УДК 004.946;519.673

И. В. Матюшкин, канд. физ.-мат. наук, нач. лаб.,
ОАО "НИИМЭ и завод Микрон"
e-mail:imatyushkin@sitronics.com

Перспективы развития современных средств проектирования клеточных автоматов

Проанализированы существующие машины клеточных автоматов (МКА), отмечены их недостатки и сформулированы требования, предъявляемые к ним современным уровнем развития нанотехнологии. Указаны направления модернизации МКА. Предложен новый МКА SoftCAM, архитектура которого зафиксирована UML-диаграммами.

Ключевые слова: *клеточные автоматы, САПР, UML*

Тематика клеточных автоматов оказалась широко востребованной в последнее десятилетие, число публикаций по ней неуклонно растет [1]. Клеточный автомат (КА) является фундаментальной абстракцией для представления параллельных вычислений подобно тому, как машина Тьюринга и конечный автомат репрезентируют последовательные вычисления [2]. Модель КА также может использоваться при эскизном проектировании мультипроцессорных систем или при исследовании сравнительно простых мультиагентных интеллектуальных систем. Прикладное значение КА связано с их использованием в качестве метода математического моделирования [3]. Например, на языке КА моделируются процессы диффузии, распространения лесного пожара, эпидемии и в целом рассчитываются пространственно-распределенные системы, включая квантовые [4]. В микроэлектронике известны применения КА для моделирования процесса травления при получении пористого кремния [5]. Не лишено значения применение КА в изобразительном искусстве, поскольку визуализации КА не уступают по красоте фрактальной графике [6].

Термин "машина клеточных автоматов" (МКА) введен Тоффоли. МКА является системой автоматизации проектирования клеточных автоматов; задавая правила функционирования клеточного автомата, можно реализовывать ту или иную математическую модель (или мультиагентную систему). Целью данной статьи является анализ существующих МКА, формулировка требований, предъявляемых к МКА современными нанотехнологиями, и выявление возможных путей развития МКА. Также впервые посредством UML-диаграмм предложена оригинальная архитектура МКА.

Обзор существующих МКА

Рассмотрим вначале "легкие" и свободно распространяемые МКА. Информация по ним суммирована в таблице. Долгое время, начиная с 70-х гг., когда игра "Жизнь" получила известность в университетской среде США и Европы, МКА создавались безызвестными энтузиастами. До сих пор создаются простые симуляторы игры "Жизнь" на основе флэш-анимации, Java-апплетов. Однако если раньше такие МКА создавались исследователями-любителями, ищущими новые конфигурации, то теперь здесь пробуют силы новички в программирова-

Характеристики основных свободно распространяемых МКА

Характеристики	Наименование программы			
	Fam life	MCell (MJCell)	Life32	Golly
Автор	Мозжухин Андрей, Фетисов Александр	Mirek Wijtowicz	Johan Bontes	Tomas Rokicki, Andrew Trevorrow, Dave Greene, Jason Summers, Tim Hutton
Год издания	1998/2002	1999/2001 (2005)	1999/2002	2005/2009
Сайт разработчика/Поддержка	http://www.fam-life.narod.ru/Нет	http://www.mirekw.com/ca/Нет	http://psoup.math.wisc.edu/Life32.html/Нет	http://golly.sourceforge.net/Да
Объем на жестком диске, Мбайт	1,2	1,75 (ядро)	1,73	9
Объем в оперативной памяти при запуске программы, Мбайт	1	13	9	12
Возможность изменять правила	Да	Да	Да	Да
Встроенные конфигурации	Да	Да	Нет	Да
Язык написания	Не документировано	Borland Delphi 5.0 (+Java)	Delphi 3.02	Delphi
Возможность замыкания границ пространства	Не документировано	Не документировано	Не документировано	Не документировано
Быстродействие, с (время расчета 10 000 ходов)	18	8	3	16 (в режиме Hyperspeed — 1)
Максимальный размер поля	1680 × 1025	100 000 × 2500	1 × 1 млн	Не ограничен

нии. Примером служит МКА Life 3D, где обычная 2D-игра визуализирована в 3D-пространстве с помощью API OpenGL. Также разработчики пакета MatLab включили в целях обучения в состав демонстрационных М-файлов небольшой симулятор игры "Жизнь".

