УДК 519.87

Бенаддия Абдельлатиф, О.Ф. Михаль
ХНУРЕ, г. Харьков, Украина, fuzzy16@pisem.net

ЛОКАЛЬНО-ПАРАЛЛЕЛЬНЫЙ СТЕКОВЫЙ АЛГОРИТМ БИНАРНОГО КЛЕТОЧНОГО АВТОМАТА

Приложение принципов локально-параллельной обработки информации к клеточным автоматам позволяет существенно увеличить размеры реализуемых моделей. Для полноты оперирования данными локально-параллельное представление данных должно быть дополнено алгоритмами, реализующими смену состояний ячеек для конкретных видов (вариантов) клеточных автоматов. Предложен и продемонстрирован на тестовых примерах один из таких алгоритмов — локально-параллельный стековый алгоритм бинарного клеточного автомата.

КЛЕТОЧНЫЕ АВТОМАТЫ, ЛОКАЛЬНАЯ ПАРАЛЛЕЛЬНОСТЬ, НЕЧЁТКАЯ ЛОГИКА, СТЕК

Введение

Актуальность проблематики, связанной с моделированием (описанием и предсказанием поведения) процессов, происходящих в *живой природе*, существенно возросла в связи с осознанием мировым сообществом в последнее десятилетие серьёзных взаимосвязанных проблем в столь разнородных направлениях, как экология, климатология, демография, социология. Перспективным аппаратом компьютерного моделирования крупноразмерных систем с распределённой обработкой информации, интерпретируемых применительно к указанным направлениям, являются *клеточные автоматы* (КА). Целесообразность применения КА для моделирования поведения систем указанного типа обусловлена простотой описания конфигурации и набора рабочих правил. Дополнительный выигрыш по объёму моделируемой системы и эффективности работы КА может быть обеспечен *локально-параллельными* (ЛП) алгоритмами [1]. Идеология ЛП алгоритмов совместима с принципами представления *нечёткой информации* [2, 3]. Для этого ключевой элемент нечёткого представления – *функция принадлежности* (ФП) – дискретизируется и масштабируется в соответствии с числом градаций, требуемым (достаточным) для описания конкретной системы. Ввиду дискретного характера и неотрицательности (по определению) значений ФП, ЛП аппарат нечёткого представления данных становится принципиально применимым также к КА. В частности, при чётком описании, как частном случае нечёткого, ФП принимает одно из двух значений (0, 1), что соответствует подвиду КА – *бинарным клеточным автоматам* (БКА) [4]. Для полноты оперирования данными необходимо дополнить аппарат описания [2, 3] ЛП алгоритмами, реализующими смену состояний ячеек КА для конкретных видов (вариантов) КА. Цель настоящей работы – разработка и исследование (оценка эффективности) ЛП варианта одной из процедур работы КА: стекового алгоритма определения уровня (степени) соседства ячеек БКА.

1. Клеточные автоматы

В основе КА [4] лежат следующие представления. Имеется регулярная решётка ячеек. Каждая ячейка может находиться в одном из конечного множества состояний. Для ячейки определено множество *соседних* ячеек – её окружение. Задаётся начальное состояние всех ячеек и набор правил перехода ячеек из одного состояния в другое. На каждой итерации, на основе правил перехода и в соответствии с текущим состоянием соседних ячеек, определяется новое состояние каждой ячейки. Таким образом, КА в целом последовательно сменяет состояния (этапы, фазы жизненного цикла, поколения).

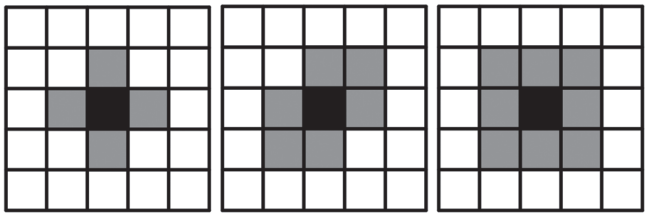
Характерно, что работа КА (переход в следующее поколение) необратима, как необратимы процессы в живой природе. Эволюционирование КА является однонаправленным, как и время в реальном физическом мире. Текущее j-е состояние КА однозначно определяет последующее $(j+1)$-е состояние, но $(j-1)$-е состояние не может быть однозначно восстановлено по текущему j-му состоянию. Таким образом, КА «автоматически» воспроизводят важную особенность реального мира – однонаправленность и необратимость времени.

