СТРУКТУРНЫЕ И СЛОЖНОСТНЫЕ ПРОБЛЕМЫ ВЫЧИСЛИМОСТИ
(Вычислительные системы)
1999 год Выпуск 165
УДК 510.1:510.51

# К АРХИТЕКТУРЕ "ХИМИЧЕСКОГО КОМПЬЮТЕРА"[1]

Е.И. Латкин

## Введение

Сегодня известно множество печатных работ, использующих метод Монте-Карло для компьютерного моделирования различных физико-химических процессов [2,6]. Обзор трудов по этой тематике можно найти в [3,8]. Настоящая работа опирается на наш собственный опыт моделирования сложных эффектов самоорганизации в определенных реакциях гетерогенного катализа [5,9,10,14,16,18].

Мы уже отмечали в [21] и [22], что существует естественное общее ядро у всех исследованных нами моделей. Мы обозначили это ядро как SCAM (Statistic Cellular Automata Machine). В работе [22] приводятся физико-химические предпосылки, оправдывающие архитектуру SCAM, а также доказывается алгоритмическая универсальность SCAM. Здесь мы дополняем это доказательство более подробными рассуждениями и доказываем также обратное утверждение о том, что поведение SCAM является алгоритмически вычислимым.

Таким образом, модель SCAM обладает достаточной выразительной силой, чтобы имитировать любые алгоритмические вычисления, и, с другой стороны, эта вычислительная сила не чрезмерна.

[1]Работа выполнена при поддержке гранта РФФИ № 99-01-0485.

Архитектура SCAM опирается на широко применяемую технику моделирования реакций гетерогенного катализа. Поэтому, мы вправе считать ее подходящей моделью для исследования высших — требующих теоретико–алгоритмического осмысления — форм самоорганизации в химических реакциях.

Аналогичные алгоритмические подходы к исследованию физических систем, демонстрирующих очень сложное поведение, упоминались в частности С.Вольфрамом (S.Wolfram) в [17]. В этом кратком обзоре обсуждается ряд примеров, когда та или иная, в принципе, простая физическая система проявляет сложное алгоритмическое поведение. В 80-е годы разными авторами были рассмотрены примеры алгоритмически универсального поведения некоторых электрических систем, модельного газа, состоящего из твердых сфер, моделей химических реакций; было показано, что проблема нахождения состояния полимера с минимальной потенциальной энергией является NP- полной. Обобщая эти примеры, С.Вольфрам формулирует область, в которой теоретико–алгоритмические методы могут быть полезны для анализа физических систем: это случай, когда в физической системе "...плотность информации конечна и может меняться лишь с конечной скоростью и в конечномерном пространстве..." [17].

Публикации по этой тематики обычно опираются на детерминированные клеточные автоматы (CA), часто применяемые для моделирования потоков газа или жидкости [3], систем химически реагирующих молекул [8], и, в общем случае, систем взаимодействующих частиц.

Дж. фон Нейман ввел клеточные автоматы в 50–е годы как модель самовоспроизводящих автоматов. Современный интерес к клеточным автоматам возродился в 80–е годы благодаря электронному устройству, названному "машиной клеточных автоматов" (CAM) и разработанному в Массачусетском технологическом институте [25]. Это устройство обладало вычислительной мощью, сравнимой с суперкомпьютером (по отношению к специальному классу задач), и в то же время было доступно по цене многим группам исследователей.

Поскольку возможности этой электронной реализации клеточных автоматов были ограничены по сравнению с автоматами

фон Неймана, необходимо было исследовать вопрос о ее алгоритмических свойствах. С.Вольфрам предложил следующую классификацию модельных физических систем, описываемых при помощи CAM: 1) стационарные, 2) периодически осциллирующие, 3) хаотические и 4) алгоритмически неразрешимые. Таким образом было установлено, что клеточные автоматы, реализуемые на CAM, достаточно сложны, чтобы описывать разнообразные системы частиц.

Вместе с тем, модели, основанные на детерминированных клеточных автоматах, скорее ближе к дифференциальным уравнениям, чем к прямой имитации сообщества частиц. Наиболее близкой аналогией клеточным автоматам служит метод квазичастиц, часто применяемый для компьютерного исследования моделей плазмы, газов, жидкостей, т.е. непрерывных сред [7]. Как и в методе квазичастиц, каждая клетка детерминированного автомата представляет большую совокупность реальных частиц, и взаимодействие между клетками определяется из усредненных взаимодействий между частицами [3,8,23,25]. Для описания такого усредненного взаимодействия необходимо учесть все три градиента плотности частиц $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, $\frac{\partial}{\partial z}$ (или только $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$ в 2-мерном случае). Разумеется, дискретный клеточный автомат огрубляет эти градиенты. Однако важнейшей чертой CA остается зависимость поведения элементарной ячейки от текущего состояния *всех* соседних ячеек.

Вместе с тем, механические и химические взаимодействия обычно вовлекают только пару молекул, поскольку вероятность тройных и более сложных столкновений относительно мала. Это означает, что для прямого описания таких процессов, каждая ячейка моделирующего автомата должна взаимодействовать только с одной из соседних ячеек в каждый определенный момент времени.

Это естественное требование легко удовлетворяется методом Монте-Карло, который лежит в основе конструкции SCAM. Однако, ограничив SCAM только парными взаимодействиями между ячейками, мы заново должны исследовать вопрос: достаточно ли таких усеченных взаимодействий для моделирования работы универсальной машины Тьюринга?

В данной работе мы даем положительный ответ на этот вопрос.

Разумеется, для такого исследования мы несколько упрощаем конструкцию SCAM, по существу выхолостив ее физико–химическое содержание. Однако это упрощение не отменяет основных выводов о теоретико–алгоритмических свойствах SCAM, но лишь избавляет наше изложение от массы несущественных деталей. Какие именно упрощения мы допускаем, более подробно обсуждается в заключении данной работы.

Определения I: один вариант статистической вычислимости

Обозначим множество целых чисел буквой $\mathbf{Z}$; и пусть символы $\mathbf{N}$, $\mathbf{Q}$ и $\mathbf{R}$ обозначают соответственно множества натуральных, рациональных и вещественных чисел в частности $\mathbf{N} \subset \mathbf{Z} \subset \mathbf{Q} \subset \mathbf{R}$ и $0 \in \mathbf{N}$.

Из теории рекурсии хорошо известно, что универсальная машина Тьюринга способна вычислить любую алгоритмически вычислимую функцию над натуральными числами [15]. Авторы [4] (ссылаясь на работу [12]) используют тот факт, что так называемые "программируемые" машины с двумя регистрами могут моделировать любую, в том числе универсальную машину Тьюринга.

Нам понадобится следующее определение программируемой машины из [4], которое мы воспроизводим.

ОПРЕДЕЛЕНИЕ 1. Программируемая машина состоит из пары регистров $\{A, B\}$, способных хранить натуральные значения (память машины), и из конечной последовательности инструкций $< I_1, \ldots, I_N >$ (программа). Работа машины начинается с первой инструкции и с заданного извне начального состояния памяти и состоит в пошаговом редактировании содержимого регистров в соответствии с текущей инструкцией программы. Полный список инструкций, из которых может состоять программа, включает:

1) $A^+$ (or $B^+$) — прибавить единицу к регистру $A$ (или $B$),

2) $A^-[n]$ (or $B^-[n]$) вычесть единицу из $A$, если $A > 0$, или перейти к инструкции номер $n$, если $A = 0$ (или то же самое для регистра $B$),

3) $Go[n]$ — безусловный переход к инструкции номер $n$,

4) $Halt$ — прекратить вычисления.

Назовем программируемую машину универсальной, если она моделирует универсальную машину Тьюринга. Вслед за авторами [4] мы построим клеточный автомат, имитирующий работу универсальной программируемой машины. Однако, в отличие от [4], наш клеточный автомат будет статистическим.

Вычисление интервалов времени, соответствующих циклам работы SCAM, вообще говоря, требует операций над вещественными числами. Однако практически, иррациональные числа никогда не используются для измерения физических величин. Поэтому мы ограничимся лишь рациональными числами, поскольку такое ограничение не является существенным и позволяет исследовать вычислительные характеристики SCAM в рамках стандартной теории алгоритмов и теории нумераций [15,19].

Зафиксируем некоторую нумерацию $\gamma : \mathbf{N} \to \mathbf{Q}$ множества рациональных чисел $\mathbf{Q}$ таким образом, чтобы свойства $\gamma_n = \gamma_m$, и $\gamma_n = m$ были алгоритмически разрешимыми по номерам $(n, m)$. Тогда частичная функция $f : \mathbf{Q}^K \to \mathbf{Q}$ называется вычислимой, если существует такая частичная рекурсивная функция $\phi$, что:

1) $\mathrm{dom}(\phi) = \{(n_1, \ldots, n_K) | f(\gamma n_1, \ldots, \gamma n_K)$ *определено*$\}$ и

2) $(\forall (n_1, \ldots, n_K) \in \mathrm{dom}(\phi)) f(\gamma n_1, \ldots, \gamma n_K) = \gamma\phi(n_1, \ldots, n_K)$.

Мы требуем, чтобы основные арифметические операции $\{+, -, \cdot, /\}$ над рациональными числами были вычислимыми. Существует единственная (с точностью до эквивалентности) нумерация поля рациональных $\mathbf{Q}$ чисел, удовлетворяющая этим требованиям — Гёделева нумерация (Göedel) [20].

Заметим, что всякая частичная рекурсивная функция $f : \mathbf{N}^K \to \mathbf{N}$ является также вычислимой частичной функцией $f : \mathbf{Q}^K \to \mathbf{Q}$ над полем $\mathbf{Q}$. Обозначим класс вычислимых функций буквой $\mathbf{C}$. Очевидно, можно рассуждать об алгоритмах непосредственно в терминах рациональных чисел $\gamma n \in \mathbf{Q}$, подразумевая их натуральные коды $n \in \mathbf{N}$ лишь неявно.

Для дальнейшего требуется обогатить эту стандартную вычислимость операциями со случайными величинами, общепринятыми при расчетах методами Монте–Карло.

Как правило, алгоритмы метода Монте–Карло опираются на бесконечную последовательность $\alpha_0, \alpha_1, \ldots$ независимых в совокупности случайных величин, равномерно распределенных в интервале $0 < \alpha_n < 1$ [24]. Мы не можем оперировать непосредственно величинами $\alpha_0, \alpha_1, \ldots$, поскольку они, вообще говоря, могут принимать иррациональные значения. Рассмотрим рациональный случайный предикат $\varphi := \{(n, x) \in \mathbf{N} \times \mathbf{Q} | \alpha_n < x\}$. Иными словами, мы фиксируем некоторое вероятностное пространство $(\Omega, \mathbf{P})$ и отображение $\omega \in \Omega \mapsto \varphi_\omega \subseteq \mathbf{N} \times \mathbf{Q}$ такое, что вероятность $\mathbf{P}\{\omega | (n, x) \in \varphi_\omega\} = x$ для всех $n \in \mathbf{N}$ и $x \in (0, 1)$. (Не указанная здесь явно $\sigma$-алгебра измеримых множеств совпадает с областью определения $\mathrm{dom}(\mathbf{P})$.)

