С.В. Гаврилов, И.В. Матюшкин, А.Л. Стемпковский

Вычислимость в клеточных автоматах[1]

Аннотация. В обзоре обсуждаются проблемы организации вычислений с помощью клеточных автоматов. Показывается, что общность парадигмы коннекционизма позволяет переносить ряд методов, применимых для нейронных сетей, в предметное поле клеточных автоматов. Специальные вопросы вычислимости рассмотрены на примерах задачи классификации плотности, проблем залпового огня и выбора королевы роя, а также алгоритмов сортировки и алгоритма параллельного умножения Атрубина.

Ключевые слова: клеточные автоматы, вычислимость, сигнал, сортировка, параллельное умножение, алгоритм Атрубина, машина Тьюринга, времяконструируемость.

Введение

Экстенсивный путь развития компьютеров, связанный с непрерывным повышением тактовой частоты и уменьшением проектной нормы интегральных микросхем, исчерпал себя, столкнувшись с экспоненциальным возрастанием тепловыделения. Одним из перспективных направлений развития архитектур для неклассических вычислений являются клеточные автоматы, в которых естественным образом реализуется параллелизм обработки информации. Поэтому обсуждение имеющихся подходов к вычислимости в клеточных автоматах (КА) представляется актуальной задачей, которой и посвящен данный аналитический обзор. В силу громадного количества публикаций по КА, в той или иной степени затрагивающих проблему вычислимости (например, работа [1], содержащая 117 ссылок на литературу), он не претендует на полноту. Так, например, исключены из рассмотрения квантовые клеточные автоматы, где функция переходов и состояние ячейки имеет принципиально квантово-вероятностный характер.

Предварительные замечания

Под вычислимостью (computability) мы будем понимать совокупность правил конфигурирования вычислителя определенного рода для реализации конкретных алгоритмов, независимо от того, подобраны ли эти правила эмпирически или сформулированы в виде общих теорем существования. В нашем случае вычислитель имеет структуру КА, причем сложность его ячейки и локальной функции перехода (ЛФП) мы заранее фиксировать не будем. Режим работы КА будет предполагаться синхронным, что обеспечивает высокую степень параллелизма вычислений (computation=computing).

Данное определение предполагает довольно широкую свободу интерпретации. При более строгом подходе вычислимость эквивалентна или, во всяком случае, должна включать в себя построенную модель вычислений и указание на область алгоритмов, реализуемых с помощью данной модели. В Википедии принято узкое, слишком инженерное определение: «… теория вычислимости и теория сложности вычислений трактует модель вычисления (model of computa-

[1] Работа выполнена в рамках программы фундаментальных исследований ОНИТ РАН, НИР «Исследование перспективных моделей вычислений и реализующих их архитектур высокопроизводительных информационно-вычислительных комплексов нового поколения».

tion) не только как определение множества допустимых операций, использованных для вычисления, но также и относительных издержек их применения. Охарактеризовать необходимые вычислительные ресурсы — время выполнения, объём памяти, а также ограничения алгоритмов или компьютера — можно только в том случае, если выбрана определённая модель вычислений». По нашему мнению, удовлетворительным неформальным определением можно считать следующее: «Модель вычислений – это совокупность аппаратно-программных средств и схема их взаимодействия между собой и пользователями, определяющая исполнение алгоритмов» [2]. Понятие вычислимости столь же изначальное, как и функция или алгоритм, не в последнюю очередь потому, что размыта степень подробности задания вычислителя и его исполнительных механизмов. Соответственно и нет общей трактовки модели вычислений, хотя её представлений известно несколько [3,4]: машина Тьюринга, машина с неограниченным числом регистров, равнодоступная адресная машина, графовая модель «операция-операнды», потоковые модели, модель акторов и т.д. Неплохая попытка систематизировать классические и неклассические модели вычислений сделана, исходя из педагогических целей, Т.С. Стефановой [5]. Важный для КА аспект недетерминированности вычислений и понятие «вычислимости» анализируются логико-философскими средствами А.М. Анисовым [6]. Весьма близкое к нашему понимание модели вычислений демонстрирует автор [7], однако в нашем обзоре вычислимость будет трактоваться при более общих предположениях относительно КА-вычислителя.

Применительно к клеточным автоматам целесообразно выделять вычислимость универсальную и специальную. Универсальная вычислимость для некоторого КА означает, что доказана его эквивалентность некоторой (быть может, даже универсальной) машине Тьюринга, что в силу тезиса Тьюринга-Черча означает реализуемость любой вычислимой функции с его помощью. При этом построение такого КА и тем более техническая реализация могут оказаться с практической точки зрения громоздкими и неэффективными. Под специальной вычислимостью мы понимаем наличие конструктивных клеточно-автоматных решений для ряда типичных задач (сортировка, вычисление значений полинома, быстрое преобразование Фурье, шифрование, и т.п.). Обращение к специальной вычислимости связано также с тем, что уже несколько десятилетий КА успешно используются в моделировании диффузии, химической кинетики, процессов самоорганизации и пр., т.е. сама вычислительная практика апостериори показывает пригодность КА как инструмента для вычислений.

Из существующих моделей вычислений наиболее близкой к КА является модель акторов [8]. Эта близость обусловлена тем фактом, что КА представляют собой наиболее простую имплементацию парадигмы коннекционизма (Рис. 1).

Лозунг радикального коннекционизма – «связи – все, элементы – ничто», т.е. для результата вычислений более значимы связи, а не особенности структуры элементов, чем бы те не были. Историческая сводка по коннекционизму (или, connectivism=connectionism) дана в [9], а его логические принципы, взятые через призму PDP (parallel distributed processing), даны в [10]. Практически все нынешние неклассические направления компьютинга вполне укладываются в коннекционистскую доктрину. Рассматривая вычислимость в КА, нельзя в той или иной степени не принимать во внимание вопросы вычислимости в коннекционизме. Так, например, наличие флага активности процессорного элемента, свойственного PDP и нейронным сетям, выглядит крайне удобным при конструировании КА-алгоритмов и важно для специальной КА-вычислимости.

Любое сколь угодно сложное поведение системы может быть объяснено, выведено из простых правил, которым подчиняются её элементы. Такова основная мысль капитальной работы Стивена Вольфрама [11], вполне лежащая в русле коннекционизма. Движение к максимальному распараллеливанию есть одновременно движение к увеличению числа и упрощению элементов при усложнении связей. По нашему мнению, основной вопрос коннекционизма с точки зрения эффективности вычислений заключается в том, где следует остановиться в этом движении? Для КА крайне неполный, но все-таки ответ, дается введением параметра Лэнгтона [12, 13], в какой-то степени помогающего конструировать КА с заданными глобальными поведенческими свойствами.

Рис.1. Образы коннекционизма в современной вычислительной технике

Уместно также дать формальное определение классического КА (подчеркнем, что именно «классического», поскольку в КА-моделях часто происходит отказ от или детализация тех или иных свойств), удовлетворяющего требованиям (Рис. 1):

- *однородности элементов* (т.е. каждый элемент/ячейка устроен одинаково, прежде всего по структуре своего состояния и предписанных ему локальных правил перехода – в этой связи теория КА называется иначе теорией гомогенных структур [14, 15]);
- *локальности связей* (т.е. выбирается способ индексирования или именования ячеек КА и постулируется универсальность выбора соседей для каждой ячейки, кроме, быть может, граничных; по сути выбирается топология КА, и в самом простом случае мы говорим об 1D, 2D, 3D-автоматах и ради удобства восприятия ассоциируем каждую ячейку с узлом решетки);
- *синхронности работы* (т.е. будущее состояние ячейки зависит от прошлых состояний самой ячейки и её соседей, что автоматически означает одновременное выполнение локальных функций перехода по всему полю КА; тем не менее, известно множество асинхронных КА-моделей)
- *дискретности состояния* (это свойство наследовано из определения конечного автомата; тем не менее, в КА-моделях часто используются непрерывные состояния, тогда говорят о КА ОДУ (Continuous-Valued CA, в зарубежной терминологии), т.к. тогда структура КА наследуется из системы обыкновенных дифференциальных уравнений).

Для универсальной вычислимости постулируется бесконечность поля КА (аналог – бесконечность ленты в машине Тьюринга). Определению классического КА отвечают «Игра Жизнь» (КА Конвэя, Game of Life = GoL) и все *элементарные* КА, детально исследованные С. Вольфрамом [11]. Элементарный КА по определению имеет множество состояний $\{0,1\}$, является одномерным, причем шаблон окрестности имеет радиус $r=1$ (т.е. в окрестности содержатся сама ячейка, её левый и правый соседи), обладает детерминированностью и синхронностью правил перехода, неограниченностью поля. Всего возможно 256 элементарных КА по вольфрамовской нумерации, кодирующей функцию перехода ($2\equiv\{0,1\}, F=\{f:2^3\to 2\}, |F|=256$) последовательной записью её значения в каждом бите (Табл. 1.).

За формальное определение классического КА примем взятое из теории однородных структур [14-16]. Пусть S – конечное множество (например, алфавит), $\mathbb{Z}^d$ – бесконечная *d*-мерная целочисленная решетка. (Глобальной) конфигурацией КА назовем отображение решетки на множество состояний $c:\mathbb{Z}^d\to S$ и введем множество всех конфигураций

Табл. 1. ЛФП элементарного КА для правил 30 и 184

Текущее состояние окрестности	1**1**1	1**1**0	1**0**1	1**0**0	0**1**1	0**1**0	0**0**1	0**0**0
Будущее состояние по правилу $30=(11110)_2$	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**
Будущее состояние ячейки по правилу $184=(10111000)_2$	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**

Примечание. Полужирным шрифтом выделено состояние центральной (активной) ячейки.