Привлекательность КА в качестве аппарата моделирования крупноразмерных пространственно распределённых систем определяется также концептуальной наглядностью, малым набором рабочих правил и, как следствие, простотой программной реализации и визуализации процесса работы.

В БКА для записи состояния ячейки используется один бит. Подбором правил работы КА допустимы различные варианты соседства ячеек. Рис. 1 иллюстрирует некоторые из возможных топологий БКА: ситуации с 4, 6 и 8 ячейками-соседями. Центральная ячейка выделена чёрным цветом, её окружение (соседи) – серым.

Ориентированный граф, представленный на рис. 2, иллюстрирует совокупность правил для БКА с 8 соседями рис. 1 (в) для широко известного варианта бинарного КА «игра Жизнь» [5]. Ячейка

КА может находиться в одном из двух состояний: 0 или 1 (узлы графа). При уровне (степени) соседства 3 или 4 (при наличии в окружении 3 или 4 единичных ячеек) – единичное состояние сохраняется или 0 переходит в 1. Иначе – 0 сохраняется или 1 переходит в 0.



а *б* *в*

Рис. 1. Бинарные КА с вариантами соседства: 4 (*а*), 6 (*б*) и 8 клеток (*в*)

При компьютерном моделировании на КА содержательность (репрезентативность, информационная ёмкость) моделей растёт с их размерами. В частности, применительно к экологии и статистической биологии для моделирования поведения растительных ареалов или животных сообществ, с учётом информативности для статистической обработки [6], представляются целесообразными модели объёмом более 10^4 ячеек. В связи с этим при компьютерной реализации реальной модели на БКА возникает проблема экономного расходования ресурсов *вычислительной системы* (ВС) – оперативной памяти, дискового пространства, времени работы программы и др. Сокращение расходуемых системных ресурсов эквивалентно общему сокращению затрат на эксплуатацию модели. С другой стороны, экономное расходование ресурсов ВС при фиксированном общем объёме ресурсов – эквивалентно повышению размера или точности модели.

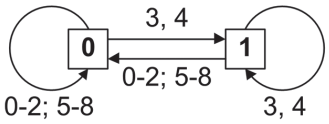


Рис. 2. Правила бинарного КА «игра Жизнь»

Каждая ячейка КА самостоятельно (независимо) изменяется в процессе работы программы. Изменения состояний i-й ячейки происходят, как отмечалось, в соответствии с правилами работы КА: ячейки влияют друг на друга, взаимно обусловливают свое поведение. При этом текущая информация о состоянии *i*-й ячейки должна храниться отдельно от информации о состоянии других ячеек. То есть, для каждой ячейки требуется отдельный элемент хранения. Стандартные форматы хранения данных – расточительны в отношении использования ресурсов ВС. Существенный выигрыш может быть обеспечен при ЛП представлении информации [2, 3].

2. Принцип локальной параллельности

Поясним принцип ЛП представления и обработки информации. Пусть имеется два n-компонентных вектора:

$$A: \{a_1, a_2, a_3,..., a_i,..., a_n\},$$
$$B: \{b_1, b_2, b_3,..., b_i,..., b_n\}, \quad (1)$$

где: $i \in (1, 2, 3,..., n)$; a_i, b_i –целые положительные числа, причём

$$a_i \le a_{max}; \quad b_i \le b_{max}; \quad a_{max} = b_{max}. \quad (2)$$

Условие (2), дополнительно налагаемое на (1), означает, что формат чисел – компонентов векторов – должен быть одинаковым и числа не могут быть сколь угодно большими.

Требуется найти покомпонентную сумму векторов:

$$C: \{ c_1, c_2, c_3,..., c_i,..., c_n \}; \quad c_i = a_i + b_i. \quad (3)$$

Согласно последовательной схеме, для этого должен быть выполнен следующий набор операций:

Шаг 1. Начальная установка: i=1.

Шаг 2. Извлечение значения операнда a_i из регистра памяти, в котором он хранится.