Для каждого $\omega \in \Omega$ определим класс $\mathbf{C}_\omega$ тех функций над $\mathbf{Q}$, которые сводятся по Тьюрингу (T-сводятся) к предикату $\varphi_\omega$. Тьюрингова сводимость означает, что найдется такой алгоритм $A$, вычисляющий функцию $f \in \mathbf{C}_\omega$, который использует оператор ветвления "*if...then...else*" с предикатом $\varphi_\omega$. Этот же алгоритм $A$ может выдать другой результат (или вообще не выдать результата), если его применить к другому оракулу $\varphi_{\overline{\omega}}, \overline{\omega} \in \Omega$. Обозначим такую зависимость результата вычислений при помощи записи $f_\omega(x_1, \ldots, x_K) = A[\varphi_\omega](x_1, \ldots, x_K)$. Мы рассмотрим особый класс таких алгоритмов, что $f_\omega(x_1, \ldots, x_K) = const$ для почти всех $\omega \in \Omega$.

ОПРЕДЕЛЕНИЕ 2. Обозначим через $\mathbf{C}^*$ класс всех таких функций $f_\omega(x_1, \ldots, x_K) = A[\varphi_\omega](x_1, \ldots, x_K)$, что для любых $x_1, \ldots, x_K \in \mathbf{Q}$ либо $(\exists y \in \mathbf{Q}) f_\omega(x_1, \ldots, x_K) = y$ для почти всех $\omega \in \Omega$, либо значение $f_\omega(x_1, \ldots, x_K)$ не определено для почти всех $\omega \in \Omega$. Назовем функции $f \in \mathbf{C}^*$ вероятностно вычислимыми. Далее обозначим через $\mathbf{A}^*$ класс всех таких алгоритмов $A$, что функция $f_\omega(x_1, \ldots, x_K) = A[\varphi_\omega](x_1, \ldots, x_K)$ является вероятностно вычислимой. Назовем $A \in \mathbf{A}^*$ вероятностными алгоритмами.

Известно, что некоторые естественные понятия вероятностной вычислимости оказываются эквивалентными классической вычислимости (см. например обзор [26], который ссылается на [11]). Это также верно для определенного нами класса, т.е. $\mathbf{C}^* = \mathbf{C}$. Мы докажем эту эквивалентность, рассматривая дерево вычислений для произвольного алгоритма $A \in \mathbf{A}^*$ и списка ар-

гументов $x_1, \ldots, x_K \in \mathbf{Q}$. (Определение дерева вычислений будет дано внутри доказательства.)

**ТЕОРЕМА** 1. *Каждая вероятностно вычислимая функция $f \in \mathbf{C}^*$ вычислима при помощи некоторого стандартного (детерминированного) алгоритма, т.е. $f \in \mathbf{C}$.*

ДОКАЗАТЕЛЬСТВО. Вместо строгого кодирования машин Тьюринга, мы воспользуемся следующей упрощенной системой записи алгоритмов со случайным оракулом.

1. Допустимы все термы, включающие базовые арифметические операции $\{x+y, x-y, x \cdot y, x/y\}$, рациональные константы и переменные. Предполагается, что результат вычисления терма не определен, если для его вычисления потребуется выполнить деление на ноль.

2. Можно использовать оператор присваивания "$\nu := t$", где $\nu$ — переменная и $t$ — терм.

3. Полный список всех переменных состоит из $\nu[0], \nu[1], \ldots$; при этом разрешается доступ к памяти по индексу. Последнее означает, что допустимы конструкции вида "$\nu[t]$", где $t$ — произвольный терм. Обычно мы пишем $x, y, z$ или другую букву вместо $\nu[i]$, если $i = const$.

4. Когда некоторая программа $A$ стартует со списком аргументов $x_1, \ldots, x_K \in \mathbf{Q}$, начальное состояние памяти загружается значениями $\nu[0] = K$, $\nu[1] = x_1, \ldots, \nu[K] = x_K$, а остальные переменные $\nu[i] = 0$.

5. Программа состоит из операторов присваивания и одного или нескольких операторов остановки "*halt*", структурированных при помощи конструкций: последовательного исполнения "*begin* $\ldots;\ldots;\ldots$ *end*", условного ветвления "*if* $\ldots$ *then* $\ldots$ *else* $\ldots$", и цикла "*while* $\ldots$ *do* $\ldots$".

6. Конструкции "*if* $\ldots$ *then* $\ldots$ *else* $\ldots$" и "*while* $\ldots$ *do* $\ldots$" могут использовать предикаты "$t < s$" или "$t = s$", построенные из произвольных термов $t$ и $s$.

7. В случае вероятностных алгоритмов, допускается дополнительная форма вероятностных предикатов "$\varphi(t, s)$" (т.е. $\alpha_t < s$), которую можно применять в конструкции "*if* $\ldots$ *then* $\ldots$ *else* $\ldots$". (Разумеется текущее значение терма $t$ должно принадлежать множеству натуральных чисел $\mathbf{N} \subset \mathbf{Q}$.)

8. Когда программа останавливается (если она останавливается), результат вычисления считывается из ячейки памяти $\nu[0]$.

Рассмотрим произвольную функцию $f \in \mathbf{C}^*$ и вероятностный алгоритм $A \in \mathbf{A}^*$ такой, что $f(x_1, \ldots, x_K) = A[\varphi_\omega](x_1, \ldots, x_K)$ для почти всех $\omega \in \Omega$. Для фиксированных произвольных параметров $x_1, \ldots, x_K \in \mathbf{Q}$ определим дерево вычислений $T_A(x_1, \ldots, x_K)$.

Корень дерева $r$ кодирует начальное состояние виртуального автомата, вычисляющего $A(x_1, \ldots, x_K)$. Корню приписывается вероятность $P(r) = 1$.

Далее по уже имеющемуся узлу $q$ определим рекурсивно все исходящие из него ветви. Исходящих ветвей может быть одна, две или ни одной, если узлу соответствует команда остановки "*halt*", или если результат исполнения приписанной к узлу инструкции не определен.

Из узла исходят две ветви, если выполняются следующие условия:

1) к узлу приписана инструкция "*if* $\alpha_t < s$ *then*... *else*..." (т.е. "*if* $\varphi(t, s)$..."),

2) условие $\alpha_t < s$ является нетривиальным, т.е. $t \in \mathbf{N}$ и $0 < s < 1$,

3) не существует конструкции "*if* $\alpha_u < w$ *then*... *else*...", приписанной к какой-либо из вершин дерева, предшествующих данной, при которой значение терма $u$ равно $u = t$.

При выполнении 3-го условия определяются две исходящие из узла $q$ дуги, ведущие в вершины $q_<$ и $q_\geq$, соответствующие выбору "*then*..." и соответственно "*else*...". Поскольку $\alpha_t$ в данном случае не зависит от предыдущих ветвлений программы, мы приписываем новым узлам вероятности $P(q_<) = P(q) \cdot s$ и $P(q_\geq) = P(q) \cdot (1 - s)$.

Из узла $q$ также могут исходить две ветви, если условия 1 и 2 выполняются, но условие 3 нарушено. В этом случае обозначим через $n$ номер $t = u_1 = \ldots = u_N$ случайной величины $\alpha_n$, которая используется в узле $q$ для выбора ветви, и уже была использована с этой же целью в предыдущих узлах $q_1, \ldots, q_N$. Рассмотрим значения $w_1, \ldots, w_N$ всех тех термов, с которыми

$\alpha_n$ сравнивалась в узлах $q_1, \ldots q_N$. Предположим, что узел $q$ соответствует значению $w' \leq \alpha_n < w''$, где две величины $w', w'' \in \in \{w_1, \ldots, w_N\}$ выбраны ближайшими друг к другу. Если $s < w'$ или $s \geq w''$, то ветвление "*if* $\alpha_i < s$ *then* ... *else* ..." на самом деле является детерминированным, и только одна дочерняя ветка исходит из узла $q$. Если же $w' \leq s < w''$, то $q$ порождает два подчиненных узла $q_<$ и $q_\geq$, которым приписываются вероятности $P(q_<) = P(q) \cdot (s - w')$ и $P(q_\geq) = P(q) \cdot (w'' - s)$.

Если инструкция программы, приписанная к узлу $q$ детерминированная, или вероятностный выбор, результат которого предопределен как описано выше, узел $q$, как правило, порождает одну исходящую ветвь к новому узлу $q'$, которому соответствует следующая инструкция программы и приписывается вероятность $P(q') = P(q)$.

Наконец узел $q$ может оказаться финальным узлом дерева $T_A(x_1, \ldots, x_K)$, не порождающим исходящих ветвей (листом). Такое случается, если текущая инструкция программы (приписанная к узлу $q$) есть инструкция "*halt*", или если текущая инструкция не может быть выполнена — значение одного или нескольких входящих в нее термов или предиката $\varphi(t, s)$ не определено: встретилось деление на нуль, либо индексирование $\nu[i]$ или $\varphi(i, s)$ с индексом $i \notin \mathbf{N}$.

Инструкция "*halt*" соответствует продуктивному окончанию программы, и сигнализирует, что в регистре $\nu[0]$ вычислен ее результат. Окончание программы по другим причинам не результативно, т.е. значение результата вычислений здесь считается неопределенным, как если бы программа не останавливалась вовсе (зациклилась).

В простейшем случае, если алгоритм $A$ строго детерминирован, его дерево вычисления $T_A(x_1, \ldots, x_K)$ является линейным, т.е. и состоит из единственной ветви $q \to q' \to q'' \to \ldots$, конечной или бесконечной. Каждому из узлов этой ветви приписывается вероятность 1, и это совпадает с вероятностью того, что вычисление пойдет именно по этой ветви.

В общем случае, рассмотрим произвольную ветвь $b = \{q_0 \to q_1 \to \ldots\}$, либо оканчивающуюся финальной вершиной (листом дерева вычислений), либо бесконечную. Вероятность

этой ветви $P(b)$ определяется как вероятность $P(q_s)$ ее последнего узла (листа), если она имеется, либо как предел $P(b) = \lim\limits_{s\to\infty} P(q_s)$, если ветвь бесконечна. Разумеется, этот предел может обращаться в нуль, несмотря на то, что по определению $P(q_s) > 0$ на всех уровнях $s \in \mathbf{N}$.

Определим $T_A^s(x_1, \ldots x_K)$ как поддерево дерева $T_A(x_1, \ldots, x_K)$, содержащее все его вершины вплоть до уровня $s \in \mathbf{N}$. (Уровень $s$ узла $q$ однозначно определяется по любой проходящей через него ветви, т.е. такой ветви $b = \{q_0 \to q_1 \to \ldots\}$, что $q = q_s$ для некоторого $q_s \in b$.) Поддерево $T_A^s(x_1, \ldots, x_K)$ конечно, и можно вычислить его гёделев номер по гёделевым кодам $A$ и $x_1, \ldots, x_K$. (Стандартная техника кодирования по Гёделю (Göedel) описана, например, в [15, 19, 20, 1].)

Рассмотрим процесс построения поддеревьев $T_A^s(x_1, \ldots, x_K)$ для $s = 0, 1, \ldots$, и далее. Узлы дерева $T_A(x_1, \ldots, x_K)$, которые завершают вычисление $A(x_1, \ldots, x_K)$ можно распознать рекурсивным образом, и значит весь процесс построения поддеревьев можно определить рекурсивно по кодам алгоритма $A$ и набора параметров $x_1, \ldots, x_K$. Обозначим через $B$ следующий (детерминированный) алгоритм: "Перебираем последовательно узлы дерева $T_A(x_1, \ldots, x_K)$ этаж за этажом, как описано выше. Если встретится любой узел $q$, соответствующий инструкции "*halt*", перебор останавливается, и результат, соответствующий узлу $q$, объявляется результатом вычисления. Если дерево $T_A(x_1, \ldots$ $\ldots, x_K)$ конечно и не содержит результативных узлов (листьев), входим в бесконечный цикл, чтобы промоделировать отсутствие результата вычислений."