$C(d,S)\equiv S^{\mathbb{Z}^d}$. Шаблон окрестности представляется кортежем *d*-мерных векторов $N=\langle \mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\rangle, \mathbf{x}_i\in\mathbb{Z}^d, i=\overline{1,n}$. Локальной функцией перехода (local rule) называется $f:S^n\to S$. За один ход симуляции происходит изменение конфигурации $c\to e\Leftrightarrow \left(\forall \mathbf{x}\in\mathbb{Z}^d\right)\left(e(\mathbf{x})=f\left(\mathbf{x}+\mathbf{x}_1,\ldots,\mathbf{x}+\mathbf{x}_n\right)\right)$, и такое изменение порождает глобальную функцию перехода (global transition function) $e=G(c), G:C\to C$. Таким образом, классический КА определяется тройкой $\langle S,N,f\rangle$, не считая *d* или решетки $\mathbb{Z}^d$, а также дискретного множества моментов времени *T*.

При допущении асинхронности КА необходимо доопределить порядок исполнения ЛФП, т.е. режим КА-симуляции, и тогда имеем четверку $\langle S,N,f,\tau\rangle$. Если допустить неклассическое задание топологии, например, гиперболическое пространство [17, 18], то целесообразно вывести аналог решетки $X\sim\mathbb{Z}^d$ (точнее, множество узлов) в отдельное определение и заменить шаблон окрестности на функцию именования. КА также можно трактовать как массив/матрицу конечных автоматов, связанных входами/выходами.

С содержательной точки зрения (при почти одинаковом формализме) семантика КА, по нашей терминологии, эксплицируется в трех подходах:

- Традиция Неймана (клеткам соответствуют фиксированные физические объекты, состояния смежных клеток определяют новое состояние клетки).
- Традиция Цузе (клетка есть математическая абстракция – конечный объем, виртуальный резервуар для мигрирующих и взаимодействующих между собой физических частиц).
- Традиция Цетлина (клетки-индивидуумы могут путешествовать по физическому пространству, т.е. координаты клетки «вкладываются» в состояние ячейки и, следовательно, менять структуру своих связей; КА есть мультиагентная система).

Специальная вычислимость

Удобнее начать обсуждение специальной вычислимости на отдельных примерах, хотя исторически более правильным было бы рассматривать вначале универсальную вычислимость. Однако, отдавая дань истории, начнем с трех классических задач КА-вычислимости, достаточно сложных, и затем разберем уже более простые.

Задача классификации плотности (Density Classification Task, DCT). Бинарный КА должен определить для своего произвольного стартового состояния: больше в нем нулей или единиц. Например, в качестве ответа, КА должен заполнить все поле нулями (единицами) и остановиться. Дополнительным условием и критерием сравнения эффективности алгоритма можно выставить максимальное число итераций до достижения финального состояния. Трудность задачи состоит в том, что отдельная клетка не знает всей глобальной конфигурации.

Элементарный КА не может решить эту задачу. Marques-Pita и др. [19] попытались найти общие черты в формулах перехода девяти одномерных КА, решающих DCT. В частности, широко известно [20] классическое решение для радиуса *r*=3. Если допустить память у элементарного КА, то решение DCT возможно [21, 22] для КА правил 184 и 226; остановимся, следуя [21], на этом подробнее.

Рассматривался элементарный КА с периодичными условиями и содержащий *N*=149 ячеек. Интуитивно ясно, чем больше нулей/единиц в начальном состоянии (IC), т.е. процент заполненности $\rho\to 0/1$, тем КА быстрее сходится к решению, и, наоборот, при $\rho\approx 0.5$ сходимость тяжелая. При статистических испытаниях IC и при равновероятных 0/1 для одной ячейки распределение $\rho(0)$ биномиальное, а не равно-

мерное. Для вычисления точности КА-симуляции бралось 10^4 IC и максимальное число итераций I=300. Для известного КА с $r = 3$ точность вычисления (число удач к числу запусков) была в пределах 65÷86%.

Простейший путь введения памяти в КА – это увеличение арности ЛФП и удвоение числа состояний:

$$\eta_i = \left\{\sigma_{i-r},...,\sigma_{i-1},\sigma_i,\sigma_{i+1},...,\sigma_{i+r}\right\},$$

$$\text{ЛФП: } \sigma_i^{t+1} = \Phi\left(\sigma_j^t \in \eta_i, \sigma_i^{t-1}\right).$$

Здесь σ_i^t – состояние i-й ячейки КА в момент t, η_i – состояние её окрестности. Авторы [22] предложили красивое решение DCT, не затрагивающую арность ЛФП и опирающееся в простейшем случае на ЛФП такого вида (majority–вентиль/оператор голосования большинства):

$$s_i^t = majority\left(\sigma_i^{t-2}, \sigma_i^{t-1}, \sigma_i^t\right), \quad \sigma_i^{t+1} = \Phi\left(s_j^t \in \eta_i\right).$$

Эффективность решения доказывается тем, что для 10^5 IC (распределение $\rho(t=0)$ бралось равномерное) лишь 114 конфигураций не были классифицированы ЛФП на основе (Ф) правила 184, все они лежали вблизи $\rho \approx 0.5$ (N=400, I=1500).

В более совершенном виде эта схема вводит фактор обучения $0 < \beta < \frac{1}{2}$ и добавляет к битовой компоненте состояния ячейки вещественную составляющую, играющую роль памяти:

$$State = \begin{pmatrix} m \\ \sigma \end{pmatrix}, \begin{cases} m_i^0 = 0.5 \\ m_i^{t+1} = m_i^t + \beta\left(\sigma_i^t - m_i^t\right) \end{cases},$$

$$s_i^t = \begin{cases} 0 : m_i^{t+1} \le 0.5 \\ 1 : m_i^{t+1} > 0.5 \end{cases}, \sigma_i^{t+1} = \Phi\left(s_j^t \in \eta_i\right).$$

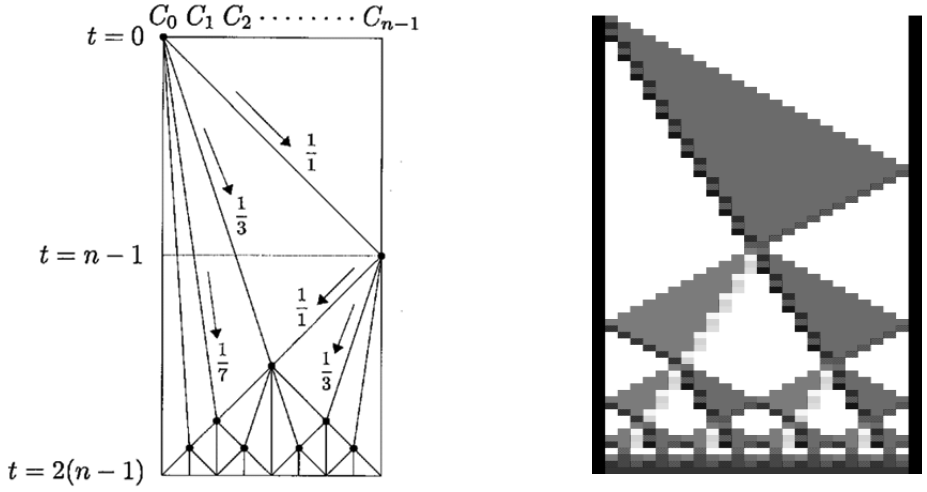
Среди 256 элементарных КА, симулированных в этой схеме, наибольшая точность (65%, $\beta \in [0.45, 0.49]$) у КА правил 184 и 226. Если увеличить глубину итераций с 300 до 900, то точность возрастает до 80% и выше, причем среднее число итераций до достижения верного решения было 220. Правила 184 (Табл. 1.) и 226 переходят одно в другое, если зеркально отразить соседей относительно центральной ячейки. Если посмотреть на таблицу правила 184, то она в терминах частиц примерно отвечает движение справа налево (скажем, комбинация 110 дает 0 как 0, пришедший справа). Само по себе, без памяти, правило 184 консервативно, сохраняя количество нулей/единиц.

Известен бинарный автомат Гакса-Курдюмова-Левина [23], решающий DCT с точностью 76%. В 1995г. Land и Belew [24] доказали, что не существует бинарного КА с радиусом r, 100%- решающего DCT для IC с плотностью $\rho(0)$, если общее число ячеек достаточно велико, а именно: $N > 4r/\rho(0)$.

Таким образом, КА-вычислимость на примере DCT обнаруживает сходство постановки задачи (бинарная классификация как стандартный тест для перцептронов) и поведения КА и нейросетей в отношении неполноты решения массовой алгоритмической проблемы, а также возможность бильярдной интерпретации наиболее эффективных ЛФП. Поиск же эффективных ЛФП возможен с помощью генетических алгоритмов и представлений эволюционного моделирования [20, 25], когда таблица ЛФП выступает в качестве хромосомы. Это замечание справедливо и для КА с памятью, каким бы способом она ни вводилась.

Залповый огонь (Fire Squad, Firing Squad Synchronization Problem = FSSP). В оригинальном описании (Myhill, 1957) задачи N солдат (один из них – генерал) становятся в одну шеренгу. Каждый солдат может общаться лишь с правым или левым соседом. Генерал даёт команду «пли!». Через некоторое время каждый солдат и генерал должны выстрелить одновременно и при этом каждый – в первый раз. Как видно, данная задача – это задача синхронизации (более формально, все ячейки КА должны прийти в особое состояние «пли!»). Она важна в различных прикладных областях; например, её решение понадобилось при разработке семисегментного дисплея часов [26]. Было предложено и изучено несколько более общих формулировок задачи, включая ряды с произвольным расположением командира, замкнутые кольца, квадраты, кубы [27] и другие топологии. Интерес представляет минимизация числа возможных состояний элементов, числа возможных ситуаций для элемента (состояние самого элемента и его соседей) и количества временных интервалов до залпа.

Следуя [28], в 1966г. Waksman опубликовал решение одномерной задачи для 16 состояний элементов. В 1967г. Balzer уменьшил это число до 8. В 1987г. Mazoyer нашёл решение этой



(a) (b)

Рис.2. Схема КА-решения FSSP на диаграмме время-пространство

(a) сложность алгоритма 2(n-1). Солдаты пронумерованы от 0 (генерал) до (n-1);
(b) сложность алгоритма 3n, используется 15 состояний.