Шаг 3. Извлечение значения операнда b_i из регистра памяти, в котором он хранится.

Шаг 4. Суммирование: $c_i = a_i + b_i$.

Шаг 5. Помещение результата c_i в регистр памяти, в котором он должен храниться.

Шаг 6. $i = i + 1$. Если $i>n$, завершение вычисления; иначе переход к Шагу 2.

Согласно ЛП схеме, для нахождения покомпонентной суммы необходимо выполнить следующий набор операций:

Шаг 1. Конкатенация:

$$A:\{a_1, a_2, a_3,..., a_i,..., a_n\} \rightarrow$$
$$\rightarrow a_\# = (a_1 \oplus a_2 \oplus a_3 \oplus ... \oplus a_i \oplus ... \oplus a_n); \quad (4)$$
$$B:\{b_1, b_2, b_3,..., b_i,..., b_n\} \rightarrow$$
$$\rightarrow b_\# = (b_1 \oplus b_2 \oplus b_3 \oplus ... \oplus b_i \oplus ... \oplus b_n).$$

Шаг 2. Извлечение значения операнда (конкатенанда) $a_\#$ из регистра памяти, в котором он хранится.

Шаг 3. Извлечение значения операнда (конкатенанда) $b_\#$ из регистра памяти, в котором он хранится.

Шаг 4. Суммирование конкатенандов:
$c_\# = a_\# + b_\#$.

Шаг 5. Помещение результата $c_\#$ в регистр памяти, в котором он должен храниться.

Шаг 6. Деконкатенация:

$$c_\# = (c_1 \oplus c_2 \oplus c_3 \oplus ... \oplus c_i \oplus ... \oplus c_n) \rightarrow$$
$$\rightarrow C:\{c_1, c_2, c_3,..., c_i,..., c_n\}. \quad (5)$$

Операции конкатенации и деконкатенации требуют дополнительного пояснения. Обычно эти термины применяются при описании обработки

символьных последовательностей. В рассматриваемом случае конкатенируются целые положительные числа в двоичном представлении.

Числа (исходные данные, промежуточные значения, результаты) при их обработке располагаются в центральном процессоре в отдельных регистрах – линейках бинарных ячеек памяти. Отсюда применяемый далее термин – *регистровое представление* (РгП). В ЛП представлении результат конкатенации есть некоторое целое положительное число, помещаемое в едином регистре, – РгП, которое интерпретируется как состоящее из сегментов согласно длине конкатенантов. При деконкатенации сегменты извлекаются из РгП. Таким образом, конкатенация есть операция уплотнения представления компонентов векторов (1) в виде сегментов в РгП $a_{\#}$ и $b_{\#}$ вида (4). Содержимые РгП $a_{\#}$ и $b_{\#}$ обрабатываются целиком, как числа. Деконкатенация – обратная операция – раскрытие РгП. Содержимое i-го сегмента c_i извлекается и помещается в индивидуальный регистр. В результате, согласно ЛП схеме, операция (в рассматриваемом случае суммирование соответствующих компонентов векторов) производится над РгП, а результат C:$\{c_1, c_2, c_3,..., c_i,..., c_n\}$ выглядит так, как если бы операции производились над парами операндов раздельно.

Сравним представленные выше пошаговые последовательности операций. Вычислительные блоки операций Шаги 2 – 5 в последовательной и ЛП схемах совпадают. Различие состоит в том, что в последовательной схеме блок выполняется n-кратно, в ЛП схеме – однократно. Этим обеспечивается выигрыш в производительности. Разумеется, процедуры конкатенация и деконкатенация (Шаги 1 и 6 ЛП схемы) являются протяженными во времени, т.е. связаны с дополнительным расходованием ресурсов ВС. Однако, если в системе реализуются сложные вычисления, например с последовательным применением нескольких ЛП алгоритмов, РгП составляются из исходных данных только в самом начале вычислительного блока, а результаты извлекаются из РгП только в самом конце. Затраты ресурсов на конкатенацию-деконкатенацию при этом могут быть пренебрежимо малы по сравнению с общими затратами ресурсов ВС. При этом выигрыш в производительности приближается к n-кратному.