Тогда, $B(x_1, \ldots, x_K)$ вычисляет в точности функцию $f(x_1, \ldots$ $\ldots, x_K)$.

Действительно: предположим, что для некоторого набора параметров $x_1, \ldots, x_K$ значение $f(x_1, \ldots, x_K)$ не определено. Тогда результат $A[\varphi_\omega](x_1, \ldots, x_K)$ является неопределенным для почти всех $\omega \in \Omega$. По построению, вероятность любого узла дерева вычисления $T_A(x_1, \ldots, x_K)$ является строго положительной величиной. Поэтому $T_A(x_1, \ldots, x_K)$ не содержит нерезультативной финальной ветви. В силу неопределенности $f(x_1, \ldots, x_K)$, дерево $T_A(x_1, \ldots, x_K)$ не содержит также и результ-

тативных листьев (соответствующих инструкции "*halt*"). Следовательно, процесс перебора узлов по алгоритму $B(x_1, \ldots, x_K)$ никогда не прекратится.

Предположим теперь, что значение $f(x_1, \ldots, x_K)$ определено и равняется $y$. Тогда существует некоторая вершина дерева $T_A(x_1, \ldots, x_K)$, соответствующая инструкции "*halt*" программы $A$ и возвращающая результат $\nu[0] = y$. Вероятность этой вершины, как и любой другой, — строго больше нуля. Поэтому, любая другая результативная вершина возвращает то же самое значение $\nu[0] = y$. Следовательно, вычисления по алгоритму $B(x_1, \ldots, x_K)$ завершается за конечное время, остановившись на любой из результативных вершин, и вернет нужный результат. ■

Свойство вероятностных алгоритмов возвращать определенный результат с вероятностью 1 идеализирует важную особенность алгоритмов метода Монте-Карло, обычно применяемых для моделирования каталитических процессов. Моделируемые физико-химические процессы демонстрируют по существу схожее поведение при различных случайных флуктуациях, хотя и не в точности схожее.

Неизбежный разброс в поведении физический системы является обычно пренебрежимым при имитационном компьютерном моделировании. Физически, здесь требуется не столько точный результат, сколько соответствие качественных характеристик процесса: периода и формы колебаний скорости реакции, формы и характер движения волн плотности реактантов и тому подобное [5, 9, 16, 18]. Такую информацию дает исследование единственной — по существу любой ветви алгоритма, соответствующей некоторой последовательности выборочных значений $\alpha_0(\omega), \alpha_1(\omega), \ldots, \ \omega \in \Omega$.

Такая определенная ветвь дерева вычислений называется реализацией алгоритма метода Монте-Карло. Понятие "реализация" является ключевым для метода Монте-Карло. Мы распространим его также на идеализированные вероятностные алгоритмы вместе с процедурой случайного "выбора", которая позволяет не упоминать явно параметр реализации $\omega \in \Omega$. По существу, процедура случайного выбора подчеркивает инвариантность струк-

туры дерева вычислений относительно того, или иного вероятностного пространства $(\Omega, \mathbf{P})$, и дает свободу использовать различные "генераторы" случайных чисел.

Случайный выбор в алгоритмах метода Монте–Карло описывается фразами вроде: "$\ldots l$ выбирается равномерно из множества $\{0, 1, \ldots, L-1\}\ldots$". Это означает, что $l$ является случайной величиной, определяемой как $l := \lfloor \alpha_n(\omega) \cdot L \rfloor$, причем если величина $\alpha_n(\omega)$ не была использована на предыдущих шагах алгоритма. Мы покажем корректность подобного стиля в следующем параграфе, и будем пользоваться им для описания также и вероятностных алгоритмов.

### Определения П: машина стохастических клеточных автоматов

Рассмотрим бесконечную 2-мерную решетку, состоящую из ячеек квадратной формы (как школьная тетрадь). Перенумеруем ячейки решетки при помощи пар целых чисел $(i, j) \in \mathbf{Z} \times \mathbf{Z}$. (Подобные ячейки служат для моделирования активных центров на поверхности катализатора.)

Предположим, что существует конечное число определенных состояний, в которых может находиться каждая ячейка решетки. (Например, различные состояния ячейки могут соответствовать различным сортам молекул, адсорбированных на соответствующем активном центре катализатора.) Мы перенумеруем эти состояния при помощи чисел $0, 1, \ldots, X$, и обозначим множество всех допустимых состояний $\{0, \ldots, X\}$ буквой $\mathbf{X}$.

Нам потребуется отображение $\xi(i, j)$, ставящее в соответствие каждой ячейке $(i, j) \in \mathbf{Z} \times \mathbf{Z}$ ее текущее состояние $\xi(i, j) \in \mathbf{X}$. Функция $\xi(i, j)$ зависит также от времени $t \in \mathbf{R}$, и мы используем подиндекс $\xi_t(i, j)$ чтобы обозначить эту зависимость. Значение $\xi_t(i, j)$ является случайной величиной, т.е. оно зависит также от неявного параметра $\omega \in \Omega$.

Поскольку мы намерены исследовать алгоритмические свойства случайного процесса $\xi_t(i, j)$, будет удобнее определить этот процесс через реализующий его алгоритм метода Монте–Карло (вместо определения через "master-equation", как, например, в [2, 6]).

В частности, это позволит нам считать, что $t \in \mathbf{Q}$.

Физические процессы, описываемые при помощи $\xi_t(i,j)$, определяются конечным набором элементарных "стадий" моделируемой химической реакции. (Например, как стадия окисления $ZCO + ZO \to CO_2 + 2Z$, вовлекающая два соседних адсорбционных центра $Z + Z$ на поверхности катализатора.) Каждой стадии реакции мы поставим в соответствие определенную частичную функцию $K : X^2 \to X^2$, показывающую, каким образом эта стадия может преобразовать состояние пары соседних ячеек. Эта функция может быть не определена для некоторых состояний $(x,y)$, что означает неосуществимость "химической реакции" между этими состояниями. Перенумеруем стадии реакции числами $1,\ldots,M$, $M \geq 1$, и обозначим правила преобразования для стадий через $K_1,\ldots,K_M$.

Алгоритм Монте-Карло имитирует серию случайных термодинамических флуктуаций, активирующих элементарные химические события (например, как адсорбция $CO + Z \to ZCO$, десорбция $ZCO \to CO + Z$, реакция и пр.). При этом, для момента времени $t = 0$ извне задается начальное состояние $\xi_0(i,j)$. Далее, моделируется некоторая *реализация* случайного процесса $\xi_t(i,j)$ при помощи последовательности случайно выбираемых трансформаций функции состояния $\xi_t(i,j) \to \xi_{t+\Delta t}(i,j)$.

Обсудим алгоритм *выбора* этих трансформаций и величины шага $\Delta t$.

Рассмотрим сначала упрощенный случай, когда модельная решетка конечна. В этом случае, шаг по времени $\Delta t$ является постоянным, и каждое преобразование $\xi_t(i,j) \to \xi_{t+\Delta t}(i,j)$ называется шагом метода Монте-Карло, или МК-шагом. Поскольку мы здесь не интересуемся детально физикой этих процессов, можно упрощенно положить $\Delta t = 1$.

Пусть $I, J \in \mathbf{N}\backslash\{0\}$ — некоторые достаточно большие положительные целые числа. Предположим, что все ячейки $(i,j)$ принадлежат множеству $\mathbf{Z}_I \times \mathbf{Z}_J$, где конечное множество $\mathbf{Z}_I = \left\{ -\left\lfloor \frac{I}{2} \right\rfloor, \ldots, \left\lfloor \frac{I-1}{2} \right\rfloor \right\}$ и $\lfloor x \rfloor$ есть целая часть числа $x$. Такая модельная решетка имеет форму прямоугольника, состоящего в точности из $I \times J$ ячеек. Ячейка (0,0) лежит в центре этого прямоугольника, если оба числа $I, J$ нечетные. Мы будем подразумевать, что границы прямоугольника замкнуты друг на друга при

помощи периодического условия $i + I \equiv i$ и $j + J \equiv j$. В частности это означает, что ячейки $\left(i, -\left\lfloor\frac{J}{2}\right\rfloor\right)$ и $\left(i, \left\lfloor\frac{J-1}{2}\right\rfloor\right)$ считаются ближайшими соседями так же, как ячейки $\left(-\left\lfloor\frac{I}{2}\right\rfloor, j\right)$ и $\left(\left\lfloor\frac{I-1}{2}\right\rfloor, j\right)$. (Внутри прямоугольника, ячейки $(i_0, j_0)$ и $(i_1, j_1)$ считаются соседними, если $|i_0 - i_1| = 1 \wedge j_0 = j_1$, или если $i_0 = i_1 \wedge |j_0 - j_1| = 1$.)

Отметим, что хотя прямоугольник содержит только $N = I \times J$ ячеек, здесь находятся $2 \cdot N$ пар ближайших соседей. (Мы не различаем симметричные пары $(i_0, j_0)$, $(i_1, j_1)$ и $(i_1, j_1)$, $(i_0, j_0)$.) Соответственно каждый МК–шаг состоит из $2 \cdot N$, так называемых, "попыток" совершить трансформацию состояния пары соседних ячеек согласно той или иной стадии реакции. Пара соседей выбирается каждый раз случайно, и в среднем за МК–шаг каждая из пар ячеек однажды вовлекается в попытку трансформации. Перенумеруем эти попытки числами $l = 0, \ldots, L-1$, где $L = 2 \cdot N$. Каждая попытка слегка преобразует функцию состояния решетки $\xi \to \xi'$. Обозначим $\xi_t^0 = \xi_t$ и $\xi_t^{l+1} = (\xi_t^l)'$. Тогда по определению $\xi_t^L = \xi_{t+\Delta t}$, т.е. МК–шаг $\xi_t \to \xi_{t+\Delta t}$ состоит в накоплении этих преобразований.

Рассмотрим теперь алгоритм микро–шага $\xi \to \xi'$. Пара вовлекаемых соседних ячеек выбирается равномерно среди $2 \cdot N$ пар. Предположим, что была выбрана пара $(i_0, j_0)$ и $(i_1, j_1)$, и что текущие состояния этих ячеек равны $\xi(i_0, j_0) = x$ и $\xi(i_1, j_1) = y$. Далее, выбирается некоторый номер "стадии реакции" $m$ равномерно среди $\{1, \ldots, M\}$. Эти два выбора — пары ячеек и номера стадии — осуществляются статистически независимо друг от друга и независимо от предыдущих шагов и микро–шагов метода Монте–Карло. Если преобразование $K_m(x, y)$ определено и его результат равен $(x', y')$, мы полагаем $\xi'(i_0, j_0) = x'$ и $\xi'(i_1, j_1) = y'$. Если же значение функции $K_m(x, y)$ не определено, текущая попытка не засчитывается (признается нерезультативной), и мы полагаем $\xi' = \xi$.

Каждой такой микро–трансформации $\xi \to \xi'$, не важно, результативной или нет, соответствует микро–сдвиг по времени $\delta t = \dfrac{\Delta t}{L}$.

Для исследования автоматов с бесконечной решеткой мы рассматриваем микро шаги $\delta t$, не пытаясь группировать их в МК–шаг $\Delta t$. (В общем случае это невозможно.)