задачи для 6 используемых состояний. Оценку снизу получил Balzer. Он доказал, что решение поставленной задачи невозможно для клеточных автоматов с 4 и менее используемыми состояниями. Также показано, что до залпа требуется не менее $2(N-1)$ шагов. Основная идея состоит в том, что генерал испускает возбуждения с разной частотой вида $\left(2^k-1\right)^{-1}$, а эти волны, отражаясь от границы поля КА (солдат «становится» генералом), интерферируют и на каком-то шаге все солдаты оказываются возбужденными (Рис. 2). Схема алгоритма с минимальным числом состояний (6) приведена Мазоэром в [29].

В ходе решения FSSP помимо регуляризации техники выделения подмножеств из множества состояний КА, использованной еще фон Нейманом, было, начиная с работ Минского, осознано удобство введения в практику конструирования КА термина/категории «сигнал» [30, 31]. Сигнал для одномерного КА трактуется как линия на диаграмме время-пространство, т.е. как унарное отношение, заданное на множестве $C \times T$. Как оказывается, это понятие тесно связано с время-конструируемостью функций [32], в частности, функциями Фишера [33].

Задача нахождения королевы роя (distributive choice problem, leader election problem, Queen Bee Problem = QBP). Прикладное значение задачи связано с оптимальным выбором конфигурации сети [34-36]. Процессоры в распределенной сети, каждый из которых выполняет один и тот же алгоритм, обязаны избрать уникальный процессор (лидер). Цель лидера – нарушить симметрию системы путем выбора процессора, который будет координировать глобальную деятельность. Выбрав лидера, можно выполнить централизованный протокол в децентрализованной среде. Присутствие лидера препятствует возникновению нестабильностей в вычислительной сети (свойство устойчивости – self-stability). Одномерная QBP часто решается на топологии кольца или даже обобщается на топологию тора. Плоская двумерная задача (Smith, 1971) или задача на топологии графа более сложна, а её решение более актуально. Биологическая терминология связана с коллективным поведением животных/насекомых, и авторы [34] широко её использовали при выборе шлюза для беспроводной сенсорной сети с помощью КА-алгоритма (автомат имел 8 состояний).

Следуя формулировке планарной QBP [36], мы имеем КА с более чем тремя состояниями: черным – «смерти», белым – «солдата», красным – «генерала/королевы». Начальное состояние представляет собой белую фигуру-паттерн на черном поле. Те, клетки, что отмечены черным, не меняют своего состояния в дальнейшем, а фигура-паттерн является связной и конечной по периметру, т.е. для каждой её ячейки

найдется по крайней мере еще одна лежащая в её окрестности ячейка фигуры-паттерна. На паттерн помимо условий связности и финитности можно наложить дополнительные условия «отсутствия дыр» и «достаточной толщины». За конечное время КА должен переработать паттерн так, чтобы только одна его ячейка была красной и до этого момента красных ячеек не появлялось. Требование одинаковости стартовых состояний паттерна существенно. Легко обобщить формулировку на *d*-мерный КА (с радиусом *r*=1). Таким образом, задачи QBP и FSSP являются взаимообратными.

Если алгоритмы для 2D QBP имеют временную сложность порядка O(*n*), где *n* – число ячеек паттерна, то алгоритм Хеммерлинга (1979) для *d*-мерный задачи имеет сложность $O(n^5)$. Описанный в [36] алгоритм сходится за (*b*+2) шагов, где *b* – длина границы паттерна. Ничитиу и др. [37] в 2001г. улучшили результат Хеммерлинга до второго порядка $\sim 4\frac{\omega(\omega+1)}{2}$, где ω – длина наибольшего из кратчайших путей, ведущих от королевы до границы паттерна в $\mathbb{Z}^d$. Описание их алгоритма достаточно сложно, здесь лишь приведем граф состояний конечного автомата, служащего ячейкой КА (Рис. 3).

Хотя клетки паттерна идентичны, окрестность ячейки может иметь или не иметь «представление о сторонах света» (compass), т.е. соседи могут с учетом геометрии поля быть упорядочены (допустим, лексикографическим порядком). Разумеется, решить QBP во втором случае сложнее. Общая идея состоит в выделении локальных лидеров (кандидатов), каждый из которых испускает сигналы.

Что дало решение QBP для вычислимости? Во-первых, более глубокое понимание сигналов; во-вторых, техника выделения подмножеств из множества состояний получила развитие/оформление в таком привычном для разработчиков процессоров понятии, как регистр. Например, авторы [36] задействовали 6 регистров, из которых два были основными (первый хранил собственно данные, второй – сигнал), а 4 – обслуживающими. В нашей работе [38] мы писали о регистрах как о компонентах состояния и выделяли, по крайней мере, два типа – собственное (данные) и активное (флаги, см. ниже).

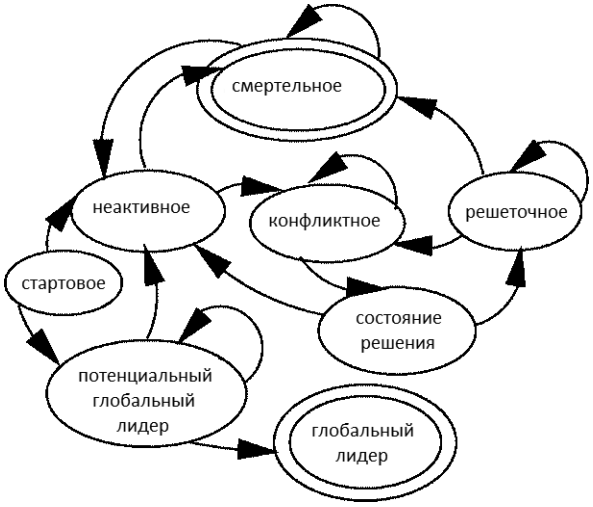


Рис.3. Редуцированная схема переходов между состояниями конечного автомата, решающего QBP

Реализация умножения. Алгоритм Атрубина. Если бы мы не имели параллельных алгоритмов хотя бы умножения натуральных чисел, это было бы скандалом для практического аспекта вычислимости в КА. Многое зависит и от вида представления данных-чисел (например, умеет ли ячейка выполнять умножение десятичных цифр или же ЛФП ориентирована на двоичный код), и сложность задачи легко недооценить. Современных работ на эту тему, к сожалению, очень мало [39]. Алгоритм, предложенный [40] еще в 1965г. Алланом Атрубиным (когда КА назывались итеративными массивами), описан в неординарной работе [30] с ординарным названием. Машина Тьюринга осуществляет перемножение чисел *a* и *b* за $\log(a)\cdot\log(b)=m\cdot n$ ходов (*m*, *n* – число разрядов в каждом числе), а сложность алгоритма Атрубина равна (*m*+*n*). Алгоритм может быть обобщен на произвольные системы исчисления, обычно упоминается в контексте систолического процессорного массива (поэтому абстракция КА не вполне подходит) и до конца [41] еще не исследован.

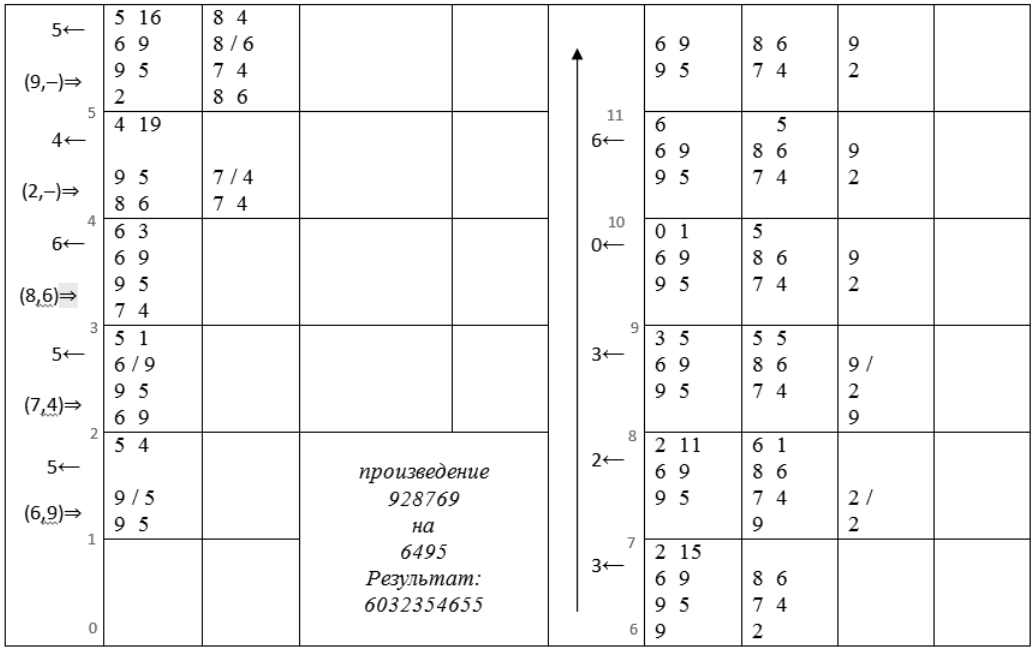
Пусть требуется умножить два числа: $A=\sum_{i=0}^{n}a(i)2^i$, $B=\sum_{i=0}^{m}b(i)2^i$, $n\geq m$. Слева на вход КА подаются пары двоичных цифр, начиная с младшего разряда: при $t=i$ $g(i)=(a_i,b_i)$. Каждая ячейка имеет пять регистров: три из них хранят значения проходящих пар (или знак состояния покоя Q, Рис. 4), четвертый хранит

g(12)	*g(11)*	*g(10)*	*g(9)*	*g(8)*	*Q*	*Q*	*t*=12
g(1)	*g(3)*	*g(5)*	*g(7)*	*Q*	*Q*	*Q*	
g(0)	*g(2)*	*g(4)*	*g(6)*	*Q*	*Q*	*Q*	

(а)

Текущее состояние	Текущее состояние левой ячейки	
	(–, Q, –)	*(h, k, –)*
(Q, Q, Q)	*(Q, Q, Q)*	*(h, Q, Q)*
(g, Q, Q)		*(h, Q, g)*
(j, Q, g)		*(h, j, g)*
(m, j, g)		*(h, j, g)*

(б)



(с)

Рис.4. Мультипликатор Атрубина

(а) Заполнение трех регистров на 12-м ходу [40].
(б) Таблица переходов первых трех регистров (one-way КА – ячейка + ячейка слева). Латинскими буквами отмечены пары битов.
(с) Пример умножения двух десятичных чисел в мультипликаторе Атрубина. В скобках показаны пары, вводимые в КА в соответствующие моменты времени. Исходящие стрелки показывают выходные сигналы и одновременно содержимое регистра [30]. Оказывается достаточным 3-х ячеек КА и 12 шагов.