Очевидно, что ЛП схема тем более эффективна, чем больше число конкатенированных сегментов n. Приближенно (при больших объёмах вычислительных ЛП блоков), выигрыш в производительности пропорционален длине регистра вычислительного устройства и обратно пропорционален длине сегмента [2, 3].

Для единства структуры данных, разрядности сегментов, задействованных под операнды $a_{\#}$ и $b_{\#}$ и результат $c_{\#}$, должны совпадать. При выполнении операции соседствующие сегменты не должны взаимодействовать, поэтому должны быть приняты меры по согласованию разрядностей. Форматы представления ЛП данных в системе должны быть организованы так, чтобы в сегментах $c_{\#}$ было изначально зарезервировано достаточно места для размещения результата операции над $a_{\#}$ и $b_{\#}$.

Выигрыш в производительности и характеристики точности системы конкурируют между собой. При фиксированной разрядности регистра производительность ЛП системы тем выше, чем больше число сегментов. Следовательно, для повышения производительности целесообразно сократить разрядность сегментов до минимума. При этом уменьшается число градаций, с которым могут быть представлены в системе соответствующие обрабатываемые величины. Таким образом, при проектировании систем с использованием принципов ЛП при выборе размера сегментов должны приниматься компромиссные решения по соотношению производительности вычислений и точности результатов.

Для операций должны быть разработаны соответствующие алгоритмы, подобные ЛП алгоритмам операций нечёткой логики, рассмотренным в [2, 3]. Имеются ограничения, определяемые длиной сегментов. В частности, рассмотренная ЛП операция суммирования реализуема только для положительных операндов ограниченного размера. Этому условию удовлетворяют, в частности, значения ФП, применяемые в нечетком теоретико-множественном описании. Применительно к ЛП представлению они масштабированы в соответствии с размерами сегментов и конкатенированы в РгП [2, 3]. Для ячеек КА так же имеется фиксированное число неотрицательных состояний. Следовательно, ЛП представление информации потенциально применимо и к КА.

3. Локально-параллельное представление КА

Упрощённо процессор ВУ Фон-Неймановского типа имеет два 32-разрядных регистра для хранения операндов и один 32-разрядный регистр для помещения результата. Процессор реализует стандартный набор арифметических и логических операций, включая поразрядную (побитовую) логику и регистровые сдвиги.

Как отмечалось (п. 2), при ЛП обработке предполагается хранение нескольких элементов информации в виде раздельных непересекающихся сегментов РгП. В случае ЛП представления КА логично разместить в отдельных сегментах РгП отдельные клетки КА. При этом формат размещения информации в РгП должен быть удобным для реализации ЛП алгоритмов обработки.

В обычной записи для хранения решётки ячеек (матрицы) КА требуется двумерный массив. При

размере поля M строк на N столбцов требуется $M{\cdot}N$ индивидуально адресуемых элементов хранения информации. В случае 32-разрядного ВУ Фон-Неймановского типа это составляет $32{\cdot}M{\cdot}N$ бит. В ЛП представлении ячейки КА хранятся в сегментах РгП. То есть индивидуально адресуемыми элементами хранения становятся РгП, а не отдельные ячейка КА. В случае БКА в РгП может быть до 32 однобитовых сегментов. Строка поля БКА может быть помещена в $[M/32]+1$ РгП (квадратные скобки обозначают взятие целой части от деления). В результате для представления всего БКА необходим массив из $([M/32]+1){\cdot}N \approx M{\cdot}N/32$ РгП (индивидуально адресуемых элементов хранения информации). Соответственно, если ячейка КА имеет n состояний, для её представления требуется k бит с соблюдением соотношения $2^k \geq n$, то есть $k \geq log_2 n$, то есть $k = [log_2 n]+1$ бит. Таким образом, в 32-разрядном ВУ для представления строки элементов КА требуется $[M{\cdot}k/32]+1$, а для представления всего массива – $M{\cdot}N{\cdot}k/32$ индивидуально адресуемых элементов хранения информации.