Важным шагом в развитии МКА было появление симулятора Life 1.05 (автор: Alan Hensel, платформа: DOS, размер дистрибутива: 200 Кбайт, которое де-факто ввело стандарт LIF для записи конфигураций и упрощенную запись правил перехода. Наряду с российской разработкой FAM-life стоит отметить Life Editor 3 (автор: Владимир Крылов при сотрудничестве с фирмой "Геймос", дата: 1991—1994 гг.), продемонстрировавший арт-возможности клеточных автоматов.

Подробнее остановимся на последней по времени МКА Golly. Помимо открытого кода и кроссплатформенности она обладает следующими достоинствами:

- размер поля ограничен только физической памятью;
- число состояний ячейки до 256;
- использует быстрый и эффективный по занимаемой оперативной памяти алгоритм расчета QuickLife;
- возможность замены алгоритма расчета, в частности, алгоритм HashLife (Билл Госпер, 1984), основанный на хранении и хэшировании уже вычисленных фрагментов конфигураций и эффективный при симуляции на длительных временах больших по размеру конфигураций;
- наличие библиотек, содержащих многие классические варианты КА: ID-игра Стивена Вольфрама, "Мир-проволока", "Поколения", автомат фон Неймана с 29 состояниями, самовоспроизводящийся автомат Фредкина;
- интерфейс для задания собственных правил перехода;
- экспорт/импорт конфигураций и паттернов для LIF-, RLE-, MCL-файлов, а также форматов macrocell и dblife;
- поддержка стандартных графических bmp, tiff, gif, png-форматов;
- большая коллекция "удачных" паттернов;
- гибкое управление симуляцией через загрузку скриптов языков Perl и Python;
- специально разработанная система справки Life Lexicon, основанная на HTML.

Для Golly (рис. 1, см. четвертую сторону обложки) отметим великолепно реализованные элементы интерфейса для задания исходных конфигураций с возможностью масштабирования поля и импорта библиотечных паттернов, а также скриптовую поддержку.

Теперь кратко рассмотрим более "тяжелые" и соответственно закрытые и частично коммерческие МКА. Н. Марголус и Т. Тоффоли [2], начиная с середины 80-х гг. XX века, проводили в Массачусетском технологическом институте работы по симуляции клеточных автоматов для нужд биологии, кристаллографии и других прикладных дисциплин. Последней моделью указанной серии является CAM-8, технически представляющая собой систолический массив процессоров, симулирующий параллельную архитектуру SIMD-типа. Собственно говоря, авторы термина "машина клеточных автоматов" изначально придавали ему более узкое значение, чем используется в данной статье. Они неявно полагали дополнительно, что физическая реализация вычислителя предполагает параллелизм при симуляции КА. И действительно, клеточный автомат может сам моделировать многопроцессорную систему, но, вместе с тем, его симуляция действительно эффективна при проведении распределенных вычислений, а не последовательных, выполняемых на компьютере с архитектурой фон Неймана. В настоящее время создан ряд компьютеров, специализированных для таких вычислений. Одними из пионерских и серьезных практических разработок в данном направлении можно отметить клеточные процессоры Легенди [7] и ML-сопроцессоры. Итальянскими специалистами [8] в 1995 г. создана вычислительная среда CAMEL, использующая КА-модель в качестве теоретической основы и успешно применяемая для моделирования.

Безусловно, стоит отметить разработку питерских ученых Л. Наумова и А. Шалыто CAME&L (2007), которая не только обладает широкими функциональными возможностями МКА (изменение правил перехода достигается подключением внешних C++ библиотек), но и обеспечивает автоматизацию проектирования программного обеспечения систем с программируемой логикой (ПЛИС).

Для рядовых исследователей, потенциальных потребителей МКА, доступ к высокопроизводительным вычислительным системам затруднителен, поэтому становится актуальной задача создания МКА "средней тяжести", которые, с одной стороны, были бы реализуемы на персональном компьютере (с возможностью, если необходимо, сетевых вычислений), но, с другой стороны, обладали бы большей функциональностью, чем ориентированные скорее на занимательность "легкие" МКА.