частичное произведение (а именно: бит ответа), а пятый – полное произведение, принимая значение от 0 до 4. Общая идея состоит в том, что по мере движения пар вправо одновременно происходит вычисление частичных произведений $a(i)\times b(j)$, их суммирование и деление по модулю 2.

Отметим, что при реализации алгоритма оказалось удобным ввести состояние покоя, т.е. $q \in S : f\left(q,q,...,q\right) = q$.

Связь с дифференциальными уравнениями. С одной стороны, обыкновенные дифференциальные уравнения (ODE) и уравнения в частных производных (PDE) не всегда являются релевантным инструментом моделирования сложных систем, например в области биологии. Указываются такие причины: сглаживание локальных флуктуаций, пренебрежение корреляциями между движениями разного сорта, запаздывание между причинами и наступлением следствия (однако есть DDE, уравнения с запаздыванием). КА позволяют лучше учесть эти особенности. С другой стороны, КА с непрерывными значениями могут рассматриваться как численный метод решения задач математической физики [42], обладающий высокой

эффективностью при распараллеливании вычислений. В частности, КА-средствами были промоделированы ряд систем типа «диффузия-реакция», реакция Белоусова-Жаботинского, а также перколяционные физические системы и магнитные системы, подчиняющиеся модели Изинга. На эту тему написано множество работ, особенно в конце 90-х гг. [43, 44].

Для понимания КА ОДУ, как названы КА с непрерывными значениями в известной статье [45], приведем КА-выражение для двумерного оператора Лапласа в контексте уравнения диффузии и разностной схемы «крест» ($c_{i,j}(t)$ – концентрация вещества в узле (i,j) в текущий момент времени t):

$$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}\right) \to \left(\partial t = \tau, \partial x = \partial y = h\right) \to$$

$$c_{i,j}(t+1) = \frac{D\tau}{h^2}\sum c_{i\pm1,j\pm1}(t) + \left(1 - \frac{4D\tau}{h^2}\right)c_{i,j}(t)$$

КА с непрерывными значениями отождествляются [46] с решетками связных отображений (Coupled Map Lattices), которые в настоящее время являются популярным объектом для исследования со стороны специалистов по итерированным отображениям и фракталам.

Решеточный газ: модели движения и столкновения частиц. В физике решеточным газом называют совокупность N частиц, движущихся в большом объеме, разделенном на M>>N малых объемов так, что в одном малом объеме не может быть больше, как правило, одной частицы. КА-симуляции динамики жидкости или газа основаны, что характерно для традиции Цузе, на сохранении количества частиц в ячейках КА и имплементации взаимодействий частиц посредством ЛФП. Рассмотрим, например, простейшую модель решеточного газа, называемую HPP по первым буквам имен её создателей (Hardy, Pomeau и de Pazzis). В двумерном КА каждая ячейка может хранить до 4-х движущихся частиц. Каждая частица имеет 4 направления движения (влево, вправо, вниз, вверх). В ячейке не может быть больше одной частицы с одним направлением движения. Таким образом, ячейка КА имеет 16 возможных состояний. ЛФП в модели HPP сохраняют общее число частиц, их общий импульс, однако, не все законы сохранения выполняются. Принципиально важен момент обратимости КА-HPP: инверсный КА имеет те же правила, но противоположные направления скоростей в начальной конфигурации.

Алгоритм сортировки (символов). Простейшее КА-решение, использующее блочный механизм Марголуса, основано на идее «всплытия» пузырьком. Сразу заметим, что более развитые методы сортировки, стремясь к минимизации прохода по массиву, в той или мере используют глобальную информацию, что делает их сомнительными для использования в КА. Для простоты примем множество состояний $S = \{0,1,\ldots,6\}$ – цвета радуги. На нечетных ходах, т.е. $t \bmod 2 \equiv 1$, поле КА разбивается на блоки по 2 ячейки каждый. На четных ходах разбиение смещается на 1 ячейку, например, вправо. ЛФП относится к блоку, описывается правилом «если цвета ячеек стоят в неправильном порядке, то поменять их» и может быть записано как:

$$b \in \{0,1\} \equiv s_i^t < s_{i+1}^t, \begin{pmatrix} s_i^{t+1} \\ s_{i+1}^{t+1} \end{pmatrix} = \begin{pmatrix} b & \bar{b} \\ \bar{b} & b \end{pmatrix}\begin{pmatrix} s_i^t \\ s_{i+1}^t \end{pmatrix}.$$

Этот механизм, что важно, допускает переформулировку для классического КА. Для этого нужно доопределить множество состояний ячейки на $\langle\{0,1\},\{0,1\},S\rangle = \langle$флаг времени, флаг блока, символ$\rangle$, взять симметрично r=1, создать ЛФП на флаги. В зависимости от флагов ячейка будет игнорировать своего левого или правого соседа. При оптимизации флаг времени можно исключить (Приложение).

На Рис. 5 показана ST-диаграмма (space-time) для такого КА, сортирующего массив символов. Когда состояние автомата перестанет изменяться, тогда наступает останов. При строгом понимании вычислимости нужно технически обеспечить это требование, что на самом деле может представлять неожиданную сложность. Сложность сортировки пузырьком, как известно, составляет $O(N^2)$. Описанный выше КА дает эмпирически $O(N)$, точнее даже $0.8N$ (Рис. 5) при $N <\sim 100$. Примечательно, что для наихудшего случая, когда наименьший элемент находится в конце поля, за время его миграции почти полностью завершаются все перестройки. На Рис. 5 легко различимы траектории движения (всплытия) подобных элементов. Нужно осознавать, что в отличие от классической модели вычислений мы не можем использовать

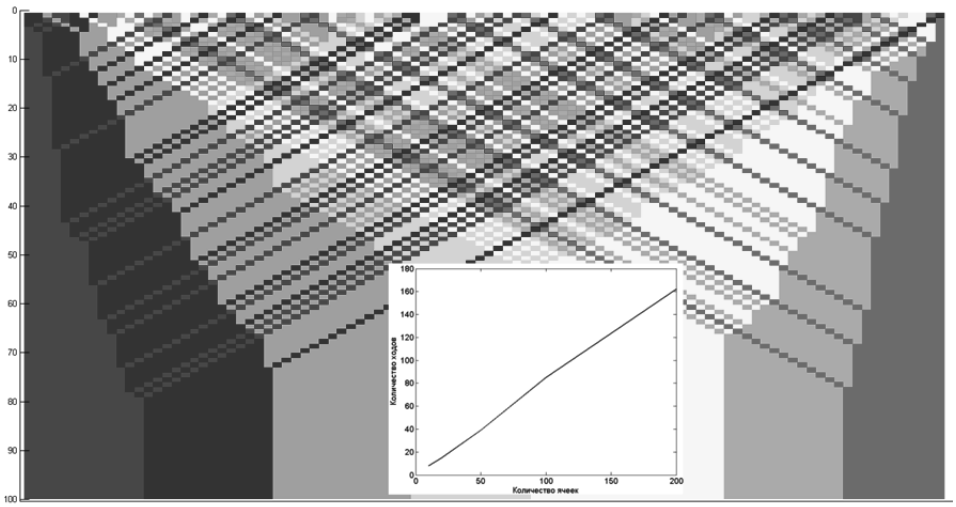


Рис.5. Диаграмма «время-пространство» (ось времени направлена вниз) для КА сортировки

Поле КА есть массив из N=100 ячеек, цвет отражает состояние ячейки (7 значений). Начальное состояние случайное. В диаграмму вклеена эмпирическая зависимость числа ходов до останова от размера поля N.

центральный процессорный элемент, мы лишены возможности переносить данные между удаленными регистрами, и потому феномен таких «движений» (с целью переноса данных) суть «родимое пятно» вычислимости в КА.

КА-реализация сортировки символьных строк потребует 2D-КА и, вероятно, добавления флагов и организации «движений», которые требуют сопряжения локальных сравнений символов по лексикографическому порядку и учета удаленной по полю КА информации. Заметим, что мы не нашли в литературе описания подобного алгоритма, если не учитывать решение [47], носящее скорее hardware-характер. Поэтому соответствующий псевдокод мы вынесли в Приложение.

Вычисление биномиальных коэффициентов. Естественное решение дается т.н. on-way автоматом, т.е. 1D-КА с радиусом, кратным $1/2$, подразумевающим асимметрию шаблона окрестности. В данном случае это ячейка и её правый сосед. Для граничной, самой правой ячейки, состояние неизменно. ЛФП задается формулой $s_i^{t+1} = s_i^t + s_{i+1}^t$, где s_i^t - состояние i-й ячейки в момент t. КА имеет N ячеек, а его начальная конфигурация задается как $s_{i<N}^0 = 0, s_{i=N}^0 = 1$. Нетрудно видеть, что реализуется треугольник Паскаля на ST-диаграмме, при этом биномиальные коэффициенты находятся сразу группой. Однако у этого элегантного решения есть два изъяна с точки зрения вычислимости: 1) недостаточная определенность в мощности множества S и нужном числе ячеек КА; 2) не прописано/реализовано условие останова.

Генерация целочисленных последовательностей. Одно из применений КА состоит в генерации случайных чисел, т.к. многие ЛФП приводят к хаотическому изменению конфигураций. Однако некоторые КА, прежде всего элементарные, порождают упорядоченность. Например, КА правила 50 с начальным состоянием (…,0,0,1,0,0…) последовательно генерирует числа 1,5,21,85,… (последовательность A002450 из Банка - oeis.org), удовлетворяющих формуле $\left(4^{t+1}-1\right)/3$. Предполагается, что каждая строка на ST-диаграмме есть двоичный код некоторого натурального числа.

