран в качества примера, так как является наиболее простым из «летающих» объектов. Он «летает» в том смысле, что перемещается за счет самовоспроизведения, происходящего с периодом в четыре шага.

*Пример 6.* В качестве примера фрактального поведения рассмотрим клеточный автомат, функционирующий по правилу:

$$y'[i][j] = y[i][j] \wedge y[i-1][j] \wedge y[i-1][j+1] \wedge y[i][j+1] \wedge y[i+1][j+1] \wedge y[i+1][j] \wedge \wedge y[i+1][j-1] \wedge y[i][j-1] \wedge y[i-1][j-1].$$

Данная функция принципиально не отличается от функции, приведенной в примере 4, так как они обе вычисляют сумму по модулю два от состояний всех соседей и самой клетки.

На рис. 5 приведена начальная конфигурация решетки, а на рис. 6 – результат ее самовоспроизведения через восемь шагов.

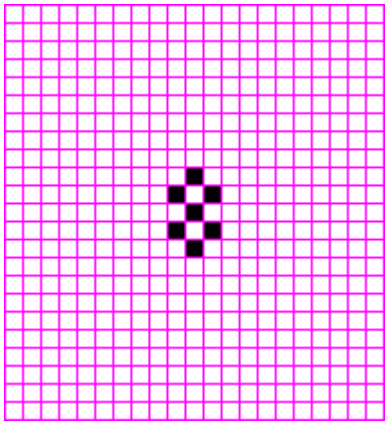

Рис. 5. Начальная конфигурация
(пример 6)

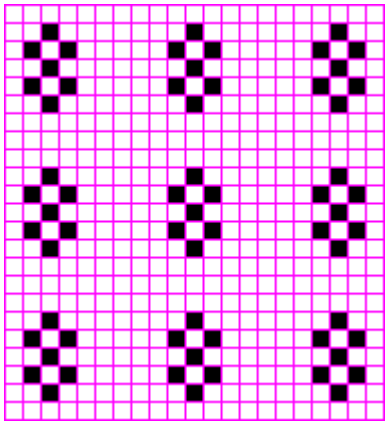

Рис. 6. Конфигурация через восемь
итераций (пример 6)

Необходимо отметить, что в силу свойств функции «сумма по модулю два», самовоспроизведение имеет место после любого числа шагов, являющегося степенью двух, начиная со значения, определяемого соотношением:

$$T = 2^{\lceil \log_2(\max(a;b)) \rceil},$$

где a и b – ширина и высота «зародыша», соответственно.

Для начальной конфигурации, изображенной на рис. 5, a=3, b=5. Поэтому T=8, и следовательно самовоспроизведение имеет место через 8, 16, 32, 64 и т.д. шагов.

***Окрестность из четырех клеток***

Обратим внимание, что при переходе к окрестностям с меньшим числом соседей, например с четырьмя (как в данном случае), программа (листинг 2) не требует изменений, за исключением, функции переходов и, быть может, начальных условий. Основное изменение функции переходов состоит в том, что в реализующей ее формуле используются не все входные переменные.

Будем рассматривать в качестве окрестности данной клетки лишь те из них, которые имеют общую сторону с ней и называются «главными соседями».

*Пример 7.* Для иллюстрации фрактального поведения автомата с такой окрестностью выберем следующую функцию переходов:

$$y'[i][j] = y[i][j] \wedge y[i-1][j] \wedge y[i][j+1] \wedge y[i+1][j] \wedge y[i][j-1].$$

Вновь используем в качестве начального заполнения конфигурацию, изображенную на рис. 5. Самовоспроизведение может происходить независимо, как по горизонтали, так и по вертикали, на шагах, номера которых определяются соотношениями:

$$T_a = 2^{\lceil \log_2 a \rceil} \quad \text{и} \quad T_b = 2^{\lceil \log_2 b \rceil}.$$

При этом период полного самовоспроизведения вычисляется на основе соотношения:

$$T = \max(T_a; T_b).$$

Для рассматриваемого «зародыша» (рис. 5) период самовоспроизведения по горизонтали $T_a = 4$ (рис. 7), а по вертикали – $T_b = 8$ (рис. 8). Поэтому полный период самовоспроизведения $T = 8$.