Нельзя не сказать кратко об упомянутых форматах файлов. Здесь используется старая концепция знакоместа, а сами файлы имеют по сути текстовый формат. В RLE-файлах мертвая клетка кодируется символом "b", а живая — символом "o". Для перехода на следующую строку используется символ $, а для окончания записи паттерна символ !. Если более двух одинаковых клеток идут подряд, то они не выписываются все, а перед со-

ответствующим символом ставится число повторений. Обязательной является также вводная строка (заголовок), в которой указываются размеры ограничивающего прямоугольника и, возможно, правила перехода. Даже программе Golly правила задаются примитивно; для автомата "Поколения" — это три числа S/B/C, где S — набор цифр от 0 до 8, определяет число "живых" соседей, при котором клетка остается "в живых"; B — набор цифр от 0 до 8, определяет число "живых" соседей, при котором "мертвая" клетка становится "живой"; C — число, определяет число ходов "умирания" клетки.

Требования, предъявляемые к МКА

Цель разработчика программного обеспечения всегда состояла в предсказании и упреждении потенциальных пожеланий заказчика (пользователя). Можно выделить четыре группы пользователей:

1. Обычные пользователи, увлеченные красотой получаемых картинок и качеством анимации.

2. Специалисты по математическому моделированию, которым важно удобство задания КА, симулирующего физические (или даже социальные) процессы.

3. Разработчики распределенных систем, нуждающиеся в наибольшей свободе для выбора нестандартных правил, прежде всего относящихся к топологии КА и усложнению структуры ячейки, включая введение вероятностных мотивов и черт гетерогенности; тогда КА превращается в мультиагентную систему, что характерно, например, для мультипроцессорных систем.

4. Специалисты-математики, пытающиеся экспериментальным путем сформулировать или косвенно подтвердить гипотезы в области теории КА.

Из-за ограниченности вычислительных возможностей компьютера требования разных групп пользователей могут противоречить друг другу. Легко сообразить, что усложнение структуры ячейки (3-я группа), означающее увеличение памяти, приходящейся на ячейку, от 1 бит, достаточного для симуляции игры "Жизнь", до 64 байт, при 3D-топологии поля размера 256 × 256 × 256 повлечет за собой задействование всей оперативной памяти типичного компьютера (1 Гбайт), а при попытке использования дискового кэша катастрофически упадет быстродействие. При поиске компромисса следует попытаться удовлетворить принципу независимости архитектуры программы от деталей реализации.

Для всех групп пользователей, особенно 2-й и 3-й, необходимо предусмотреть расширение МКА на тот случай, когда потребуется вводить информацию непосредственно в КА во время симуляции. Классический подход к КА как к закрытой системе, реализованный в работе [2] и изначально полагаемый в теории КА, не приводит к успеху при моделировании существенно открытых систем или систем, нацеленных на обработку сигналов (в частности, когда КА рассматривается как объект управления).

При создании информационных систем обычно требуется вначале составить логическую модель предметной области или, по крайней мере, сформировать ее концептуальный базис. В нашем случае это по большей части, хотя и неполностью, сделано самими математиками при формулировке понятия КА. Основными концептами здесь выступают: ячейка, шаблон соседства, локальная функция переходов, конфигурация. При симуляции частично гетерогенных КА можно говорить об индексах соседства или функции соседства, а также о типах ячеек. Кроме того, при анализе опыта многочисленных программных реализаций игры Конуэя оказываются полезными следующие свойства МКА:

- широкое использование библиотек начальных фигур;
- интерактивное задание начальной конфигурации;
- управление скоростью симуляции;
- интерфейсные шаблоны для визуализации поля КА.

Основное требование к МКА состоит в том, чтобы предоставлять пользователю возможность средствами среды разработчика (IDE) конструировать КА из "кирпичиков" (они комплиментарны упомянутым выше понятиям), выбирать вариант его графического/файлового представления, вычислять общие характеристики его симуляции и, возможно, управлять процессом симуляции.

Конкретизируя это требование для разработчика, можно выделить такие желаемые свойства МКА:

- явное задание структуры ячейки (и ее типа);
- свободное задание в рамках одной МКА 1D-, 2D-, 3D-топологии и сложных топологий (например, на сфере или на поверхности геометрических тел), а также граничных условий для поля КА, в частности, его тороидальность;
- явное задание типа шаблона окрестности либо для каждой ячейки, либо для групп ячеек;
- возможность задания нешаблонной окрестности для индивидуальных ячеек;
- возможность динамического, т. е. во время симуляции, переопределения типа ячейки и ее шаблона окрестности;
- предоставление библиотек окрестностей и библиотек локальных функций перехода;
- возможность внедрения внешнего кода для локальных функций перехода;
- графический интерфейс для задания начальной конфигурации либо из ранее сохраненного файла, либо путем конструирования из библиотечных фигур (с возможностью их мультипликации);