Алгоритмы компрессии и криптографии данных. Мы ограничимся лишь указанием некоторых работ по этой обширной тематике. В таких задачах математическим аппаратом часто является конволюция (или свертка) функций и быстрое преобразование Фурье (FFT). Связь FFT и КА обсуждается в [48-50], причем в последней публикации поле КА рассматривалась основой для создания некой специализированной сети FFT-вычисления. В [48] вводится общее понятие клеточно-автоматного преобразования (Cellular Automata Transforms, CAT), пригодное помимо FFT для преобразований Лапласа, вейвлет и т.д., и выделяется 8 типов

соответствующих автоматов. В [49] дана схема реализации FFT и более ортогонального преобразования через пошаговое взаимодействие двух КА: тоталистичного и реверсивного.

Заметим, что требование реверсивности КА существенно в задачах криптографии и кодирования информации [51]; при этом важно учитывать энтропийный аспект КА-симуляции.

Распознавание языков с помощью КА. Уже давно известны применения как конечных, так и клеточных автоматов к языковым грамматикам и системам. Отметим, например, нетривиальное применение языковых КА к моделированию графена. В 70-х гг. появились работы по грамматикам с формой (Shape Grammars), т.е. правилам генерации двумерных или трехмерных символьных объектов, что являлось прогрессом нашего представления о строках как о линейных массивах символов. В работах Т. Шпилера [52, 53] на основе подхода (SG→CA approach) спроектированы графеновые конструкции.

КА в урбанистике и геодезии (геоинформационных системах). В сфере планирования городского развития, прежде всего транспортных путей, КА проявили себя особенно ярко и мощно. Объект исследования наложил отпечаток и на структуру используемых КА. Простейшим КА, симулирующим движение, является автомат правила 226. В более сложных, т.н. 2D constrained CA, используется окрестность радиуса 6 со 113 соседями, а в ЛФП для каждого соседа вначале оценивается «потенциал перехода», затем происходит сортировка клеток по его возрастанию и ячейки с наибольшим потенциалом участвуют в дальнейших вычислениях [54].

Выбор ЛФП. Хотя КА требуют однородности, но неоднородности в них могут быть введены формально однородными средствами. С точки зрения опыта разработки микропроцессоров очень важно, чтобы одна ячейка КА «умела» выполнять не одну фиксированную ЛФП (операцию), а обладала диапазоном «умений» и могла выбирать, какую ЛФП из набора она будет выполнять в текущий момент. Вероятностные и т.н. альтернирующие КА удовлетворяют этому полезному свойству. В простейшем случае при выполнении некоторого условия ЛФП производится, а при невыполнении – ничего не происходит. Таким условием может быть розыгрыш равнораспределенной случайной величины ξ на отрезке [0;1], и если испытание $\xi \in [0;\omega], 0 < \omega < 1$, то ЛФП выполняется. В КА с альтернацией таким условием выступает принятие некоторого значения специальной компонентой состояния ячейки. Мы использовали эту схему ранее при конструировании класса КА с фиксацией. [55].

КА с альтернацией применялись для распознавания языков; в таком контексте уместно привести наброски определения, следуя [56, 57]. КА с альтернацией – это совокупность $\langle S, \delta, \#, A, F \rangle$, где: 1) S – конечное непустое множество состояний, разбитое на две части: существенную и универсальную; 2) $\# \notin S$ - состояние границы (удобный способ маркировки граничных ячеек, не вводя для них особую ЛФП, но расширяя ординарную ЛФП); 3) $A \subseteq S$ - непустое множество входных символов; 4) $F \subseteq S$ - множество разрешенных состояний; 5) δ – конечное непустое множество ЛФП вида $f : (S \cup \#)^3 \to S$.

Универсальная вычислимость

Еще на самой заре истории КА вопросы универсальной вычислимости, трактуемой через призму машины Тьюринга, стояли в фокусе исследований и были вполне в общем русле алгоритмической тематики. В частности, давался вопрос о существовании самовоспроизводящихся конфигураций в поле КА, о возможности создания универсального КА, эмулирующего работу любого КА при заданной начальной конфигурации. На эти вопросы были даны определенные ответы, предложены конструкции КА, но все-таки это мало что дает для практических вопросов вычислимости.

В вопросе универсальности важную роль имеет теорема из теории машин Тьюринга (МТ), утверждающая универсальность МТ с числом состояний ленты 4 и состояний головки 7 (или в комбинации 6 и 6). Известно, что КА «Игра Жизнь» представляет собой универсальный компьютер, т.е. для любой заданной МТ и входного слова W можно сконструировать начальную конфигурацию GoL такую, что все клетки в конце концов умирают если и только если МТ принимает (разрешает) W.

Однако существуют и более сильные теоремы. Например, в 1987г. Дэн Гордон доказал конструктивным путем [58], что:

– работа машины Тьюринга с алфавитом ленты мощности *m* и алфавитом состояний мощности *n* может быть эмулирована одномерным, тоталистичным КА (т.е. новое состояние ячейки зависит только от суммы состояний соседей) с шаблоном окрестности три (*r*=3) и с числом состояний $54m^2(n+1)+72mn+19m+18n+9$;

– существует универсальный КА, являющийся одномерным и тоталистичным, с 9139-ю состояниями ячейки.

Иногда об универсальности КА говорят в более сильном смысле, т.е. универсальный КА должен быть в состоянии эмулировать любой КА той же размерности. Известный двумерный автомат фон Неймана с 29-ю состояния был универсальным (помимо того, что в нем существовали самовоспроизводящиеся конфигурации). Затем его результат был улучшен до 20 состояний. Что касается одномерных автоматов, то доказано существование КА, эквивалентных МТ с комбинацией (*m,n*), которые имеют (*m*+2*n*) состояний. И, соответственно, одномерный универсальный КА будет иметь $18=4+2\cdot 7=6+2\cdot 6$ состояний. В работе [59] этот результат улучшен до 14 состояний, а также доказано, что любой одномерный автомат может быть эмулирован тоталистичным. Ряд теорем сформулирован для полутоталистичных (semitotalistic) КА, где аргументом ЛФП является не сумма состояний всех ячеек окрестности, включая, быть может, саму ячейку (totalistic), а пара аргументов <состояние ячейки, сумма состояний соседей>. Например, GoL полутоталистична. Универсальный полутоталистичный КА Гордона содержит 967 состояний (вместо 9139).

Среди элементарных КА доказано (M.Cock), что КА правила 110 универсален; предположительно, универсален и КА правила 54.

В последнее время появились исследования по времяконструируемым (time-constructible) функциям в контексте КА [60]. Существуют два различных определения:

1) целочисленная функция $f:\mathbb{N}\to\mathbb{N}$ называется времяконструируемой, если существует целое $n_0>0$ и МТ, которая, имея на входе слово 11..1 ($\forall n>n_0$ единиц), остановится спустя ровно $f(n)$ шагов;

2) … если такая МТ остановится через $O(f(n))$ шагов и в качестве ответа выдаст бинарное или унарное представление $f(n)$.

Во-первых, эти определения легко могут быть перенесены в область КА; во-вторых, существует связь между времяконструируемостями посредством МТ и посредством КА. А именно, как доказано в [60]:

– если $f(n)$ вычислима посредством МТ за $O(f(n)-n)$ шагов, то $f(n)$ КА-времяконструируема;

– если две функции КА-времяконструируемы, то их сумма, произведение и их экспоненты также КА-времяконструируемы;

– если $f(n), g(n)$ - КА-времяконструируемы и $\lim(f(n)/g(n))=0, f(n)\geq n$, то найдется такой язык, что его слова будут распознаваемы некоторым КА за время $g(n)$, но не всяким КА за время $f(n)$.

Клеточный автомат называется обратимым (ОКА), если существует другой КА, такой что $(G\circ F)(c)=(F\circ G)(c)=I(c)=c$, где c – произвольная глобальная конфигурация, а *G*, *F* - глобальные функции перехода первого и второго КА, соответственно [61-64]. Определение конфигурации типа «сад Эдема» достаточно сложно, что связано с ограниченностью паттерна (при бесконечности поля классического КА), но в простом смысле: садом Эдема называется (глобальная) конфигурация, не имеющая для себя предшественника. Из теоремы Кёртиса-Хедлунда-Линдона (чаще пишут Хедлунда, Curtis-Hedlind-Lyndon theorem) следует теорема о садах Эдема (Garden-of-Eden theorem, 1963):

– КА является обратимым тогда и только тогда, когда он не содержит конфигурации типа «сад Эдема».

Концепция садов Эдема была предложена в 60-х гг. Муром и Майхилом, см. анализ [65]. В частности, GoL допускает сады Эдема, поскольку в ней существуют конфигурации, имеющие несколько предшественников.

В 1977г. Т.Тоффоли показал, как любой КА размерности *d* можно эмулировать (*d*+1)-мерным ОКА. Следствием этого результата является утверждение о существовании двумерного ОКА. В. 1989г. Морита и Харао показали как эмулировать любую обратимую МТ одномерным ОКА. Поэтому справедлива теорема:

– Существует одномерный ОКА, который обладает универсальной вычислимостью.

Обратимые КА являются «царским путем» в область бильярдных вычислений. К настояще-

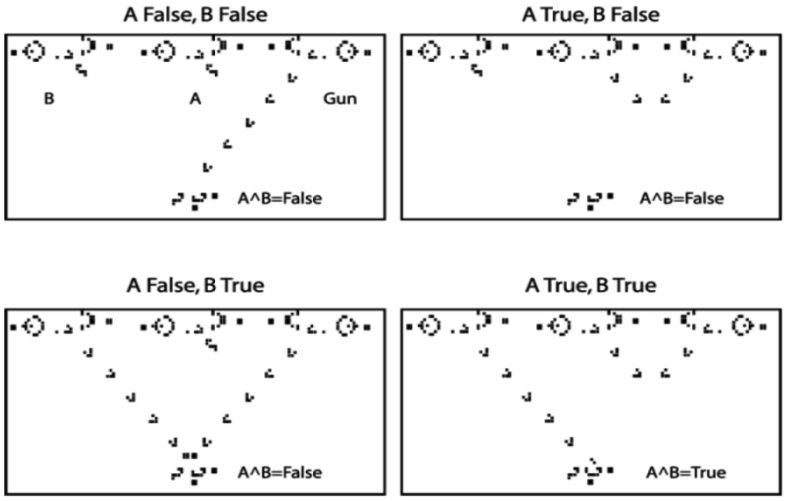


Рис.6. Вентиль 2И, реализованный в пространстве игры GoL [67]

му времени создана обширная концепция Particle-Machine Model of Computation [66, 67] – вычислений, основанных на соударениях. Аналогично GoL привлекательной выглядит мысль идентифицировать сигналы как трансмиттеры информации и столкновения как логические операции, но построение компьютера-коллайдера в одномерном пространстве сопряжено с техническими трудностями (например, связанными с взаимопересечением сигналов). Большую роль в создании этой концепции сыграли работы Фредкина, Тоффоли, Марголуса и по HPP-КА, тематически решающие задачи по решеточным газам.