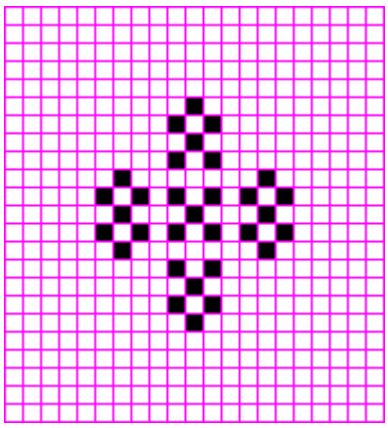

Рис. 7. Конфигурация через четыре шага (пример 7)

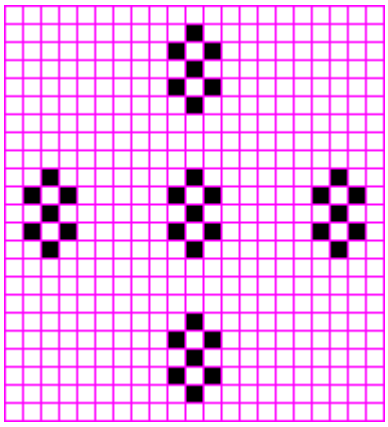

Рис. 8. Конфигурация через восемь шагов (пример 7)

Клеточный автомат с окрестностью из четырех клеток демонстрирует более разнообразное (в смысле самовоспроизведения) поведение, по сравнению с предыдущим автоматом, клетки которого обладают большим числом соседей. Это объясняется тем, что в нем имеет место принципиально другой механизм передачи информации от «диагональных соседей» клетки.

*Пример 8.* Сохранив функцию переходов и выбрав в качестве начальной конфигурации единицу в центральной точке решетки, на пятнадцатом шаге получим

конфигурацию, изображенную на рис. 9, а на двадцать девятом – изображенную на рис. 10.

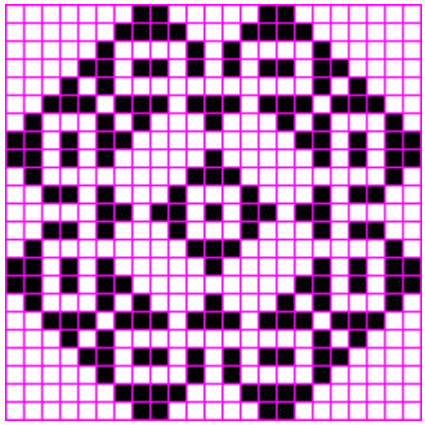

Рис. 9. Конфигурация на
пятнадцатом шаге (пример 8)

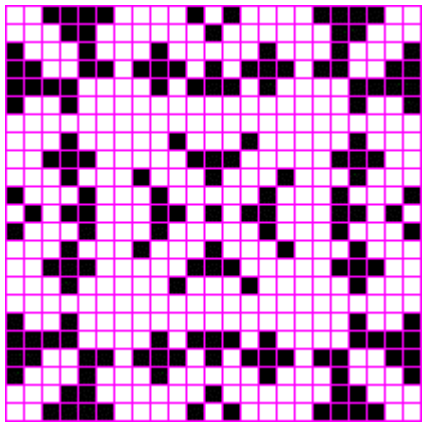

Рис. 10. Конфигурация на
двадцать девятом шаге (пример 8)

При этом отметим, что, несмотря на то, что решетка размером 23 на 23 может находиться в $2^{23 \cdot 23}$ состояниях, для выбранных функции переходов и начальных условий, она может быть лишь в 1024 различных состояниях, повторяющихся периодически. Сокращение числа состояний в данном случае обеспечивается только за счет размеров решетки и «заворачивания» ее в тор. Если бы решетка имела большие размеры, то на этих шагах состояния были бы другими, так как после одиннадцатого шага противоположные края рассматриваемой решетки начинают влиять друг на друга. Это объясняется тем, что информация распространяется за шаг на одну клетку, и поэтому из центральной клетки она дойдет до границы за (23–1)/2 шагов.

## Автоматы с клетками без памяти

Обратим внимание на то, что во всех рассмотренных выше примерах в функции переходов каждой клетки, наряду с входными переменными, соответствующи-

ми состояниям клеток ее окрестности, входит также и переменная, отражающая состояние самой клетки. Поэтому этот класс клеточных автоматов может быть назван «автоматами из клеток с памятью».

Принципиально другой класс клеточных автоматов возникает, если отказаться от использования состояния данной клетки в качестве входной переменной функции переходов. Такой класс назовем «автоматами с клетками без памяти».

При этом отметим, что, несмотря на то, что в автоматах этого класса клетки памятью не обладают, за счет переобозначения переменных, такие автоматы в целом имеют память.

Этот класс автоматов является простейшим, применение которого обеспечивает самовоспроизведение.

*Пример 9.* Приведем пример простейшего клеточного автомата, обеспечивающего самовоспроизведение. Его можно построить, если в примере 4, при тех же начальных условиях, упростить функцию переходов:

$$y'[i] = y[i-1] \wedge y[i+1].$$

Поведение этого автомата во времени показано на рис. 11.