Для БКА наиболее плотным (компактным, экономным, с минимальным расходованием ресурсов ВУ) является однобитовое сегментное представление. Например, поле БКА размером 30×30 элементов может быть размещено в одномерном массиве A из 30 штук 32-разрядных чисел в формате integer: $(a_1, a_2,..., a_j,..., a_{29}, a_{30})$. Сложность проявляется при реализации работы БКА. Как отмечалось, алгоритм функционирования КА предполагает рассмотрение окружения каждой клетки и принятие в зависимости от этого решения, будет ли клетка «живой» (значение «1») или «мёртвой» (значение «0») в следующем цикле жизни КА. В частности, для БКА с 8-ю соседями (рис. 1 а), согласно правилам (рис. 2), при ЛП вычислениях для j-й строки (j-го РгП a_j) должны суммироваться 8 штук РгП:

$$(a_{(j\text{-}1)}{<<}p) + (a_{(j\text{-}1)}) + (a_{(j\text{-}1)}{>>}p) + (a_j{<<}p) +$$

$$+ (a_j{>>}p) + (a_{(j+1)}{<<}p) + (a_{(j+1)}) + (a_{(j+1)}{>>}p). \quad (6)$$

В (6) «$<<p$» и «$>>p$» означают процедуры регистрового сдвига на p бит (длину сегмента) в сторону старших или младших разрядов соответственно. В рассматриваемом случае БКА $p = 1$, а для представления результата по каждому сегменту требуется 4 бита, поскольку имеется 9 градаций возможных значений сегментной суммы: (0, 1, 2,..., 7, 8). Как отмечалось, соседние сегменты не должны взаимодействовать. Для этого ЛП система должна быть организована так, чтобы в сегментах было изначально зарезервировано достаточно места для размещения результата. Типовым подходом при реализации подобных ЛП вычислений является прореживание РгП A [2, 3] с использованием битовых масок.

$$E_1 = (000100010001...0001)_2;$$
$$E_2 = (001000100010...0010)_2;$$
$$E_3 = (010001000100...0100)_2;$$
$$E_4 = (100010001000...1000)_2. \quad (7)$$

При этом массив РгП A преобразуется в 4 массива A_i,: $(a_{i1}, a_{i2},..., a_{ij},..., a_{i29}, a_{i30})$; $a_{ij} = a_j \wedge E_i$; $i \in (1, 2, 3, 4)$; где «$\wedge$» – побитовое логическое И. В полученных 4 массивах все сегменты содержатся в точности по одному разу. При этом они не взаимодействуют при проведении операции. После прореживания процедура выполняется последовательно для каждого из A_i, с последующим сведениием результатов в единый РгП – сложением либо теоретико-множественным объединением. Эффективность ЛП алгоритма при этом снижается четырёхкратно.

Аналогично, при 6 соседях (рис. 1 б) имеется 7 градаций, а при 4 соседях (рис. 1 в) – 5 градаций, поэтому для хранения каждого из таких результатов потребуется 3 бита. Соответственно, для ЛП обработки потребуется прореживание A на три массива, с применением системы битовых масок

$$E_1 = (001001001...001)_2;$$
$$E_2 = (010010010...010)_2;$$
$$E_3 = (100100100...100)_2. \quad (8)$$

Эффективность ЛП алгоритма снижается при этом только трёхкратно.

Прореживание системой масок (7) или (8) и соответствующее кратное снижение эффективности ЛП обработки снимается полностью с применением *стекового* ЛП алгоритма.

4. Стековый локально-параллельный алгоритм

Сортировка материала, попавшего в k-й сегмент стека, реализуется «методом пузырька». Поясним работу ЛП стекового алгоритма на примере. Пусть имеются 8-битные числа

$$A_1 = 188_{10} = 10111100_2,$$
$$A_2 = 113_{10} = 01110001_2,$$
$$A_3 = 235_{10} = 11101011_2,$$
$$A_4 = 25_{10} = 00011001_2, \quad (9)$$

случайным образом выбранные в интервале от 0 до 255. Нижний индекс обозначает основание системы счисления представления числа. В двоичном представлении для наглядности лидирующие нули сохранены.