- использование различных шаблонов оформления для КА, что особенно критично для 3D-структур;
- интерфейс для управления процессом симуляции, например, останова по требованию или автоостанова при "гибели" всех ячеек;
- возможность присоединения к глобальной функции перехода заданной пользователем функции, принудительно изменяющей состояния определенных ячеек (чаще всего граничных);
- сбор статистики симуляции, ее сохранение и визуализация (например, в игре Конуэя может рассчитываться число встретившихся периодических конфигураций или для каждой ячейки вычисляться ее среднее состояние).

Пути развития МКА

В настоящее время интерфейсы МКА уже хорошо развиты, предоставляя возможности экспорта/импорта конфигураций, управления цветовой гаммой и масштабированием при визуализации, а также изменения скорости симуляции. Многие программы обладают обширными библиотеками с присоединенной к ним справочной системой. Достигнут определенный прогресс в алгоритмах симуляции и стандартизован формат файлов конфигурации.

С нашей точки зрения, главной виной разработчиков "легких" МКА является загипнотизированность игрою "Жизнь", отсутствием инновационности в реализации базовых понятий клеточных автоматов с момента появления MKA Life 1.05. Из этого и следуют представляющиеся перспективными пути развития МКА, связанные с:

- усложнением структуры ячейки (от бита через байт к трехчленной монаде "вход—память—выход");
- усложнением топологии поля, заключающимся не только в переходе к 3D, но и в усложнении понятия "шаблона окрестности" (известно, например, что наноматериалы часто имеют пористую структуру со сложными связями, и моделирование таких структур инициирует прогресс МКА в этом направлении);
- введением черт гетерогенности/индивидуализации в классическое определение КА, т. е. в одном поле могут располагаться группы ячеек двух или трех типов, различающихся по окрестности и по правилам перехода;
- введением черт открытости, стохастичности и иерархичности архитектуры, что в целом приближает КА к классу нейронных сетей (можно еще добавить признак самомодифицируемости правил перехода, что приблизит КА к системам искусственного интеллекта и усилит эволюционный характер КА), но представляется все-таки отдаленной перспективой.

Указанные пути развития сохраняют значение и для "тяжелых" МКА. В меньшей степени специфичны для МКА тенденции, общие для современных САПР и носящие хотя и важный, но более технический характер: распараллеливание алгоритмов и вычислений, увеличение быстродействия за счет схем хеширования и предсказания, более эффективное использование оперативной памяти, кроссплатформенность, использование удаленных интерфейсов и поддержка сетевых вычислений, совершенствование форматов данных.

Описание МКА SoftCAM

Нами предлагается новый подход к созданию МКА, нашедший воплощение в архитектуре программы SoftCAM. Он неявно продолжает линию разработчиков Golly по скриптовой поддержке