Для реализации логических вентилей (Рис. 6) необходимы [67]:

– Некоторые формы электрических импульсов, репрезентирующих входы -> **глайдеры.**

– Провода для передачи импульсов -> **траектории движения глайдеров.**

– Процессорный элемент, ассоциирущий входы и вычисляющий булевый результат -> **столкновение глайдеров.**

– Элемент, помещенный после процессорного, выдающий выход -> **столкновение глайдеров с неподвижной фигурой (фигурами).**

Как справедливо пишет А.Н. Непейвода [68], «более смелое направление разработки реверсивных процессоров — вычислители на базе клеточных автоматов. Теоретическая модель реверсивного клеточного автомата обладает рядом очень естественных свойств, что выгодно отличает ее от модели реверсивной машины Тьюринга… Биллиардная модель Тоффоли–Фредкина была описана на основе блочных клеточных автоматов… Эта модель свободна от недостатков, связанных с копированием данных, однако возникает проблема эффективного программирования на клеточных автоматах. Пока что неизвестны способы ее разрешить иначе, чем переборными методами».

Заключение

Прикладные задачи информационных технологий требуют, несмотря на прошедшие семьдесят лет истории развития клеточных автоматов, анализа практических вопросов вычислимости в КА, поэтому все больше значение приобретает специальная вычислимость. Некоторые рецепты могут быть взяты из богатого опыта конструктивных доказательств достаточно универсальных теорем, другие – из взаимодействия с другими коннекционистскими моделями, прежде всего нейросетевыми. Третьи рецепты могут быть получены путем обобщения практики математического моделирования прикладных задач в каких-то предметных областях. При этом клеточные автоматы, будучи гомогенным структурами, обладают своей вычислительной спецификой по сравнению, например, с нейронными сетями или, в ряде случаев частными методами вычислительной математики (конечных элементов, конечных объемов, частиц-в-ячейке).

Несмотря на многообразные применения и, следовательно, сложность обобщения, приходится констатировать, что для множества сравни-

тельно простых и стандартных задач отсутствует какая-либо систематизированная библиотека их клеточно-автоматных решений. Ярким примером является параллельное умножение, для которого старое и полузабытое решение Атрубина 60-х гг. лишь в 90-х гг. получило верификацию, притом, будучи сформулировано не в шаблоне классического КА, а в терминах систолического массива процессорных элементов. Другим примером является обычная сортировка слов. Есть и более сложные задачи: нахождение обратной матрицы, собственных значений, массового вычисления значений полиномов или стандартных функций. При этом эти решения будут зависимыми от выбора исследователем конфигурации КА, например, переход к гексагональной сетке может оказаться перспективных не только в задачах диффузии, но и в задачах, например, распознавания языков. Хотя области применения КА стремительно расширяются и с помощью КА решаются сложные задачи, каким-то странным образом из фокуса внимания исследователей уходят более простые и «прозаические» задачи.

Для КА-вычислимости центральными являются два вопроса. Первый осознан уже давно и формулируется так: «Как следует задать локальные правила перехода, чтобы они обеспечивали желаемую глобальную динамику?». Или в более широкой постановке: «Как следует сконфигурировать КА, чтобы его глобальная конфигурация сходилась к стационарной и содержала в себе ответ?». Или, наоборот, несколько сужая: «Как управлять процессом вычислений, если данные распределены по полю КА, а процесс расчета принципиально децентрализован?». Все эти формулировки показывают многогранность проблемы. Второй вопрос заключается в следующем: «Как наиболее оптимально спроектировать ячейку КА, чтобы этот КА мог быть легко перепрограммирован под возможно большее число задач?». Или, еще так: «Что должен уметь делать элемент КА, чтобы с минимальными затратами времени или площади решить задачи из некоторого класса?». Или так: «Каким минимальным набором состояний и минимальным набором вариативных ЛФП должен обладать элемент КА-вычислителя, чтобы последний обладал универсальностью и эффективностью?». Или так: «Каково оптимальное соотношение между числом ячеек КА, их конструктивной сложностью, определяемой числом состояний и сложностью ЛФП, для эффективного решения стандартных задач?»

Приложение. Сортировка с помощью клеточных автоматов

1. Сортировка букв в одномерном массиве

Исходное предположение: ячейка КА может сравнивать два символа, каждый сортируемый символ первоначально «вписан» в ячейку.

Количество ячеек N: $0 \le i \le N-1$. *Соседство:* 3 ячейки (или 2 для граничных): *i*-я – центральная, $i-1,\ i+1$ – боковые. *Границы* не замкнутые. *Состояние ячейки* представлено двумя компонентами $\langle S,F \rangle$: *S*- целое число (буква), *F*- булево.

Начальные состояния для каждой из ячеек:

- *S*: случайное целое число (или введенное пользователем);
- *F: (i/2)%2* где «/» - целочисленное деление, «%»–остаток от деления по модулю.

Правила перехода:

$$S_i := (F_i = F_{i+1}) \cdot ((S_i > S_{i+1}) \cdot S_{i+1} + (S_i \le S_{i+1}) \cdot S_i) + (F_i = F_{i-1}) \cdot ((S_i < S_{i-1}) \cdot S_{i-1} + (S_i \ge S_{i-1}) \cdot S_i), \quad F_i := F_{i-1}$$

Правила для граничных ячеек:

Для 0-й ячейки:

$$S_0 := (F_0 = F_1) \cdot ((S_0 > S_1) \cdot S_1 + (S_0 \le S_1) \cdot S_0),\ F_0 := 1 - F_1$$

Для (*N*-1)-й ячейки:

$$S_0 := (F_{N-1} = F_{N-2}) \cdot (S_{N-1} < S_{N-2}) \cdot S_{N-2},\ F_{N-1} := F_{N-2}$$

2. Сортировка слов в матрице двумерного КА

Допустим, есть N слов, максимальное количество букв в слове не превосходит M. Тогда будем рассматривать двумерный массив размера $N*M$. Для слов, размера меньшего M, оставшиеся ячейки доопределим символом пробела.

Состояние ячейки имеет 4 компонента (три флага):

- S: Символ (число), собственно данные;
- W: Флаг, указывающий направление смены состояния S (0, 1, 2, 3). Если сравнение двух символов удовлетворяет неравенству «>» или «<» в зависимости от порядка сортировки, то W=1. Если символы равны – W:=2, в остальных случаях W:=0. W=3–флаг для M-го столбца, определяющий конец слова (граничное условие).
- F: Флаг, определяющий блочность строк (0, 1)
- K: Флаг, определяющий активность ячейки (0 или 1).

Соседство: 5 соседних ячеек (центральная, справа, снизу, слева, сверху) по шаблону Неймана «крест». *Границы:* замкнуты, т.е. правая продолжается слева.

Начальное состояние:

S: случайное целое число (или введенное пользователем);

W: 0

F: $(y / 2)\%2$ где «/» - целочисленное деление, «%»–остаток от деления по модулю 2, где y задается номером строки.

K: 0

Начальное состояние для границ:

Для M-го столбца W: =3, для 0-го столбца K:=1

Сравнение строк проходит по аналогии с одномерным случаем: сравниваем по две строки с одинаковым флагом F. Проиндексируем: U, D–верхняя и нижняя, L, R–левая и правая соседние ячейки. Сама ячейка не индексируется.

Правила перехода (псевдокод):

Для компоненты K:

$K:=(K_L=1)$

Для компоненты F:

$If\ K_R==1$

$F= (K_R=1)*F_D+(K_R\neq1)*F$

Для граничных ячеек:

Для N-й строки $F= (R_K=1)*(1-U_F)+(R_K\neq1)*F$

Для компонент W, K и S:

```
if(K=1)
if(F=F_U)
                W:=((W_L=2)+ (W_L=3))*((S<S_U) +(S=S_U)*2)+
                +((W_L=1)* (W!=3))*1
else if(F=F_D)
                W:=((W_L=2)+ (W_L=3))*((S>S_D)*1+(S= S_D)*2)+
                +((W_L=1)* (W!=3))*1
end;
        // Для граничных ячеек (M-го столбца)
If(W=3)
        if(F=F_U)
                S:=(W_L=2)*(S<S_U)*S_U +( W_L=1) *(S=S_U)* S_U)+
                +((W_L!=2)* (W!=1))*S
        else if(F=F_D)
                S:=(W_L=2)*( S>S_D)*S_D +( W_L=1) *(S=S_D)* S_D)+
                +((W_L!=2)* (W!=1))*S
        end;
```

```
                F:=F_D
                //Для граничной N-й строки F:1-F_U
end;
else if((K_R=1)*(W!=3))
        F:=F_D
        //Для граничной N-й строки F:1-F_U

if(F=F_U)
                 W:=(W=1 + W=2)*0+(W!=1 * W!=2)*W
                S:= (W=1)*S_U+(W!=1)*S
else if(F=F_D)
                W:=(W=1 + W=2)*0+(W!=1 * W!=2)*W
                S:= (W=1)*S_D+(W!=1)*S
end;
end;
```

Проиллюстрируем работу КА на поле 7*6 (семь слов, каждое не более 6 знаков). Ниже показаны состояния КА, причем компоненты состояния отображаются в виде ${}^{K}S_{W}^{F}$ (как в таблице Менделеева).