ОПРЕДЕЛЕНИЕ 3. Рассмотрим некоторые числа $X, M \in \mathbf{N}$, причем $M > 0$. Обозначим $\mathbf{X} = \{0, \ldots, X\}$, и пусть для каждого $m = 1, \ldots, M$, задано частичное отображение $K_m : \mathbf{X}^2 \to \mathbf{X}^2$, а также начальное состояние решетки $\xi_0 : \mathbf{Z} \times \mathbf{Z} \to \mathbf{X}$. Кортеж $(X, M, \{K_m\}_{m=1,\ldots,M}, \xi_0)$ называется статистическим клеточным автоматом (СКА, или SCA по–английски), если выполнены следующие условия компактности:

- $\xi_0(i,j) = 0$ почти всюду на решетке, за исключением, быть может, лишь конечного числа ячеек $(i,j) \in \mathbf{Z} \times \mathbf{Z}$, и
- $K_m(0,0)$ не определено для всех $m = 1, \ldots, M$.

Машина статистических клеточных автоматов (СКАМ, или SCAM по–английски) определяется как следующий алгоритм функционирования компактных статистических клеточных автоматов, рассматриваемый вместе с некоторым "генератором" случайных чисел $\alpha_0, \alpha_1, \ldots$.

Пусть $(X, M, \{K_m\}_{m=1,\ldots,M}, \xi_0)$ — некоторый SCA и $\alpha_0, \alpha_1, \ldots$ - генератор случайных чисел.

Рассмотрим реализацию $\xi_t(\omega)$ стохастического процесса $\xi_t$, определяемую некоторым значением параметра $\omega \in \Omega$, где $(\Omega, \mathbf{P})$ — вероятностное пространство генератора случайных чисел. Параметр, обычно $\omega$, подразумевается при описании алгоритмов метода Монте–Карло, но упоминается явно. Такие фразы как "...$m$ выбирается случайно..." подразумевают использование фиксированной реализации случайных чисел $\alpha_0(\omega), \alpha_1(\omega), \ldots$ (такая реализация называется генератором *псевдослучайных* чисел).

Реализация процесса $\xi_t$ состоит из последовательности критических моментов времени $t_0 < t_1 < \ldots$ и соответствующих функций состояния $\xi^{(0)}, \xi^{(1)}, \ldots$ таких, что $t_0 = 0$ и $\xi^{(0)} = \xi_0$. Функция состояния $\xi_t$ является постоянной в промежутке от одного критического момента $t_s$ до следующего $t_{s+1}$, и если обозначить через $\xi^{(0)}$ состояние функции $\xi_t$ в момент времени $t = t_s$, то $\xi_t = \xi^{(s)}$ для всех $t$ в интервале $t_s \leq t < t_{s+1}$. Каждый критический момент $t_{s+1}$, $s \in \mathbf{N}$, соответствует микро–шагу метода Монте–

Карло $\xi^{(s)} = \xi \to \xi' = \xi^{(s+1)}$, состоящему в попытке трансформации текущего состояния случайно выбранной пары соседних ячеек при помощи случайно выбранного правила $K_m$, $m = 1, \ldots, M$.

В отличие от случая, когда решетка конечна, здесь дистанция $\delta t_s = t_{s+1} - t_s$ зависит от текущего состояния решетки $\xi^{(s)}$ и определяется *нетривиальной* частью функции $\xi^{(s)}$. Обозначим через $L(s)$ количество всех нетривиальных пар соседних ячеек, т.е. таких пар $(i_0, j_0), (i_1, j_1) \in \mathbf{Z} \times \mathbf{Z}$, что состояние хотя бы одной из двух ячеек нетривиально — либо $\xi^{(s)}(i_0, j_0) \neq 0$, либо $\xi^{(s)}(i_1, j_1) \neq 0$. По определению компактного SCA, число $L(0) < \infty$, и $L(s) \leq L(0) + 3 \cdot s$, поскольку каждый микро-шаг может дать не более одной новой нетривиальной ячейки, и соответственно — не более трех новых нетривиальных пар. Следовательно, для любого $s$ число $L(s)$ конечно, и мы можем положить $\delta t_s = \dfrac{\Delta t}{L(s)}$. (Параметр $\Delta t$ физической скорости реакции определяется ее спецификой, и мы положим его равным $\Delta t = 1$, так же, как и в случае с конечной решеткой.)

Алгоритм микро-шага $\xi^{(s)} \to \xi^{(s+1)}$ для бесконечной решетки совпадает с алгоритмом для конечной решетки, отличаясь лишь способом выбора пары вовлекаемых соседних ячеек.

В бесконечном случае, пара ячеек выбирается равномерно не по всей решетке, а лишь по нетривиальной части текущей конфигурации $\xi^{(s)}$. Зафиксируем простую нумерацию $\pi : \mathbf{N} \to \mathbf{L}$ всех пар ячеек решетки $\mathbf{Z} \times \mathbf{Z}$, показанную на рис.1. (Ячейки показаны большими квадратами с индексами $(i, j)$, и пары соседних ячеек обозначены отрезками между ними. Эти отрезки (пары) маркированы небольшими кружками с номерами пар.) Тогда для каждого $s$ определено множество $\{p_1, \ldots, p_L\}$ номеров, соответствующих всем парам ячеек, нетривиальным согласно функции текущего состояния $\xi^{(s)}$, где $L = L(s)$. Номер $p_l$ вовлекаемой на микро-шаге $s \to s + 1$ пары $\pi p_l$ выбирается равномерно среди множества $\{p_1, \ldots, p_L\}$. Затем, с вероятность $\dfrac{1}{2}$ выбирается упорядочение: будем ли мы считать, что $\pi p_l = ((i_0, j_0), (i_1, j_1))$, либо что $\pi p_l = ((i_1, j_1), (i_0, j_0))$.

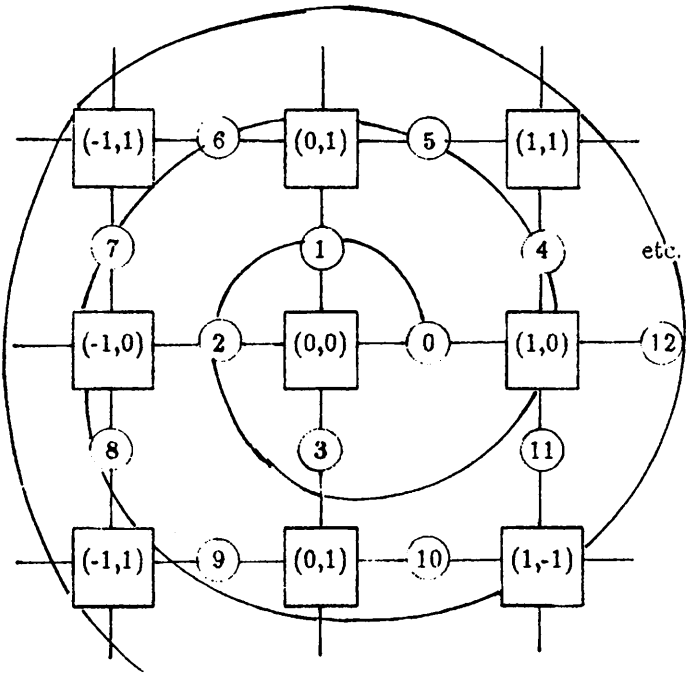


**Рис.1. Спиральный обход всех пар ячеек бесконечной решетки $\mathbf{Z} \times \mathbf{Z}$.**

Покажем, каким образом этот алгоритм T–сводится к предикату $\varphi_\omega$, где $\varphi_\omega = \{(n,x) \in \mathbf{N} \times \mathbf{Q} | \alpha_n(\omega) < x\}$. На каждом микро-шаге $\xi^{(s)} \to \xi^{(s+1)}$, необходимо выбирать номер $l$ среди $\{0,\ldots,L(s)-1\}$, чтобы выбрать одну из множества пар $\{\pi p_1,\ldots,\pi p_{L(s)}\}$, нетривиальных в текущей конфигурации $\xi^{(s)}$. Простейшая формула $l := \lfloor \alpha_n \cdot L(s) \rfloor$ не допустима, поскольку доступны только операции над рациональными числами.

Рассмотрим следующий вероятностный алгоритм выбора, опирающийся на рациональный предикат $\varphi(n,x)$ вместо вещественного числа $\alpha(\omega)$ :

```
procedure Choose_Some_l (integer n, L);
        assert L > 0;
begin integer l;
        rational x;
        x := 1/L;
        l := 0;
        while φ(n, x) do begin
                x := x + 1/L;
                l := l + 1;
        end while;
        return l;
end
```

Каждому значению параметра $\omega \in \Omega$ соответствует определенная ветвь дерева вычислений этого алгоритма. Предположим, что случайная величина $\alpha_n$ не была использована на этой ветви вычислений до рассматриваемого вызова процедуры $Choose\_Some\_l(n, L)$. Тогда, число $l$ здесь выбирается равномерно на множестве $\{0, \ldots, L-1\}$.

Обеспечить каждый такой вызов новым псевдослучайным числом $\alpha_n(\omega)$ совсем не сложно, если $\omega \in \Omega$ фиксировано. Зарезервируем специальный счетчик $n$, и положим $n := 0$ в начале первого шага имитации SCA. После каждого микрошага $\xi^{(s)} \to \xi^{(s+1)}$ достаточно положить $n := n+3$. Тогда число $\alpha_n(\omega)$, $n = 3s$, можно использовать для выбора $l(s)$ на шаге $\xi^{(s)} \to \xi^{(s+1)}$. Числа $\alpha_n(\omega)$, где $n$ имеет вид $3s+1$, используются для выбора ориентации вовлекаемой пары ячеек, и $n = 3s+2$ — для выбора стадии реакции $m \in \{1, \ldots, M\}$.

## Вычислимость статистических клеточных автоматов

Мы определили машину статистических клеточных автоматов (SCAM) через Тьюрингову сводимость к предикату $\varphi_\omega$, равному $\{(n, x) \in \mathbf{N} \times \mathbf{Q} | \alpha_n(\omega) < x\}$, где $\alpha_0, \alpha_1, \ldots$ — случайные числа и $\omega \in \Omega$ — неявный параметр из вероятностного пространства $(\Omega, \mathbf{P})$. В общем случае, предикат $\varphi_\omega$ не рекурсивен, — это одна из трудностей метода Монте-Карло. К счастью, дерево вычислений статистического клеточного автомата в целом устроено

гораздо проще, чем отдельные реализации его поведения. Аналогично теореме 1, утверждающей обычную вычислимость вероятностных алгоритмов, можно воспользоваться рекурсивностью дерева вычислений статистического клеточного автомата.

**ЛЕММА** 1 (о конечности). *Предположим, что* $(X, M, \{K_m\}_{m=1,\ldots,M}, \xi_0)$ *некоторый статистический клеточный автомат, и* $t \in \mathbf{Q}$ *— строго положительное рациональное число. Тогда существует лишь конечное число различных конфигураций* $\xi_t(\omega)$ *(несмотря на то, что множество существенно различных* $\omega \in \Omega$ *бесконечно).*

**ДОКАЗАТЕЛЬСТВО.** Мы уже отмечали выше, что $L(s) \leq L(0) + 3s$ для любой конфигурации $\xi^{(s)}$, $s \in \mathbf{N}$. Следовательно, $\delta t(s) \geq \dfrac{1}{L(0)+3s}$, и значение $(s+1)$-го критического момента времени $t_{s+1} = \delta t_0 + \ldots + \delta t_s$ превышает величину $\dfrac{1}{3}\ln\left(1 + \dfrac{3s}{L(0)+1}\right)$ независимо от параметра $\omega \in \Omega$. Поэтому, существует такое $s$, что $(\forall \omega \in \Omega) t_{(s+1)}(\omega) \geq t$.