Интерпретируем двоичные позиции, как сегменты в ЛП представлении. Таким образом, выражения (9) означают, что имеется четыре 8-сегментных РгП. Размер каждого сегмента – 1 бит. Стеками являются столбцы сегментов. Имеется ЛП набор из 8 стеков. Работа данного ЛП набора стеков заключается в том, что после первоначального заполнения единичные сегменты «проваливают-

ся» максимально вниз, а нулевые, соответственно, «всплывают» вверх согласно алгоритму сортировки «методом пузырька», который применяется к РгП посегментно, одновременно (локально-параллельно) ко всем сегментам. С набором данных (9) это выглядит следующим образом:

$$
\begin{array}{cccc}
\begin{matrix}10111100\\01110001\\11101011\\00011001\end{matrix} & \}\rightarrow\{ & \begin{matrix}10111100\\01110001\\00001001\\11111011\end{matrix} & \}\rightarrow \\
\rightarrow\{\ \begin{matrix}10111100\\00000001\\01111001\\11111011\end{matrix} & \}\rightarrow\{ & \begin{matrix}00000000\\10111101\\01111001\\11111011\end{matrix} & \}\rightarrow \\
\rightarrow\{\ \begin{matrix}00000000\\10111101\\01111001\\11111011\end{matrix} & \}\rightarrow\{ & \begin{matrix}00000000\\00111001\\11111101\\11111011\end{matrix} & \}\rightarrow \\
 & \rightarrow\{ & \begin{matrix}00000000\\00111001\\11111001\\11111111\end{matrix} &
\end{array}
\qquad (10)
$$

Выражение (10) изображает последовательную смену 7 состояний набора РгП (1). Первое состояние соответствует исходному (1), последнее – ситуации с полностью отсортированными сегментными стеками: все единицы максимально смещены вниз. Сортировка (ЛП работа стеков) организована за 6 шагов – переходов между состояниями. Фигурные скобки и стрелки переходов показывают, какие именно из РгП сопоставляются на каждом следующем шаге алгоритма сортировки.

Рассмотренный процесс включает два вложенных цикла. Внешний цикл обеспечивает (n-1)-кратное (где n – число РгП, в примере $n = 4$) повторение внутреннего цикла с сокращением его объёма от (n-1) (три первые смены состояний) до 1 (последняя смена состояний).

Первый (самый правый, соответствующий младшим разрядам чисел $A_1 - A_4$) стек не претерпевал изменений, т.к. оказалось, что изначальная расстановка единиц в нём не требует сортировки. 4-е и 5-е состояния – тождественны, поскольку в результате сопоставления РгП A_3 и A_4 оказалось, что в этом цикле не происходит вертикального перемещения единиц («всплывания пузырьков»). Показателен 3-й справа разряд. В нём тремя последовательными обменами единица спускается из РгП A_1 в РгП A_4.

Рассмотрим бинарную процедуру сопоставления, обозначенную на (10) значком «→». Процедура включает арифметические операции «–» и «+», а так же побитовые (поразрядные) логические операции «END» и «XOR». Входными данными являются A и B, выходными – изменённые значения A и B. В процедуре используются константа $E_0 = 11111...11$ («ЛП единица») и промежуточная переменная M. Приведём описание процедуры в обозначениях, принятых в [7]:

```
Local_Parallel_Stack()
1  M ← A “END” ( B “XOR” E0 )          (11)
2  A ← A – M
3  B ← B + M
```

В ходе выполнения в посегментном битовом представлении для всех четырёх возможных комбинаций входных данных происходит следующее.

Исходное состояние:

$A = ...1...1...0...0...$;
$B = ...1...0...1...0...$.

Константа:

$E_0 = ...1...1...1...1...$.

Промежуточные состояния:

B “XOR” $E_0 = ...0...1...1...0...$;
$M = A$ “END” (B “XOR” E_0) $= ...0...1...0...0...$.

Состояние после завершения процедуры:

$A = ...1...0...0...0...$;
$B = ...1...1...1...0...$.

Как следует из представленного, в тех сегментных стеках, в которых имеется комбинация (1, 0), происходит преобразование (1, 0) → (0, 1), то есть при наличии «свободного места» единица «проваливается» вниз по стеку. Остальные стековые комбинации остаются без изменения.