${}^{1}t_{0}^{0}$	${}^{0}a_{0}^{0}$	${}^{0}b_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}e_{0}^{0}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{1}k_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}o_{0}^{0}$	${}^{0}n_{0}^{0}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{1}b_{0}^{1}$	${}^{0}r_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}i_{0}^{1}$	${}^{0}n_{0}^{1}$	$\begin{matrix}0&1\\&3\end{matrix}$
${}^{1}o_{0}^{1}$	${}^{0}r_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}n_{0}^{1}$	${}^{0}g_{0}^{1}$	${}^{0}e_{3}^{1}$
${}^{1}h_{0}^{0}$	${}^{0}a_{0}^{0}$	${}^{0}t_{0}^{0}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{1}a_{0}^{0}$	${}^{0}p_{0}^{0}$	${}^{0}p_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}e_{0}^{0}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{1}c_{0}^{1}$	${}^{0}u_{0}^{1}$	${}^{0}s_{0}^{1}$	${}^{0}t_{0}^{1}$	${}^{0}o_{0}^{1}$	${}^{0}m_{3}^{1}$

Начальное состояние ячеек поля КА

${}^{0}t_{1}^{0}$	${}^{1}a_{0}^{0}$	${}^{0}b_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}e_{0}^{0}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}k_{1}^{0}$	${}^{1}l_{0}^{0}$	${}^{0}o_{0}^{0}$	${}^{0}n_{0}^{0}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}b_{0}^{1}$	${}^{1}r_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}i_{0}^{1}$	${}^{0}n_{0}^{1}$	$\begin{matrix}0&1\\&3\end{matrix}$
${}^{0}o_{0}^{1}$	${}^{1}r_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}n_{0}^{1}$	${}^{0}g_{0}^{1}$	${}^{0}e_{3}^{1}$
${}^{0}h_{1}^{0}$	${}^{1}a_{0}^{0}$	${}^{0}t_{0}^{0}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}a_{1}^{0}$	${}^{1}p_{0}^{0}$	${}^{0}p_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}e_{0}^{0}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}c_{0}^{1}$	${}^{1}u_{0}^{1}$	${}^{0}s_{0}^{1}$	${}^{0}t_{0}^{1}$	${}^{0}o_{0}^{1}$	${}^{0}m_{3}^{1}$

1-й ход

${}^{0}k_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}b_{1}^{0}$	${}^{1}l_{0}^{0}$	${}^{0}e_{0}^{0}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}t_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}o_{1}^{0}$	${}^{1}n_{0}^{0}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}b_{0}^{1}$	${}^{0}r_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{1}i_{0}^{1}$	${}^{0}n_{0}^{1}$	$\begin{matrix}0&1\\&3\end{matrix}$
${}^{0}o_{0}^{0}$	${}^{0}r_{0}^{0}$	${}^{0}a_{0}^{1}$	${}^{1}n_{0}^{1}$	${}^{0}g_{0}^{1}$	${}^{0}e_{3}^{1}$
${}^{0}a_{0}^{0}$	${}^{0}p_{0}^{0}$	${}^{0}t_{1}^{0}$	$\begin{matrix}1&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}h_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}p_{1}^{0}$	${}^{1}l_{0}^{0}$	${}^{0}e_{0}^{0}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}c_{0}^{1}$	${}^{0}u_{0}^{1}$	${}^{0}s_{0}^{1}$	${}^{1}t_{0}^{1}$	${}^{0}o_{0}^{1}$	${}^{0}m_{3}^{1}$

3-й ход

${}^{1}b_{0}^{1}$	${}^{0}r_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}i_{0}^{1}$	${}^{0}n_{0}^{1}$	$\begin{matrix}0&1\\&3\end{matrix}$
${}^{1}k_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}o_{0}^{0}$	${}^{0}n_{0}^{0}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{1}a_{0}^{0}$	${}^{0}p_{0}^{0}$	${}^{0}p_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}e_{0}^{0}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{1}t_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}b_{0}^{1}$	${}^{0}l_{0}^{1}$	${}^{0}e_{0}^{1}$	$\begin{matrix}0&1\\&3\end{matrix}$
${}^{1}c_{0}^{1}$	${}^{0}u_{0}^{1}$	${}^{0}s_{0}^{1}$	${}^{0}t_{0}^{1}$	${}^{0}o_{0}^{1}$	${}^{0}m_{3}^{1}$
${}^{1}o_{0}^{0}$	${}^{0}r_{0}^{0}$	${}^{0}a_{0}^{0}$	${}^{0}n_{0}^{0}$	${}^{0}g_{0}^{0}$	${}^{0}e_{3}^{0}$
${}^{1}h_{0}^{0}$	${}^{0}a_{0}^{0}$	${}^{0}t_{0}^{0}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$

18-й ход

${}^{0}a_{0}^{1}$	${}^{1}p_{0}^{1}$	${}^{0}p_{0}^{1}$	${}^{0}l_{0}^{1}$	${}^{0}e_{0}^{1}$	$\begin{matrix}0&1\\&3\end{matrix}$
${}^{0}b_{0}^{1}$	${}^{1}r_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}i_{0}^{1}$	${}^{0}n_{0}^{1}$	$\begin{matrix}0&1\\&3\end{matrix}$
${}^{0}c_{0}^{0}$	${}^{1}u_{0}^{0}$	${}^{0}s_{0}^{0}$	${}^{0}t_{0}^{0}$	${}^{0}o_{0}^{0}$	${}^{0}m_{3}^{0}$
${}^{0}h_{0}^{0}$	${}^{1}a_{0}^{0}$	${}^{0}t_{0}^{0}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&0\end{matrix}$	$\begin{matrix}0&0\\&3\end{matrix}$
${}^{0}k_{0}^{1}$	${}^{1}l_{0}^{1}$	${}^{0}o_{0}^{1}$	${}^{0}n_{0}^{1}$	$\begin{matrix}0&1\\&0\end{matrix}$	$\begin{matrix}0&1\\&3\end{matrix}$
${}^{0}o_{0}^{1}$	${}^{1}r_{0}^{1}$	${}^{0}a_{0}^{1}$	${}^{0}n_{0}^{1}$	${}^{0}g_{0}^{1}$	${}^{0}e_{3}^{1}$
${}^{0}t_{0}^{0}$	${}^{1}a_{0}^{0}$	${}^{0}b_{0}^{0}$	${}^{0}l_{0}^{0}$	${}^{0}e_{0}^{0}$	$\begin{matrix}0&0\\&3\end{matrix}$

37-й ход (конец сортировки)

Литература

1. Anas N. Al-Rabadi, Conservative Reversible Elementary Cellular Automata and their Quantum Computations // In book «Cellular Automata - Innovative Modelling for Science and Engineering», http://cdn.intechopen.com/pdfs-wm/15007.pdf
2. В.Е. Туманов. Основы проектирования реляционных баз данных. Эл.ресурс - http://www.intuit.ru/goods_store/ebooks/8322
3. Гергель В.П., Стронгин, Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. Учебное пособие – Нижний Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003. 184 с.
4. Алексеев В.Е., Таланов В.А. Графы. Модели вычислений. Структуры данных: Учебник. – Нижний Новгород: Изд-во ННГУ, 2005. 307 с.
5. Стефанова Т.С. Отбор содержания обучения неклассическим вычислительным моделям // Известия РГПУ им. А.И. Герцена. 2008. №58, с.440-451 URL: http://cyberleninka.ru/article/n/otbor-soderzhaniya-obucheniya-neklassicheskim-vychislitelnym-modelyam (дата обращения: 15.04.2015).
6. Анисов А.М. Классическая вычислимость и признаки индетерминизма // Логические исследования, #14, 2007, С. 5-26.
7. John E. Savage. VLSI Models of Computation // in e-book “Models of Computation: Exploring the Power of Computing” by John E. Savage, released of July 23, 2008, pp.575-603 -cs.brown.edu/~jes/book/pdfs/ ModelsOfComputation.pdf
8. Carl Hewitt, Peter Bishop, and Richard Steiger. 1973. A universal modular ACTOR formalism for artificial intelligence. In Proceedings of the 3rd international joint conference on Artificial intelligence (IJCAI'73). MorganKaufmannPublishersInc., SanFrancisco, CA, USA, 235-245.
9. Medler, David A. A Brief History of Connectionism. Neural Computing Surveys, 1(2), 1998 – p.18-72.
10. D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors. Parallel Distributed Processing, volume 1:Foundations. MIT Press, Cambridge, MA, 1986
11. Wolfram S. A New Kind of Science.- N.Y.: Wolfram Media, 2002.
12. C. Langton, “Computation at the Edge of Chaos,” Physica D, 42:12-37, 1990.
13. Jarkko Kari. Cellular Automata: Tutorial, http://users.utu.fi/jkari/ca/CAintro.pdf
14. Кудрявцев В., Подколзин А., Болотов А. Основы теории однородных структур. — Москва: Москва, 1990. — С. 296.
15. Аладьев В.З. Классические однородные структуры. Клеточные автоматы. FultusBooks. 2009.
16. J. Kari Theory of cellular automata:A survey/ Theoretical Computer Science 334 (2005) 3 – 33.
17. Жизнь на плоскости Лобачевского, http://habrahabr.ru/post/168421/
18. Margenstern M. Small Universal Cellular Automata in Hyperbolic Spaces. A CollectionofJewels, Springer, 2013, -327 pp.
19. M. Marques-Pita and L.M. Rocha [2008]."Conceptual Structure in Cellular Automata: TheDensity Classification Task".In: Artificial Life XI: Eleventh International Conference on theSimulation and Synthesis of Living Systems, S. Bullock, J. Noble, R. A. Watson, and M. A.Bedau (Eds.). MIT Press, InPress.
20. H. Juillé and J.B. Pollack. “Coevolving the ‘ideal’ trainer: Application to the discovery of cellular automata rules.” In: J.R. Koza, W. Banzhaf, K. Chellapilla, M. Dorigo, D.B. Fogel, M.H. Garzon, D.E. Goldberg, H. Iba and R.L. Riolo (eds.). Genetic Programming 1998: Proceedings of the Third Annual Conference, San Francisco, CA: Morgan Kaufmann, 1998.
21. Christopher Stone, Larry Bull. Solving the Density Classification Task Using Cellular Automaton 184 with Memory //Complex Systems, 18, 2009, pp. 329-34 http://www.complex-systems.com/pdf/18-3-4.pdf
22. R. Alonso-Sanz and L. Bull, “Elementary Coupled Cellular Automata with Memory,” Automata 2008: Theory and Applications of Cellular Automata (A. Adamatzky, R. Alonso-Sanz, A. Lawniczak, G. J. Martinez, K. Morita, and T. Worsch, eds.), Frome, UK: Luniver Press, 2008 pp.72-99.
23. Gacs, P., Kurdyumov, L., and Levin, L. (1978). One-dimensional uniform arrays that wash out finite islands. Probl. Peredachi. Inform., 14:92–98.
24. Land M., Belew R. K. No perfect two-state cellular automata for density classification exists //Physical review letters. – 1995. – Т. 74. – №. 25. – С. 5148.
25. M. Mitchell, J. P. Crutchfield, and P. T. Hraber, “Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments”, Physica D, 75 (1994), 361-391.
26. Yamashita K. et al. The Firing Squad Synchronization Problems for Number Patterns on a Seven-Segment Display and Segment Arrays //IEICE transactions on information and systems. – 2010. – Т. 93. – №. 12. – С. 3276-3283.
27. Umeo H., Nishide K., Kubo K. A simple optimum-time FSSP algorithm for multi-dimensional cellular automata //arXiv preprint arXiv:1208.2761. – 2012.
28. М.В. Поникаров, Использование игр клеточных автоматов для синхронизации в распределённых системах // бизнес_информатика №3(05)–2008 г., 31-36
29. Mazoyer J. On optimal solutions to the firing squad synchronization problem //Theoretical Computer Science. – 1996. – Т. 168. – №. 2. – С. 367-404.
30. Helene Vivien. An Introduction to CELLULAR AUTOMATA http://www.liafa.jussieu.fr/~yunes/ca/archives/bookvivien.pdf
31. Mazoyer J., Terrier V. Signals in one-dimensional cellular automata // Theoretical Computer Science, vol.217, 1999, pp.53-80.
32. Iwamoto C. et al. Constructible functions in cellular automata and their applications to hierarchy results //Theoretical Computer Science. – 2002. – Т. 270. – №. 1. – С. 797-809.
33. P.C. Fisher, Generation on primes by one dimentional real time iterative array, J. ACM 12 (1965) 388-394.
34. Mahata K., Das S. Gateway node in WSN as the queen bee in a honey bee colony //Communications, Devices and Intelligent Systems (CODIS), 2012 International Conference on. – IEEE, 2012. – С. 270-273.
35. Banda R. N. D. P. Anonymous Leader Election in One- and Two-Dimensional Cellular Automata // Student Dissertation Thesis, Comenius University in Bratislava, 2014 - http://peterbanda.net/Peter_Banda-Comenius_Dissertation-2014.pdf