Для любого SCA, дерево вычислений содержит не более $\leq 2^{s+1}$ узлов на уровнях $\leq s$. Поскольку каждая реализация $\xi_t(\omega)$ соответствует некоторой ветви дерева вычислений, существует $\leq 2^{s+1}$ различных таких реализаций. ■

Каждой вершине $q$ дерева вычислений статистического клеточного автомата приписана некоторая вероятность $P(q)$, как описывается в доказательстве теоремы 1. Обозначим через $b(\omega)$ ту ветвь дерева вычислений SKA, которая соответствует указанному $\omega \in \Omega$. По построению, $P(q) = \mathbf{P}\{\omega \in \Omega | b_s(\omega) = q\}$, где $s$ — уровень вершины $q$ в дереве.

Фрагмент $T^s$ всех вершин уровня $\leq s$ дерева вычислений $T$ конечен. Значит, существует гёделева нумерация всех таких фрагментов, и можно исследовать вычислимость некоторых естественных функций на этих фрагментах.

Для каждого статистического клеточного автомата $(X, M, \{K_m\}_{m=1,\ldots,M}, \xi_0)$ функция $S \mapsto T^s$ рекурсивно вычислима так же, как и отображение $q \mapsto P(q)$. Заметим, что несмотря на бесконечность множества реализаций $b(\omega)$, проходящих через определенную вершину $q \in T$, критический момент времени

$t(q) = t_s(\omega)$, где $s = s(q)$ — уровень $q$ в $T$, определяется однозначно, и функция $q \mapsto t(q)$ рекурсивно вычислима.

Рассмотрим некоторое положительное число $t \in \mathbf{Q}$, тогда существует лишь конечное число таких $q \in T$, что $t(q) \le t$. Пусть $\widehat{t}$ — наибольшее из таких $t(q)$. Мы можем вычислить гёделев номер набора $\langle q_1, \ldots, q_V \rangle$ всех таких вершин, что $t(q_\nu) = \widehat{t}$, а также рекурсивно поставить в соответствие гёделев номер конфигурации $\xi_t(\nu)$ каждому варианту $q_\nu$, $\nu = 1, \ldots, V$, и вычислить ее вероятность $P(q_\nu)$.

Рассмотрим гёделеву нумерацию всех финитных функций $f$: $\mathbf{N} \to \mathbf{Q}$, т.е. таких функций, что $f(n) = 0$ для почти всех $n \in \mathbf{N}$ (для всех $n$ кроме лишь конечного их множества). Функция распределения, вычисляющая вероятность $P\{\omega \in \Omega |$ *код* $\xi_t(\omega)$ *равен* $n\}$ по числу $n$ как раз является такой финитной функцией, и ее гёделев номер рекурсивно вычисляется по заданному моменту времени $t \in \mathbf{Q}$.

Эти рассуждения фактически доказывают следующую теорему.

ТЕОРЕМА 2 (о вычислимости). *Для каждого статистического клеточного автомата и (рационального) момента времени $t \ge 0$, все возможные реализации текущего состояния решетки $\xi_t$ составляют конечное множество. Функция распределения, заданная на гёделевых номерах этих состояний финитна и, ее гёделев номер рекурсивно вычисляется по гёделеву коду клеточного автомата и по числу $t$.*

## Алгоритмическая универсальность SCAM

Ни один статистический клеточный автомат $A = (X, M, \{K_m\}_{m=1,\ldots,M}, \xi_0)$ никогда не останавливается, т.е. его дерево вычислений $T_A$ не имеет концевых вершин (листьев). Нам, однако, понадобиться некоторая аналогия команды остановки автомата, чтобы имитировать остановку машины Тьюринга.

Введем с этой целью соглашение считать, что реализация $\xi(\omega)$ "останавливается" в определенный момент времени $t = t_s$, если в этот момент центральная ячейка решетки $\xi_t(0,0)$ переключается в особое состояние $x_0 \in \{0, \ldots, X-1\}$. Оговорим также,

что определяемый ниже алгоритмически универсальный СКА будет сохранять состояние $\xi_t(0,0) = x_0$ для всех последующих моментов времени $t \geq t_s$. Мы будем применять мнемоническое обозначение $x_0 =$ "$\underline{Halt}$" для этого особого состояния.

Вслед за авторами работы [4], рассмотрим универсальную программируемую машину **M**, т.е. такую программируемую машину, которая способна вычислять все частично-рекурсивные функции, подобно универсальной машине Тьюринга. Пусть $\langle C_1, \ldots, C_N \rangle$ — последовательность инструкций (программа) машины **M** (см. определение 1). Мы построим статистический клеточный автомат $\mathbf{A} = (X, M, \{K_m\}_{m=1,\ldots,M}, \xi_0)$, имитирующий работу машины **M**. Число возможных состояний для ячеек автомата мы зарезервируем равным $X = N + 16$ и введем для этих состояний мнемонические обозначения: $0, \underline{Halt}, \underline{C_1}, \ldots$ $\ldots, \underline{C_N}, \underline{a}, \underline{a^+}, \underline{a^-}, \underline{A}, \underline{A^+}, \underline{A^-}, \underline{A^*}, \underline{b}, \underline{b^+}, \underline{b^-}, \underline{B}, \underline{B^+}, \underline{B^-}, \underline{B^*}$.

Аналогично [4], мы определим 1 мерный клеточный автомат, функция состояния которого $\xi_t(i)$ зависит только от одной координаты $i \in \mathbf{Z}$ вместо $(i,j) \in \mathbf{Z} \times \mathbf{Z}$. Разумеется, установленный теоремой 2 результат о вычислимости СКА справедлив и для 1-мерного случая.

Центральная ячейка решетки $\xi(0)$ будет имитировать счетчик инструкций программируемой машины **M**, и мы полагаем ее начальное состояние равным $\xi_0(0) = \underline{C_1}$. Далее полагаем начальное состояние $\xi_0(-1) = \ldots = \xi_0(-A_0) = \underline{A}$, и $\xi_0(1) = \ldots = \xi_0(B_0) = \underline{B}$, где $A_0$ и $B_0$ — начальные значения регистров $A$ и $B$ программируемой машины. Наконец, мы устанавливаем специальные маркеры в $\xi_0(-A-1) = \underline{a}$ и $\xi_0(B+1) = \underline{b}$ и полагаем остальные ячейки равными $\xi_0(i) = 0$.

Для автомата **A** определяется единственное правило трансформаций $K$: $\mathbf{X}^2 \to \mathbf{X}^2$. Приведенная ниже таблица описывает все подстановки, определяемые преобразованием $K$ с точностью до симметрии: если некоторая замена $(x,y) \to (x',y')$ приведена в таблице, то подразумевается, что симметричная подстановка $(y,x) \to (y',x')$ также определена для отображения $K$. С другой стороны, остальные подстановки считаются недопустимыми.

Таблица

**Полный список всех нетривиальных трансформаций вида $(x, y) \to (x', y')$, определенных для правила $K : \mathbf{X}^2 \to \mathbf{X}^2$, управляющего универсальным клеточным автоматом $\mathbf{A}$**

| Если $\underline{C_n} = A^+$ : | Если $\underline{C_n} = B^+$ : | Если $\underline{C_n} = Go[m]$ : |
|---|---|---|
| $(\underline{C_n}, \underline{a}) \to (\underline{C_n}, \underline{a^+})$ | $(\underline{C_n}, \underline{b}) \to (\underline{C_n}, \underline{b^+})$ | $(\underline{C_n}, \underline{a}) \to (\underline{C_m}, \underline{a})$ |
| $(\underline{C_n}, \underline{A}) \to (\underline{C_n}, \underline{A^+})$ | $(\underline{C_n}, \underline{B}) \to (\underline{C_n}, \underline{B^+})$ | $(\underline{C_n}, \underline{b}) \to (\underline{C_m}, \underline{b})$ |
| $(\underline{C_n}, \underline{A^*}) \to (\underline{C_{n+1}}, \underline{A})$ | $(\underline{C_n}, \underline{B^*}) \to (\underline{C_{n+1}}, \underline{B})$ | $(\underline{C_n}, \underline{A}) \to (\underline{C_m}, \underline{A})$ |
| Если $\underline{C_n} = A^-[m]$ : | Если $\underline{C_n} = B^-[m]$ : | $(\underline{C_n}, \underline{B}) \to (\underline{C_m}, \underline{B})$ |
| $(\underline{C_n}, \underline{a}) \to (\underline{C_m}, \underline{a})$ | $(\underline{C_n}, \underline{b}) \to (\underline{C_n}, \underline{b})$ | Соглашение I: |
| $(\underline{C_n}, \underline{A}) \to (\underline{C_n}, \underline{A^-})$ | $(\underline{C_n}, \underline{B}) \to (\underline{C_n}, \underline{B^-})$ | Если $n = N$, то |
| $(\underline{C_n}, \underline{a^-}) \to (\underline{C_{n+1}}, \underline{a})$ | $(\underline{C_n}, \underline{b^-}) \to (\underline{C_{n+1}}, \underline{b})$ | полагаем |
| $(\underline{C_n}, \underline{A^*}) \to (\underline{C_{n+1}}, \underline{A})$ | $(\underline{C_n}, \underline{B^*}) \to (\underline{C_{n+1}}, \underline{B})$ | $C_{n+1} = \underline{Halt}$ |
| Инкремент $A$ : | Инкремент $B$ : | Соглашение II: |
| $(\underline{A^+}, \underline{A}) \to (\underline{A^+}, \underline{A^+})$ | $(\underline{B^+}, \underline{B}) \to (\underline{B^+}, \underline{B^+})$ | Если подстановка |
| $(\underline{A^+}, \underline{a}) \to (\underline{A^+}, \underline{a^+})$ | $(\underline{B^+}, \underline{b}) \to (\underline{B^+}, \underline{b^+})$ | $(x, y) \to (x', y')$ определена здесь, то |
| $(\underline{a^+}, 0) \to (\underline{A^*}, \underline{a})$ | $(\underline{b^+}, 0) \to (\underline{B^*}, \underline{b})$ | симметричная замена $(y, x) \to (y', x')$ |
| $(\underline{A^+}, \underline{A^*}) \to (\underline{A^*}, \underline{A})$ | $(\underline{B^+}, \underline{B^*}) \to (\underline{B^*}, \underline{B})$ | также определена |
| Декремент $A$ : | Декремент $B$ : | Соглашение III: |
| $(\underline{A^-}, \underline{A}) \to (\underline{A^-}, \underline{A^-})$ | $(\underline{B^-}, \underline{B}) \to (\underline{B^-}, \underline{B^-})$ | Если какой-то замены или симметричной к ней здесь нет, она запрещена |
| $(\underline{A^-}, \underline{a}) \to (\underline{a^-}, 0)$ | $(\underline{B^-}, \underline{b}) \to (\underline{b^-}, 0)$ | |
| $(\underline{A^-}, \underline{a^-}) \to (\underline{A^*}, \underline{a})$ | $(\underline{B^-}, \underline{b^-}) \to (\underline{B^*}, \underline{b})$ | |
| $(\underline{A^-}, \underline{A^*}) \to (\underline{A^*}, \underline{A})$ | $(\underline{B^-}, \underline{B^*}) \to (\underline{B^*}, \underline{B})$ | |

Напомним, что множеством $\{A^+,\ B^+,\ A^-[n],\ B^-[n],\ Go[n],\ Halt\}$, $n = 1,\ldots,N$, исчерпывается все множество возможных команд для машины **M**. При этом подразумевается, что условный или безусловный оператор "*go to*", исполнямый командами $A^-[n], B^-[m]$, или $Go[m]$ при $m < 1$ или $m > N$, эквивалентен команде "$Halt$". Аналогично, программируемая машина останавливается, если ее последняя инструкция в программе $C_N$ не приводит к переходу к какой-либо команде $C_n$, $1 \le n \le N$, а завершается обычным образом, и следующей $N+1$ инструкции не существует. Иными словами, мы подразумеваем, что $C_{N+1} = \underline{Halt}$, чтобы упростить описание таблицы.