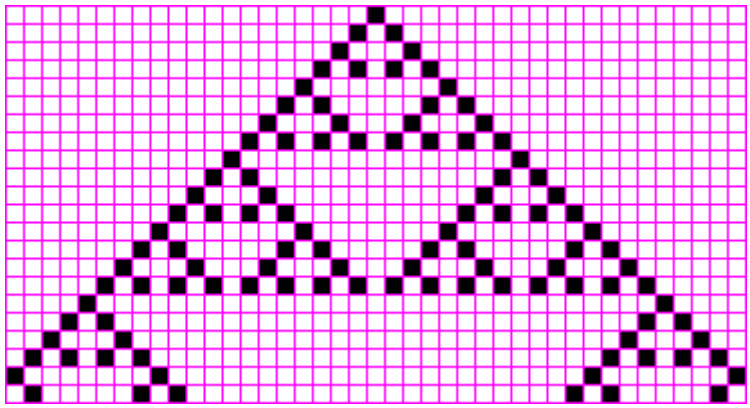

Рис. 11. Поведение одномерного
автомата с клетками без памяти

Интерактивная программа, реализующая пример двумерного автомата рассматриваемого класса с функцией переходов

$$y'[i][j] = y[i-1][j] \wedge y[i][j+1] \wedge y[i+1][j] \wedge y[i][j-1],$$

приведена в работе [14].

## Заключение

В настоящее время на смену термину «искусственный интеллект», пришел более широкий термин – «искусственный интеллект и искусственная жизнь» [15].

По мнению авторов, настоящую работу можно рассматривать, как введение в «искусственную жизнь».

Кроме того, отметим, что активно развивается такая наука, как синергетика. Ее название происходит от греческих слов «син» – совместный и «эргос» – действовать [16]. Поэтому синергетика – это наука о совместном согласованном поведении многих элементов, как единого целого в составе сложной системы. Из изложенного следует, что данная работа может рассматриваться также и как введение в «синергетику». Она может быть полезна также в физике [17], биологии и т.д. Широкий круг вопросов, посвященных решению задач с использованием клеточных автоматов, рассмотрен в работе [18].

Автоматы, клетки которых могут пребывать в числе состояний большем двух, будут рассмотрены в отдельной работе, которая также будет размещена на сайте http://is.ifmo.ru в разделе «Статьи».

Работа выполнена при поддержке Российского фонда фундаментальных исследований по гранту №02-07-90114 «Разработка технологий автоматного программирования»

## Литература

1. *Минский М.* Вычисления и автоматы. М.: Мир, 1971.
2. *Пайтген Х.О., Рихтер П.Х.* Красота фракталов. Образы комплексных динамических систем. М.: Мир, 1993.
3. *Вирт Н.* Алгоритмы и структуры данных. СПб.: Невский диалект, 2001.
4. *Пригожин И., Стенгерс И.* Порядок из хаоса. Новый диалог человека с природой. М.: Эдиториал УРСС, 2000.
5. *Пайс А.* Гении науки. М.: Институт компьютерных исследований, 2002.
6. *Мигдал А.Б.* Красота в науке. М.: Молодая гвардия, 1979.
7. *Тоффоли Т., Марголус Н.* Машины клеточных автоматов. М.: Мир, 1991.
8. *фон Нейман Дж.* Теория самовоспроизводящихся автоматов. М.: Мир, 1971.
9. *Шалыто А., Туккель Н.* От тьюрингова программирования к автоматному // Мир ПК. 2002, №2.
10. *Варшавский В.И., Мараховский В.Б., Песчанский В.А., Розенблюм Л.Я.* Однородные структуры. М.: 1973.
11. Интервью с основателем программы «Mathematica» С. Вольфрамом // Компьютер-информ. 2002. №2 – http://www.ci.ru.
12. *Дьяконов В.* Mathematica 4: учебный курс. СПб.: Питер, 2001.
13. *Гарднер М.* Математические досуги. М.: Мир, 1972.
14. *Федотьев А.* Клеточный автомат – http://rain.ifmo.ru/~fedotiev/.
15. *Тарасов В.Б.* От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. М.: Эдиториал УРСС, 2002.
16. *Колесников А.А.* Шанс для рывка // Поиск. 2002. №42.
17. *Фейнман Р.* Моделирование физики на компьютерах // Квантовый компьютер и квантовые вычисления. Ижевск: Регулярная и хаотическая динамика, 1999.
18. *Wolfram S.* A New Kind of Science. Wolfram Press, 2003.

## Об авторах

**Наумов Лев Александрович** – студент кафедры «Компьютерные технологии» Санкт-Петербургского государственного института точной механики и оптики (технического университета) – СПбГИТМО (ТУ). E-mail: levnaumov@mail.ru.

**Шалыто Анатолий Абрамович** – профессор кафедры «Компьютерные технологии» СПбГИТМО (ТУ). E-mail: shalyto@mail.ifmo.ru. Сайт: http://is.ifmo.ru.