36. Beckers A., Worsch T.A perimeter–time CA for the queen bee problem //Parallel Computing. – 2001. – Т. 27. – №.5. – С. 555-569.
37. Nichitiu C., Mazoyer J., Rémila E. Algorithms for leader election by cellular automata //Journal of Algorithms. – 2001. – Т. 41. – №. 2. – С. 302-329.
38. Горнев Е. С., Матюшкин И. В., Теплов Г. С. Анализ концепций неклассического компьютинга и парадигмы коннекционизма // Электронная техника. Серия 3: Микроэлектроника. 2015. № 2 (158). С. 45-66.
39. HEEN Olivier, Efficient constant speed-up for one dimensional cellular automata calculators // Parallel Computing 23: pp1663-1671, 1997.
40. ATRUBIN A.J. A one-dimensional real-time iterative multiplier // IEEE Transactions on Electronic Computers EC-14 : pp394-399, 1965.
41. Even S., Litman A. A systematic design and explanation of the Atrubin multiplier //Sequences II. Methods in Communication, Security, and Computer Science – Springer New York, 1993. – С. 189-202.
42. Бандман О.Л. Инварианты клеточно-автоматных моделей реакционно-диффузионных процессов // ПДМ . 2012. №3. С.108-120.
43. Weimar J.R. Cellular automata for reaction-diffusion systems //Parallel computing. – 1997. – Т. 23. – №. 11. – С. 1699-1715.
44. Toffolli T. Cellular automata as an alternative to (rather than approximation of) differential equations in modeling physics // Physica D. 1984. V. 10. P. 117 – 127.
45. Ванаг В. К. Исследование пространственно распределенных динамических систем методами вероятностного клеточного автомата //Успехи физических наук. – 1999. – Т. 169. – №. 5. – С. 481-505.
46. Paola Flocchini, Vladimir Cezar. Radial View of Continuous Cellular Automata // Fundamenta Informaticae XXI (2001) 1001–1018.
47. Michal Bidlo, ZdenekVasicek, and Karel Slany. Sorting Network Development Using Cellular Automata // In Evolvable Systems: From Biology to Hardware. 9th International Conference, ICES 2010, York, UK, September 6-8, 2010, Proceedings, LNCS 6274. London: Springer London, 2010. p. 85-96.
48. Lafe O. Data compression and encryption using cellular automata transforms //Intelligence and Systems, 1996, IEEE International Joint Symposia on. – IEEE, 1996. – С. 234-241.
49. Thomas Zheng. Discrete Orthogonal Transforms Using Cellular Automata // NKS2004 (New Kind of Science Develop), April 22-25, 2004 http://www.wolframscience.com/conference/2004/presentations/
50. Shaw B., Wong L. C. Algorithmic Self-Assembly of Circuits. – 2002.
51. О.О. Евсютин, А.А. Шелупанов Основные подходы к использованию математического аппарата теории клеточных автоматов для решения задач кодирования информации // Доклады ТУСУРа, № 2 (32), июнь 2014, С.60-65.
52. Thomas H. Speller. A Study within the Nanosystem Architecture Domain: Self-assembly of Graphene // NKS2007 Conference, July 13, 2007.
53. Speller, T.H., Jr., D. Whitney, and E. Crawley, Using Shape Grammar to Derive Cellular Automata Rule Patterns. Complex Systems, 2007. 17: p. 79-102.
54. White R, Engelen G, Uljee I, 1997, "The use of constrained cellular automata for high-resolution modelling of urban land-use dynamics" Environment and Planning B: Planning and Design 24(3) 323 – 343.
55. Krasnikov G. Y., Matyushkin I. V., Korobov S. V. Visualization of Cellular Automata in Nanotechnology. – 2014. Vol.(3), № 3 С. 98-114.
56. Buchholz T., Klein A., Kutrib M. Real-Time Language Recognition by Alternating Cellular Automata //Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics. – Springer-Verlag, 2000. – С. 213-225.
57. Terrier V. Characterization of real time iterative array by alternating device //Theoretical computer science. – 2003. – Т. 290. – №. 3. – С. 2075-2084.
58. Gordon D. On the computational power of totalistic cellular automata //Mathematical systems theory. – 1987. – Т. 20. – №. 1. – С. 43-52.
59. Jiirgen A., Culik K. A simple universal cellular automaton and its one-way and totalistic version //Complex Systems. – 1987. – Т. 1. – С. 1-16.
60. Iwamoto C. et al. Constructible functions in cellular automata and their applications to hierarchy results //Theoretical Computer Science. – 2002. – Т. 270. – №. 1. – С. 797-809.
61. A.L. Stempkovsky, P.A. Vlasov, G.V. Kozin. Algorithmic Environment for VLSI Design on Cellular Automata // Proceedings of a Joint Symposium : Information Processing and Software, Systems Design Automation, Academy of Sciences of the USSR, Siemens AG, FRG, Moscow, June 5/6, 1990, Springer-Verlag pp.308-312.
62. Стемпковский А.Л., Осипов Л.Б., Селезнев С.З. Исследование вопросов реализации нейросети по СИП-технологии для построения отказоустойчивых однородных архитектур. Информационные технологии и вычислительные системы. - М., 1995, №9, С.58.
63. Стемпковский А.Л., Осипов Л.Б., Селезнев С.З. Проблемы реализации отказоустойчивых архитектур нейрочипов по технологии систем с интеграцией на пластине. Информационные технологии. - М., 1997. №5. С.15.
64. Стемпковский А.Л. Отказоустойчивые архитектуры микроэлектронных вычислительных систем. Информационные технологии и вычислительные системы. - М., 2001. - № 2-3. С.40.
65. Amoroso S., Cooper G. The Garden-of-Eden theorem for finite configurations //Proceedings of the American Mathematical Society. – 1970. – Т. 26. – №. 1. – С. 158-164.
66. Jakubowski M. H. Computing with solitons in bulk media. – PhD, Princeton University, 1999, pp.158.
67. Rennard, Jean-Philippe. Implementation of Logical Functions in the Game of Life. //Adamatzky A. Collision-Based Computing, Springer, pp.491-512, 2002.
68. А.Н. Непейвода. Реверсивные вычисления: обзор мирового опыта // Материалы конференции «Параллельные вычисления и задачи управления», Москва, 24-26 октября 2012, A204 - http://paco2012.ipu.ru/procdngs/A204.pdf

Матюшкин Игорь Валерьевич. Старший научный сотрудник АО «НИИ молекулярной электроники». Окончил Московский физико-технический институт в 1997 году. Кандидат физико-математических наук, доцент. Автор около 90 печатных работ и одной монографии. Область научных интересов: метаматематика, математическое моделирование, клеточные автоматы, технология микроэлектроники. E-mail: imatyushkin@mikron.ru

Гаврилов Сергей Витальевич. Заведующий отделом Института проблем проектирования в микроэлектронике РАН. Окончил Московский физико-технический институт в 1983 году. Доктор технических наук, профессор. Автор свыше 100 научных работ и двух монографий. Область научных интересов: системы автоматизации проектирования СБИС, методы оптимизации интегральных схем, логический синтез, анализ помехоустойчивости. E-mail: Sergey.V.Gavrilov@ippm.ru

Стемпковский Александр Леонидович. Директор Института проблем проектирования в микроэлектронике РАН, академик РАН. Доктор технических наук, профессор. Область научных интересов: системы автоматизированного проектирования, микро- и наноэлектронные интегральные схемы. E-mail: ippm@ippm.ru