ЛЕММА 2. *Определенный выше статистический клеточный автомат* **A** *имитирует работу универсальной программируемой машины* **M**. *Иными словами:*

*если машина* **M** *останавливается при заданных начальных значениях регистров* $A$ *и* $B$, *то работа автомата* **A** *также завершается состоянием* $\xi_t(0) =$ "$\underline{Halt}$" *для почти всех реализаций, и при этом число символов* $\underline{A}$ *записанных в ячейки слева от центральной в точности равно значению регистра* $A$ *машины* **M** *в момент остановки, и то же самое верно для регистра* $B$. *Критический момент* $t$ "*остановки*" *статистического клеточного автомата случаен, т.е. зависит от реализации;*

*если же машина* **M** *не останавливается, тогда все ветви дерева вычислений для автомата* **A** *бесконечны, и ни одна его реализация не завершается командой* "$Halt$".

*Далее рассмотрим процесс вычислений* $r_0 \to r_1 \to \ldots$ *машины* **M**. *Здесь* $r_0$ *— ее начальное состояние (т.е. тройка* $\langle n = 1, A_0, B_0\rangle$), $r_1$ *— состояние машины после того, как была выполнена первая инструкция, и т.д. Тогда, для почти всякой реализации* $\xi[0] \to \xi[1] \to \ldots$ *автомата* **A** *существует такая под-последовательность* $s_0 < s_1 < \ldots$ *микро-шагов алгоритма Монте-Карло, что состояние* $\xi[s_k]$ *в точности моделирует состояние* $r_k$.

*Более того, вероятность того, что* $\xi_t(\omega)$ *рано или поздно достигнет состояния, моделирующего шаг* $r_k$ *(не застрянет по дороге), равна единице для всякого* $k$ *вплоть до остановки машины* **M**.

ДОКАЗАТЕЛЬСТВО. Выберем некоторое $k \geq 0$ и предположим, что состояние $r_k = \langle n, A_k, B_k \rangle$, где $n$ — номер текущей инструкции машины **M** и $A_k, B_k$ — состояния регистров. Покажем, как отыскать такое $s_k$, что $\xi[s_k](0) = "\underline{C_n}"$, а также $\xi[s_k](-1) = \ldots = \xi[s_k](-\underline{A_k}) = "\underline{A}"$, и $\xi[s_k](-\underline{A_k} - 1) = "\underline{a}"$, и $\xi[s_k](1) = \ldots = \xi[s_k](\underline{B_k}) = "\underline{B}"$, и $\xi[s_k](\underline{B_k} + 1) = "\underline{b}"$, и остальные ячейки решетки $\xi[s_k](i) = 0$.

Если $k = 0$, то $r_k = \langle 1, A_0, B_0 \rangle$. По определению, начальное состояние $s_0 = 0$ автомата **A** как раз моделирует начальное состояние $r_0$ машины **M**. Рассмотрим, каким образом автомат **A** моделирует переход $r_k \to r_{k+1}$.

Предположим, что реализация $\xi = \xi[s_k]$ моделирует состояние $r_k = \langle n, A_k, B_k \rangle$. (Мы будем писать $\xi[s_k](i)$ вместо $\xi^{(s_k)}(i)$, чтобы не использовать двойные индексы.)

Рассмотрим наиболее сложный случай, когда $C_n$ — команда вычитания, например "$A^-[m]$".

Объем $L$ нетривиальной части текущего состояния реализации $\xi$ равен $A_k + B_k + 5$. Следовательно, вероятность выбора некоторой нетривиальной пары ячеек $\langle i, i+1 \rangle$ для микро-шага $\xi \to \xi'$, равная $\dfrac{1}{L}$, отлична от нуля. Обозначим $x = \xi(i)$, и $y = \xi(j)$.

Если пара $\langle x, y \rangle$ не равна $\langle \underline{A}, \underline{C_n} \rangle$ или $\langle \underline{a}, \underline{C_n} \rangle$ (с точностью до симметрии), то правило трансформаций $K(x, y)$ не определено, и этот микро-шаг составляет $\xi' = \xi$.

Если $\langle x, y \rangle = \langle \underline{a}, \underline{C_n} \rangle$, то $A_k = 0$, и машина **M** должна перейти к инструкции номер $m$ (выполнить переход "*go to* [$m$]"). Соответственно состояние автомата **A** преобразуется так, что $\xi'(0) = "C_m"$. В данном случае, полагаем $s_{k+1} = s_k + 1$. Не исключено при этом, что автомату **A** понадобится сделать много нерезультативных микро-шагов $\xi = \xi' = \xi'' = \ldots$ прежде, чем будет выбрана нужная пара ячеек, и он придет в состояние $\xi[s_{k+1}](0) = "C_m"$. Тогда величина $s_{k+1} = s_k + w + 1$, где $w$ — количество потребовавшихся нерезультативных шагов.

Если $\langle x, y \rangle = \langle \underline{A}, \underline{C_n} \rangle$, то $i = -1$ и $\xi(i) = "\underline{A}"$. По правилу $K$, новое состояние принимается равным $\xi'(i) = "\underline{A^-}"$ и другие ячейки не затрагиваются. Поскольку $K(\underline{A^-}, \underline{C_n})$ не определено,

эта пара ячеек не будет изменена при последующих микро-шагах автомата.

Единственная пара ячеек, для которой возможна трансформация, — это пара $\langle i-1, i\rangle$, расположенная на одну ячейку левее. Возможны два варианта: $\xi'(i-1) = "\underline{A}"$, либо встретится концевой маркер $\xi'(i-1) = "\underline{a}"$. В первом случае, на следующих микро-шагах автомата будет продолжаться распространение пометки "$-$" над символами "$\underline{A}^-$", согласно правилу $K(\underline{A}, \underline{A}^-) = (\underline{A}^-, \underline{A}^-)$ и до тех пор, пока не встретится пара $\langle \underline{a}, \underline{A}^-\rangle$ с терминальным маркером "$\underline{a}$".

Когда на некотором шаге $s > s_k$ метка "$-$" достигнет последнего символа "$\underline{A}$", расположенного в позиции $\xi[s](-A_k)$, концевая комбинация $\langle \underline{a}, \underline{A}^-\rangle$ будет замещена на $\langle 0, \underline{a}^-\rangle$ на некотором шаге $s' > s$. В этот момент, количество заглавных букв "$\underline{A}$" в текущем состоянии решетки $\xi[s']$ станет равным $A_k - 1$.

Помеченный минусом маркер "$\underline{a}^-$" рядом с "$\underline{A}^-$" инициирует подстановку $\langle \underline{a}^-, \underline{A}^-\rangle \to \langle \underline{a}, \underline{A}^*\rangle$ и прикрепляет символ "$*$" к самой левой заглавной "$\underline{A}$". Такая пометка символизирует, что вычитание единицы успешно завершено, и счетчик команд должен быть заменен подстановкой $C_n \to C_{n+1}$. Это произойдет после того, как пометка "$*$" продвинется при помощи серии подстановок $\langle \underline{A}^*, \underline{A}^-\rangle \to \langle \underline{A}, \underline{A}^*\rangle$ до ячейки $\xi[s'](-1)$ на некотором шаге $s'' > s'$. По дороге, с символов "$\underline{A}$" снимается пометка "$-$".

Наконец, на одном из последующих шагов, пара $\langle \xi(-1) = "\underline{A}^*", \xi(0) = "C_n"\rangle$ обращаться в $\langle \underline{A}, \underline{C}_{n+1}\rangle$, или в $\langle \underline{A}, \underline{Halt}\rangle$, если $n = N$. Номер $s_{k+1}$ тогда равен номеру того микро-шага, когда это происходит.

Успешное завершение команды "$A^-[m]$" также происходит, если $A_k = 1$, и соответственно после замены $\langle \underline{a}, \underline{A}^-\rangle \to \langle 0, \underline{a}^-\rangle$ маркер "$\underline{a}^-$" попадает непосредственно в ячейку $i = -1$. В этом случае замена $C_n \to C_{n+1}$ или $C_n \to Halt$ происходит согласно правилу $\langle \underline{a}^-, \underline{C}_n\rangle \to \langle \underline{a}, \underline{C}_{n+1}\rangle$.

Таким образом, для каждого состояния $\xi[s_k]$, имитирующего состояние $r_k$, существует продолжение $\xi[s_{k+1}]$, имитирующее следующее состояние $r_{k+1}$. Более того, любое продолжение эволюции клеточного автомата из состояния $\xi[s_k]$ либо приводит в состояние, моделирующее $r_{k+1}$, либо приводит к бесконечной цепочке из

*промежуточных* состояний автомата **A**, не моделирующих никакого состояния машины **M**.

Следовательно, если машина **M** когда-либо останавливается, найдется такая реализация $\xi_t$, которая также приводит к состоянию "*Halt*". В противном случае, автомат **A** не обладает реализациями, приводящими к состоянию "*Halt*".

Предположим, что работа машины **M** останавливается, и рассмотрим вероятностные свойства имитирующего ее автомата **A**.

Пусть $q_0$ — корень дерева вычислений для автомата **A**. По определению, он моделирует начальное состояние $r_0$ машины **M**. Положим $Q_0 = \{q_0\}$ и определим $Q_{k+1}$ как $\bigcup\{Q(q_k)|q_k \in Q_k\}$, где $Q(q_k)$ — множество всех вершин дерева, которые получаются из $q_k$ таким же путем, как состояние $\xi[s_{k+1}]$ было построено из $\xi[s_k]$ в предыдущих абзацах. Это означает, что $q_{k+1} \in Q(q_k)$ если и только если существует такой фрагмент ветви $q_k \to \ldots \to q_{k+1}$ в дереве вычислений, что $q_{k+1}$ моделирует $r_{k+1}$, и каждый узел в этом фрагменте кроме $q_k$ и $q_{k+1}$ является "*промежуточным*".

Узел $q$ в этом фрагменте мы называем промежуточным, если соответствующее состояние $\xi(q)$ остается равным $\xi(q_k)$, т.е. в подфрагменте $q_k \to \ldots q$ все микро шаги не изменяют состояния решетки (нерезультативны), или же $\xi(q)$ содержит ячейки, маркированные надиндексами "+", "−" или "*." Промежуточные вершины не могут моделировать новое состояние машины **M**, но они представляют определенные стадии моделирования транзакции $r_k \to r_{k+1}$.