После завершения сортировки стеков (столбцов текущего набора РгП (10)), имеется весь исходный материал для формирования результата – нового набора значений соответствующего фрагмента КА. Так, если рассматривать пример (10) как относящийся к работе БКА с четырьмя соседями (рис. 1 а), разность

$$(A_2 - A_1) = (00111001)_2 - (00000000)_2 = (00111001)_2$$

будет соответствовать правилу наличия трёх «живых» ячеек-соседей. В ней единицами отмечены те позиции (ячейки КА), которые «выжили» в новом поколении КА, поскольку в текущем поколении они были окружены тремя единичными соседями. Соответственно, нулями обозначены «не выжившие» ячейки, которые в текущем поколении были окружены меньшим либо большим числом соседей.

Аналогично, выглядит ситуация с другим числом соседей (рис. 1 б, в) и более сложным набором правил. Так, в случае БКА с 8 соседями (рис. 1 в) и «выживанием» ячейки при 3 или 4 соседях (рис. 2) стек состоит из восьми РгП $B_1, B_2,..., B_m,..., B_8$. Здесь, как и в (9), B_8 – основание стека, B_1 – его вершина. При упорядочении РгП аналогично представленному в (10), после прохождения сортировки в стеках, значение «1» в некоторой позиции m-го РгП обозначает наличие у соответствующей

ячейки не менее $(9-m)$ «единичных» ячеек-соседей. Поэтому ситуация с 3 или 4 «соседями» отображается в виде «1» в B_6 и B_5, соответственно, при наличии «0» в B_5 и B_4, соответственно. Результирующее выражение данного комбинированного решающего правила – $(B_4 - B_6)$.

5. Обсуждение результатов

Работа БКА с применением предложенного стекового алгоритма смоделирована в варианте с 8 соседями (рис. 1 а) на 32-разрядном процессоре, на языке Python. Размер поля модели 32×32 ячейки. Моделированием подтверждена работоспособность стекового алгоритма. В процессе разработки выявлены следующие особенности стекового варианта построения КА.

Позитивной стороной ЛП стекового алгоритма КА является универсальность подхода применительно к различным вариантам правил поведения КА. Универсальность проявляется в независимости от объёма и конфигурации окружения (соседства), а так же в простоте выбора порогового уровня или диапазона уровней при определении «выживающих» ячеек. Этим обеспечивается простота смены значений для ключевых параметров модели (числа ячеек-соседей, конфигурации соседства и правил перехода разметки ячеек) при экспериментальном подборе при исследовании применительно к конкретной моделируемой предметной области.

Достоинством стекового варианта алгоритма КА является существенное (в 3 – 4 раза) сокращение времени обработки по сравнению с рассмотренным (п. 3) альтернативным ЛП вариантом – с прореживанием РгП.

Ограничением выполненного моделирования ЛП стекового варианта БКА является размер поля: строка поля целиком помещена в одно РгП. Для размещения более длинных строк требуется несколько РгП и дополнительные алгоритмические решения по состыковке их концов (младшего сегмента i-й РгП и старшего сегмента $(i+1)$-й РгП) для правильной обработки соседства ячеек.

В разработанном варианте стековый алгоритм детерминировано реализует два вложенных цикла. В результате, возможны, хотя и редки, случаи «холостого» прохождения цикла, типа отмеченного выше, продемонстрированного в примере (10) – тождественность 4-го и 5-го состояний. Сокращение количества переборов за счёт изъятия этих «холостых» прохождений цикла – проблематично. Введение соответствующих дополнительных проверок только усложнит и замедлит алгоритм, поскольку возможные проверки в совокупности будут в лучшем случае соизмеримы по сложности и объёму кода с основной процедурой (11).

В связи со сказанным, последующими направлениями исследования могут явиться, в частности:

– модифицирование алгоритма на случай представления строки поля КА несколькими РгП;

– поиск оптимизационных алгоритмических решений, в частности по сокращению «холостых» циклов;

– распространение стекового алгоритма на более широкий класс КА (большее число состояний ячейки, конфигурации соседства, отличные от иллюстрируемых рис. 1, многомерные и многослойные структуры с послойными функциями влияния).

Кроме того ретроспективно целесообразны обобщения КА совместно с принципом ЛП и стековым подходом в реализации базового алгоритма:

– введение весовых коэффициентов для уровней (при задании уровня срабатывания в виде диапазона значений) и отдельных полей конфигурации окружения;

– интерпретация вводимых весовых коэффициентов как значений ФП;

– вероятностная интерпретация ФП.

Введение вероятностных представлений и элементов нечёткости – позволяет существенно разнообразить поведение КА и искать их устойчивые долгоживущие состояния (в смысле выживаемости КА) помимо зависимости от конкретных правил поведения КА.

Применительно к приложениям типа моделирования различных аспектов функционирования живой природы нечёткий и вероятностный КА подходы принципиально интересны тем, что ими вводится (учитывается, изначально задаётся) мера детерминированности (известности, измеримости, наблюдаемости) в поведении моделируемой системы. Таким образом, не исключаются из рассмотрения неизвестные влияющие факторы, всегда присутствующие в моделируемой системе.

Выводы

Локально-параллельное представление информации обеспечивает повышенную эффективность работы программного обеспечения. Принципы локально-параллельной обработки применимы к клеточным автоматам и позволяют существенно увеличить размеры модели. Предложен и продемонстрирован на тестовых примерах локально-параллельный стековый алгоритм бинарного клеточного автомата. Стековый подход в равной степени эффективен применении к различному числу и вариантом расположения ячеек-соседей; а также инвариантен по выбору порогового уровня или диапазона уровней для определения «выживающих» ячеек.

Список литературы: 1. *Бенаддия, Абдельлатиф.* Перспективы реализации келлюлярных авторматов на локально-параллельных алгоритмах [Текст] / *Бенаддия Абдельлатиф* // Материалы международной научно-практической

конференции студентов и аспирантов “Информационные технологии в экономике и образовании”. – М.: Российский университет кооперации, 2008. – С. 45–48. **2.** *Михаль, О.Ф.* Моделирование распределенных информационно-управляющих систем средствами локально-параллельных алгоритмов обработки нечеткой информации [Текст] / *О.Ф. Михаль* // Проблемы бионики: Всеукр. межвед. науч.-техн. сборник. – 2001. – Вып. 54. – С. 28–34. **3.** *Михаль, О.Ф.* Принципы организации систем нечеткого регулирования на однородных локально-параллельных алгоритмах [Текст] / *О.Ф. Михаль, О.Г. Руденко* // Управляющие системы и машины. 2001. № 3. С. 3–10. **4.** *Тоффоли, Т.* Машины клеточных автоматов [Текст] : Пер. с англ. / *Т. Тоффоли, Н. Марголус.* – М., «Мир» 1991. – 280 с. **5.** *Люггер, Д.Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем [Текст] / *Д.Ф. Люггер* М.: Изд. дом «Вильямс», 2005. – 864 с. **6.** *Любищев, А.А.* Дисперсионный анализ в биологии [Текст] / *А.А. Любищев.* – М.: Изд-во Моск. ун-та, 1986. 200 с. **7.** *Кормен, Т.* Алгоритмы: построение и анализ [Текст] / *Т. Кормен, Ч. Лейзерсон, Р. Ривест* М.: МЦНМО, 2001. – 960 с.

Поступила в редколлегию 12.09.2011

УДК 519.87

Локально-паралельний стековий алгоритм бінарного клітинного автомату / Бенаддія Абдельлатіф, О.П. Міхаль // Біоніка інтелекту: наук.-техн. журнал. – 2011. – № 3 (77). – С. 112-118.

Локально-паралельне подання інформації забезпечує підвищення эфективности работи програмного забезпечення. Додання принципів локально-паралельної обробки до клітинних автоматів дозволяє суттєво підвищити розміри моделей. Запропоновано і продемонстровано на тестових прикладах локально-паралельний стековий алгоритм бінарного клітинного автомату.

Іл. 2. Бібліогр.: 7 найм.

UDK 519.87

Local-parallel stack algorithm of the binary cellular automaton. / Benaddia Abdellatif, O.Ph. Mikhal // Bionics of Intelligense: Sci. Mag. – 2011. – № 3 (77). –P. 112-118.

Local-parallel presentation of information provides raised efficiency of the work of software. Application of principle of local-parallel processing to cellular automaton allows greatly enlarge the sizes of models. Local-parallel stacked algorithm of the binary cellular automaton is offered and demonstrated on test examples.

Fig. 2. Ref.: 7 items.