Поскольку алгоритм функционирования SCA — вероятностный, длина $w(q_k, q_{k+1})$ фрагмента $q_k \to \ldots \to q_{k+1}$ может быть различна для разных $q_{k+1} \in Q(q_k)$. Среди них существует однако такой фрагмент, что каждая трансформация $q \to q'$ в нем результативна. По построению автомата **A**, такой фрагмент — единственный; обозначим его длину как $w(q_k)$. Другие фрагменты могут содержать некоторую долю тривиальных ребер $q \to q$, когда пара ячеек для трансформации выбирается неудачно. Тем не менее, количество результативных шагов $q \to q'$ с $q \neq q'$ в любом таком фрагменте равно $w(q_k)$.

Наконец, существуют полностью нерезультативные ветви, начинающиеся с $q_k$, которые содержат бесконечную цепочку неудачных выборов пары ячеек (см. рис.2,3). На рис.2 узлы $q_{k+1}, \underline{q}_{k+1}$, и $q_{k+1}$ различны, но их состояния равны: $\xi(q_{k+1}) = \xi(\underline{q}_{k+1}) = \xi(q_{k+1})$. На рис.3 каждый узел $q_k$ из $Q_k$ соответствует шагу $r_k$ универсальной программируемой машины **M**. Существует бесконечно много таких тупиковых ветвей, начинающихся от различных узлов $q$ между множествами $Q_k$ и $Q_{k+1}$ (т.е. от тех $q$, которые попадают в один из фрагментов вида $q_k \to \ldots \to q_{k+1}$).

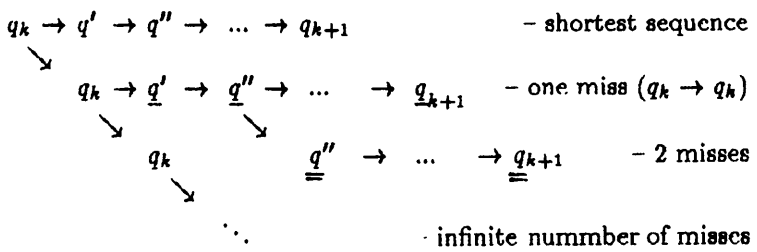


Рис.2. Некоторые ветви, начинающиеся от $q_k$

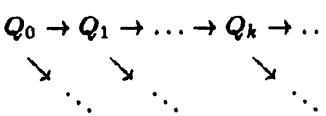


Рис.3. Сгруппированное дерево вычислений для автомата **A**

При доказательстве теоремы 1 каждому узлу $q$ дерева вычислений был приписан вероятностный вес. По построению, этот вес в точности равен вероятности $P(q)$, определяемой как $\mathbf{P}\{\omega|\xi_t(\omega)$ *проходит через* $q\}$.

Для произвольного подмножества $Q \subseteq T_A$ положим $P(Q) := \Sigma\{P(q)|q \in Q\}$.

Покажем, что для каждого $k \in \mathbf{N}$ суммарная вероятность $P(Q_k) = 1$.

Заметим, что в дереве вычислений автомата $\mathbf{A}$ присутствует лишь счетное число тупиковых ветвей, и вероятность узлов $P(q) \to 0$ стремится к нулю вдоль каждой такой ветви. Согласно предложению 2 (следующему ниже) суммарная вероятность $P_{\text{dead}}$, определяемая как $\mathbf{P}\{\omega|\xi_t(\omega)$ — *тупиковая ветвь*$\}$, равняется нулю.

Таким образом,вероятность $\mathbf{P}\{\omega|\xi_t(\omega)$ *моделирует работу* $\mathbf{M}\}$ равна 1.

Далее для каждого узла $q \in Q_k$ обрежем ветвь дерева $T$, исходящую из этого узла, и обозначим получившееся поддерево как $T[k]$. Подмножество $Q_k$ в точности совпадает с множеством концевых вершин (листьев) дерева $T[k]$, но при этом $T[k]$ содержит бесконечное множество тупиковых ответвлений, если $k > 0$. Согласно предложению 3 , суммарная вероятность концевых узлов $P(Q_k)$ равна 1.

Поэтому, вероятность $\mathbf{P}\{\omega|\xi_t(\omega)$ *достигает* $r_k$ *при некотором* $t\}$ равна 1. ■

Следующая основная теорема прямо следует из леммы 2.

ТЕОРЕМА 3 (об универсальности). *SCAM алгоритмически универсальна, т.е. существует универсальный статистический клеточный автомат.*

## Приложение: представление взвешенных деревьев

Рассмотрим некоторое дерево $T$ и его узел $q \in T$. Обозначим через $T[q]$ множество всех узлов, непосредственно исходящих из узла $q$. Дерево $T$ называют конечно ветвящимся, если множество $T[q]$ конечно для любого узла $q \in T$. Узел $q \in T$ называется концевым (или листом), если множество $T[q]$ пусто.

Последовательность $g = (q_0 \to q_1 \to \ldots) \in T$ узлов $q_0, q_1, \ldots$ назовем ветвью, если $q_0$ — корень дерева $T$, каждая стрелка соответствует паре узлов, связанных как $q_{k+1} \in T[q_k]$, и последовательность либо кончается некоторым листом $q_s$ дерева $T$, либо является бесконечной. Обозначим множество всех ветвей дерева $T$ через $G_T$. Для каждой $g \in G_T$ и $s \in \mathbf{N}$ обозначим $g_s := q_s$, если $g = (q_0 \to q_1 \to \ldots)$ и длина $g$ больше $s$, или будем

считать значение $g_s$ не определенным, если $g = (q_0 \to \ldots \to q_s)$ и $s > S$. Для каждого $q \in T$ и $g \in G_T$ пишем $q \in g$, если $\exists s \;\; q = g_s$.

Предположим, что $(\Omega, \mathbf{P})$ — некоторое вероятностное пространство. (Здесь неявно подразумеваемая $\sigma$-алгебра $\mathrm{dom}(\mathbf{P}) \subseteq 2^\Omega$ однозначно задана мерой $\mathbf{P}$.)

**ОПРЕДЕЛЕНИЕ** 4. Отображение $\omega \in \Omega \mapsto g(\omega) \in G_T$ называется *случайным процессом* на дереве $T$, если для каждого узла $q \in T$ вероятностный вес $P(q) := \mathbf{P}\{\omega \in \Omega | q \in g(\omega)\}$ определен (т.е. множество $\{\omega | q \in g(\omega)\}$ является $\mathbf{P}$-измеримым).

Очевидно, вероятностный вес, определенный некоторым вероятностным процессом, обладает следующими свойствами:

1) вес корневого узла дерева $P(q_0) = 1$;

2) для всех $q \in T$, если $T[q] \neq \emptyset$, то вес $P(q)$ равняется сумме $\Sigma\{P(q') | q' \in Tq\}$.

**ОПРЕДЕЛЕНИЕ** 5. Каждое отображение $p : T \to [0, 1]$, удовлетворяющее данным условиям 1 и 2, называется весовой функцией. Дерево $T$ вместе с любой фиксированной весовой функцией $p$ называется *взвешенным деревом*.

Следующая лемма утверждает, что любое взвешенное дерево можно представить при помощи вероятностного процесса над каноническим вероятностным пространством $\Omega = [0, 1]$ и $\mathbf{P} = dx$ (мера Лебега). Обозначим через $\mathbf{I}$ множество непустых замкнутых интервалов $[a, b] \subseteq [0, 1]$.

**ЛЕММА** 3 (о представлении). *Пусть $T$ — конечно ветвящееся дерево и $p : T \to [0, 1]$ — весовая функция. Тогда существуют такие представления $V : G_T \to \mathbf{I}$ и вероятностный процесс $g : [0, 1] \to G_T$, что:*

1) $(\forall x \in [0, 1]) \quad x \in V(g(x))$,

2) $(\forall q \in T) \;\; p(q) = P(q)$, *где* $P(q) = \mathbf{P}\{x \in [0, 1] | q \in g(x)\}$.

**ДОКАЗАТЕЛЬСТВО.** Определим произвольное линейное упорядочение $(T[q], \leq)$ для всех таких $q \in T$, что $T[q] \neq \emptyset$. Эти локальные упорядочения индуцируют лексикографический линейный порядок на множестве $(G_T, \leq)$, причем для любого подмножества $G \subseteq G_T$ имеются включения $\inf(G) \in G$ и $\sup(G) \in G$.

Индукцией по уровню $s(q)$ узла $q$ легко определить такое отображение $V : T \to \mathbf{I}$ что:

1) для любого $q \in T$ длина отрезка $V(q)$ равняется $p(q)$,

2) если $V(q) = [a, b]$, и $T[q] = \{q_1, \ldots, q_N\}$, $N \geq 1$, и $V(q_n) = [a_n, b_n]$, то $a_1 = a$, и $b_N = b$, и $(\forall n = 1, \ldots, N-1) a_{n+1} = b_n$.

Для любого $g \in G_T$ определим $V(g) = \cap\{V(q) | q \in g\}$. Поскольку $V(q)$ компактно для любого $q$, то $V(g) \neq \emptyset$.

Рассмотрим произвольное $x \in [0, 1]$ и положим $g(x) := \inf\{g \in G_T | x \in V(g)\}$. Заметим, что узел $g(x)$ определен корректно. Во-первых, $x \in V(q_0) = [0, 1]$, где $q_0$ — корень дерева. Предположим, что мы нашли такой фрагмент ветви $(q_0 \to \ldots \to q_s)$, что $x \in V(q_s)$. Если $q_s$ лист дерева, то $x \in V(g)$, где $g = (q_0 \to \ldots \to q_s)$. В противном случае, рассмотрим любой такой узел $q \in T[q_s]$, что $x \in V(g)$, и определим $q_{s+1} = q$. Этот индуктивный процесс либо завершится, достигнув некоторого листа $q_s \in T$, либо породит бесконечную ветвь $g = (q_0 \to q_1 \to \ldots)$ такую, что $x \in V(g)$.

В любом случае, множество $\{g | x \in V(g)\}$ не пусто, и $g(x)$ определено.

По определению, $x \in V(g(x))$ для всех $x \in [0, 1]$.

Остается доказать, что для всех $q \in T$ вероятность $P(q)$ корректно определена и равняется весу $p(q)$.

Рассмотрим некоторый узел $q \in T$ и предположим, что $[a, b] = V(q)$. По построению, $V(g) \subseteq [a, b]$ для всех таких $g \in G_T$, что $q \in g$. Покажем, что с другой стороны $(a, b) \subseteq \{x | q \in g(x)\}$, где $(a, b)$ — открытый интервал $a < x < b$.

Для произвольного $x \in (a, b)$ рассмотрим $g(x) = (q_0 \to q_1 \to \ldots)$ и предположим, что $\forall s\ q_s \neq q$. Пусть $S$ — наибольший такой номер $s$, что $q_s$ предшествует узлу $q$ в структуре дерева $T$. (По меньшей мере, корень $q_0$ является предшественником узла $q$.) Множество $T[q_S]$ не пусто, поскольку $q_S$ предшествует $q$ и $q_S \neq q$. Более того, найдутся два различных узла $q', q'' \in T[q_S]$ таких, что:

- $q'$ предшествует $q$ в некоторой ветви дерева, проходящей через $q$;
- $q'' = q_{S+1}$.

По построению, либо $V(q') \cap V(q'') = \emptyset$, либо это пересечение содержит только единственную точку — один из концов интервала $V(q')$. Поскольку отрезок $[a,b] = V(q) \subseteq V(q')$, только один из концов интервала $[a,b]$ может попасть в $V(q'')$.

Имеем: $x \in V(q'')$ и $x \in [a,b] \subseteq V(q')$. Значит $x = a$ или $x = b$. Но мы выбирасм $a < x < b$. Значит, предположение $q \notin g(x)$ не верно. Таким образом показано, что $(a,b) \subseteq \{x | q \in g(x)\} \subseteq [a,b]$. Следовательно, $P(q) = \mathrm{mes}([a,b]) = p(q)$. ■

Лемма 3 показывает, что на множестве $G_T$ ветвей произвольного взвешенного дерева $(T,p)$ можно определить каноническую меру $\mathbf{P}_p$.

Подмножество $G \subseteq G_T$ измеримо, если множество $\bigcup\{V(g) | g \in G\}$ измеримо относительно меры Лебега на интервале [0,1]. Положим $\mathbf{P}_p(G) = \mathrm{mes}\{x | g(x) \in G\}$. По определению, $\mathbf{P}_p(\{g | q \in g\}) = p(q)$ для всех $q \in T$.

Для любого вероятного процесса $\omega \in \Omega \mapsto g(\omega) \in G_T$ такого, что $\mathbf{P}\{\omega | q \in g(\omega)\} = p(q)$, индуцированая им мера $P(G) = P\{\omega | g(\omega) \in G\}$ в точности совпадает с канонической мерой $\mathbf{P}_p$. Назовем такие вероятностные процессы совместимыми с заданной весовой функцией.

Следующее предложение прямо вытекает из совпадения канонической меры и меры, индуцированной совместимым с весовой функцией вероятностным процессом.

**ПРЕДЛОЖЕНИЕ** 1. *Пусть $(T,p)$ — взвешенное дерево и $\mathbf{P}_p$ — каноническая мера на $G_T$. Назовем ветвь $g \in G_T$ несущественной, если $\mathbf{P}_p(\{g\}) = 0$. Пусть $G$ — некоторое счетное множество несущественных ветвей. Тогда для любого вероятностного процесса $\omega \in \Omega \mapsto g(\omega) \in G_T$, совместимого с весовой функцией $p$, индуцированная процессом мера $P(G)$ равна нулю.*

Дерево вычислений $T = T_A(x_1, \ldots, x_N)$ для вероятностного алгоритма $A$ — важный пример взвешенного дерева. Рассмотрим подмножество $G_0 \subseteq G_T$ таких ветвей, для которых результат вычисления $T = T_A(x_1, \ldots, x_N)$ не определен (алгоритм зацикливается, либо встречается операция с неопределенным результатом).Функция, вычисляемая $T_A(x_1, \ldots, x_N)$, может быть определена для почти всех реализаций тогда и только тогда, когда каноническая мера $\mathbf{P}_A(G_0)$ равна нулю.

Предположим, что ветвь $g \in G_0$ бесконечна, т.е. что алгоритм здесь зацикливается. Тогда существует некоторая характеристическая функция $\chi : G_T \to \mathbf{N}$, периодическая на каждой такой ветви $g \in G_0$, т.е. такая, что $(\exists S, S_0)(\forall s \geq S_0)\chi(g_{s+S}) = \chi(g_s)$ (например: указатель текущей команды). Отсюда следует, что мощность множества $G_0$ не более чем счетна. Если при этом все ветви множества $G_0$ — несущественные, то $P(G_0) = 0$.

**ПРЕДЛОЖЕНИЕ 2** (о зацикливании). *Пусть $G_0$ — некоторое подмножество ветвей взвешенного дерева. Предположим, что каноническая мера $\mathbf{P}\{g\} = 0$ для каждой $g \in G_0$, и что существует характеристическая функция $\chi : G_0 \to \mathbf{N}$ переодическая на каждой ветви $g \in G_0$. Тогда $\mathbf{P}(G_0) = 0$.*

Наконец, рассмотрим взвешенные деревья, имеющие не более, чем счетное множество бесконечных ветвей. Этот очевидный случай использовался нами для доказательства леммы 2 в предыдущем параграфе.

**ПРЕДЛОЖЕНИЕ 3** [о весах]. *Предположим, что каждая бесконечная ветвь взвешенного дерева $(T, p)$ является несущественной, и что множество бесконечных ветвей дерева не более, чем счетно. Тогда, суммарный вес $\Sigma\{p(q)|q -$ лист дерева $T\}$ равен единице.*

## З а к л ю ч е н и е

В данной работе мы придерживались "анти-прикладного" подхода, пытаясь получить чисто математические результаты как обобщение опыта прикладных исследований. Поэтому важно еще раз подчеркнуть, какие здесь вносятся упрощения по сравнению с "физической реальностью".

• Во-первых, мы представили в нашей модели время рациональными числами вместо вещественных. На самом деле, это — не существенно. Никакие приборы не измеряют физические величины с точностью до иррациональных чисел, и использование всей вещественной прямой не является обязательным, если не привлекается аппарат математического анализа.

• Более существенно, что мы пренебрегли скоростями элементарных физических процессов $K_1, \ldots, K_M$, формирующих химическую реакцию. Часто, поведение реакции критически чувстви-

тельно даже к малым вариациям констант $k_1, \ldots, k_M$, описывающих скорости этих процессов. Тем не менее, наше упрощение не отменяет принципиального факта, что поведение химических реакций может быть весьма сложным. В данной работе мы рассмотрели алгоритмическую меру этой сложности, установили ее верхнюю границу. Показано, что поведение любой химической реакции не может быть более сложным, чем алгоритмически универсальное, и что эта граница сложности является достижимой.

• Наиболее серьезной проблемой нашего подхода является следующее. Наша конструкция универсального SCAM нарушает принцип локального равновесия. В общем случае, если некоторая трансформация квантового состояния молекулы допустима, вероятность обратного превращения также должна быть отлична от нуля (хотя может быть и исчезающе мала). Наша таблица, описывающая универсальный автомат $A$, целиком состоит из необратимых правил. Это означает, что хотя алгоритмическая универсальность в химии теоретически возможна, для большинства простых реакций она оказывается недостижимой. Разумеется, этого и следует ожидать в простых случаях. С другой стороны, человеческий мозг доставляет пример химической системы, вероятно приближающейся к верхней границе алгоритмической сложности.

Настоящая работа не могла бы быть завершена (и даже начата) без опыта компьютерного моделирования конкретных химических реакций. Такой опыт был накоплен в результате сотрудничества с коллегами из Института катализа (ИК) СО РАН А.Л.Вишневским (внезапно скончался в 1995), В.И.Елохиным, В.В.Городецким, Б.С.Бальжинимаевым, А.В.Мышляевым (Тувинский комплексный отдел СО РАН), Е.Д.Реснянским, Л.Э.Шейнином.

Автор выражает также признательность за консультации и критические замечания В.И.Савченко (ИК), В.П.Жданову (ИК), Л.А.Москвитину (Институт математики СО РАН).

Литература

1. BARWISE J. (ed). Handbook of mathematical logic. (North Holland Publishing Company, Amsterdam, New York, Oxford, 1977.)

2. BINDER K. (ed). Monte Corlo methods in statistical physics. — Berlin: Springer–Verlag, 1979 .

3. Lattice methods and their applications to reacting sistems /Chen S., Dawson S.P., Doolen G.D. u.a.// Computers Chemical Engineering, 1995. - Vol. 19, № 6/7. - P. 617–646.

4. GOLES E., MAAS A., MARTINEZ S. On the limit set of some universal cellular automata //Theoretical Computer Science, 1993. - Vol. 110. — P. 53–78.

5. ELOKHIN V.I. VISHNEVSKII A.L., LATKIN E.I. Two approaches in modelling the critical phenomena (self–oscillations and spatio–temporal self–organisation) in heterogeneous catalytic reactions //Abstracts of 11$^{th}$ International Congress on Catalysis, Baltimore, 1996. - P. 321.

6. FICHTHORN K.A., WEINBERG W.H. Theoretical foundations of dynamical Monte Carlo simulations //J.Chem.Phys., 1991. Vol 95, № 2. — P. 1090–1096.

7. HOCKHEY R.W., EASTWOOD J.W. Computer simulation using particles. (McGraw–Hill. — 1979.)

8. KAPRAL R. Discrete models for chemically reacting systems //Journal of Mathematical Chemistry. 1991. — Vol. 6. — P. 113-163.

9. LATKIN E.I., ELOKHIN V.I., GORODETSKII V.V. Monte Carlo model of self–oscillations in CO + $O_2$/Pt(100) reaction caused by nonlinear law of surface reconstruction //Abstracts of Memorial G.K.Boreskov conference "Catalysis on the eve of the XXI century. Science and engineering.", July 7-11, 1997, Novosibirsk, Russia, — Part II. — P. 24–25.

10. Lattice statistical model for phase transition of catalyst for $CO_2$ oxidation/ Latkin E.I., Sheinin D.E., Elokhin V.I. u.a.// React. Kinet. Catal. Lett., 1995. — Vol. 56, № 1. — P. 169–178.

11. Computability by probabilistic machines/ De Leeuw K., Moore E., Shannon C.E. u.a. // Automata studies. - Princeton, N.J.: Princeton University Press (Annals of mathematics studies; № 34).

12. MINSKY M. Computation: finite and infinite machines. (Prentice–Hall, Englewood Cliffs, NJ, 1967).

13. Von NEUMAN J. Theory of self-reproducing automata. (University of Illinois Press, Urbana & London, 1966.)

14. Monomolecular adsorption on rough surfaces with dynamically changing morphology/ Resnyanskii E.D., Latkin E.I., Myshlyavtsev A.V. u.a.// Chem.Phys. Lett., 1996. - Vol. 248. — P. 136-140.

15. ROGERS H. Theory of recursive functions and effective computability. (McGraw-Hill, New York, 1967).

16. VISHNEVSKII A.L., LATKIN E.I., ELOKHIN V.I. Autowaves on catalyst surface caused by carbon monoxide oxidation kinetics: imitation model //Surface Review and Letters, 1995. — Vol. 2, № 4. — P. 459-469.

17. WOLFRAM S. Undecidability and intractability in theoretical physics //Phys. Rev. Lett., 1984. — Vol. 54, № 8. — P. 1-4.

18. ЕЛОХИН В.И., ЛАТКИН Е.И. Статистическая модель колебательных и волновых явлений на поверхности катализатора и реакции окисления CO //Докл. РАН, 1995. — Том 344, № 1. - С. 56-61.

19. ЕРШОВ Ю.Л. Теория нумераций. — М.: Наука, 1977. — 416 с.

20. ЕРШОВ Ю.Л. Проблемы разрешимости и конструктивные модели. — М.: Наука, 1980. - 416 с.

21. ЛАТКИН Е.И. SCAM: гибкая модель гетерогенного катализатора //Тезисы 9-й Международной конференции "Математические методы в химии" (ММХ-9), Тверь, май 1995. — Часть II. — С. 118-119.

22. ЛАТКИН Е.И. SCAM: химический компьютер //Теория вычислений и языки спецификаций. Новосибирск, 1995. — Вып. 152: Вычислительные системы. — С. 140-151.

23. ЛОСКУТОВ А.Ю., МИХАЙЛОВ А.С. Введение в синергетику. — М.: Наука, 1990. — 272 с.

24. МИХАЙЛОВ Г.А. Некоторые вопросы теории методов Монте Карло. Новосибирск: Наука, 1974. — 142 с.

25. ТОФФОЛИ Т., МАРГОЛУС Н. Машины клеточных автоматов. — М.: Мир, 1991. — 280 с.

26. УСПЕНСКИЙ В.А., СЕМЁНОВ А.Л. Теория алгоритмов: основные открытия и приложения. — М.: Наука, 1987. — 288 с.

Поступила в редакию
1 сентября 1999 года