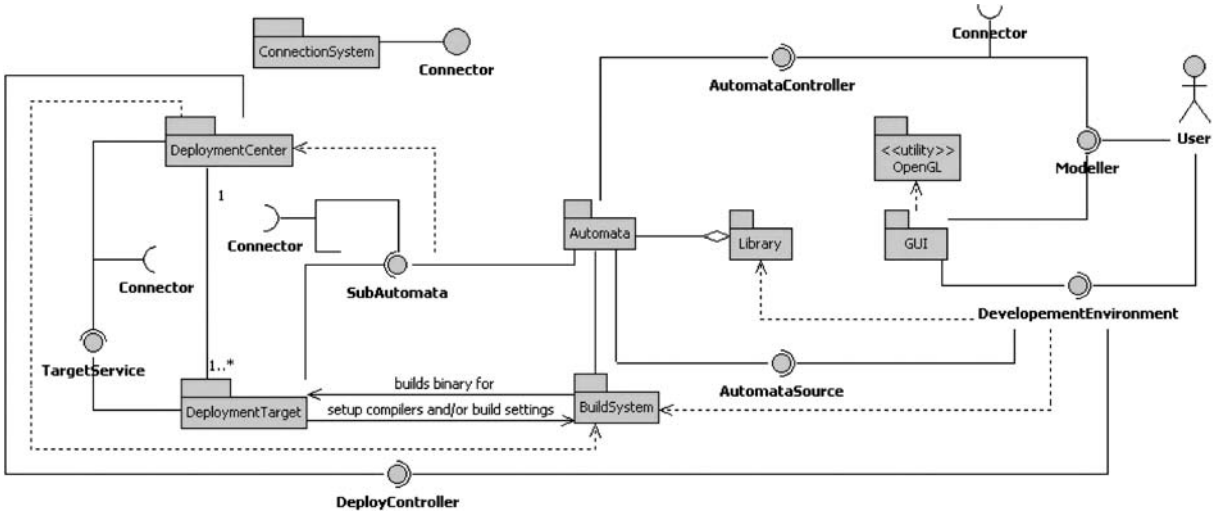

Рис. 3. Архитектура МКА SoftCAM в виде UML-диаграммы классов

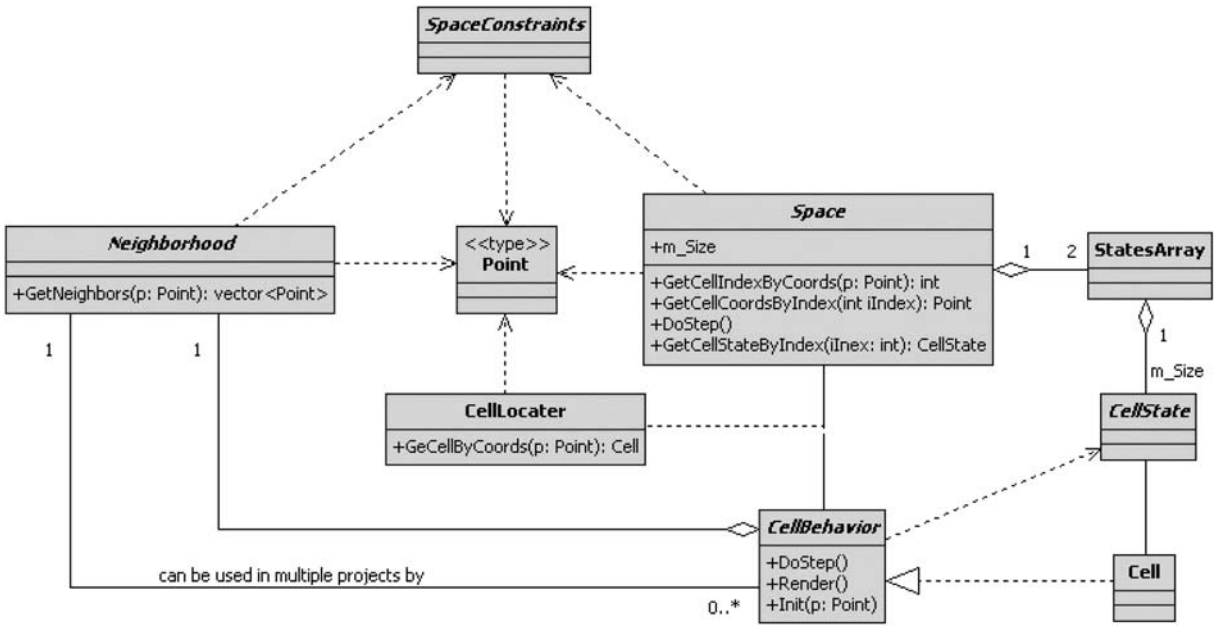

Рис. 4. Представление клеточного автомата в виде UML-диаграммы классов

(Perl и Python), делающее МКА более открытой и "доверительной" для пользователя. Однако интерпретатор кода с этих языков нуждается в дополнительной инсталляции, а общее быстродействие МКА неизбежно падает. Поэтому в SoftCAM встроен компилятор языка C++ и дополнительные библиотеки. Основные технические характеристики SoftCAM: объем на винчестере — 131 Мбайт, из них система компиляции — 110 Мбайт; занимаемая оперативная память при запуске — 9,3 Мбайт; быстродействие — 16 шагов конфигурации в секунду при размерах поля 640 × 480 и частоте процессора 3 ГГц (алгоритм симуляции не оптимизировался). Отметим, что МКА SoftCAM находится еще в стадии разработки.

Интерфейс программы скромный и ориентирован на пользователя, обладающего минимальными навыками программирования. Написанные пользователем внутри XML-структуры (рис. 2, см. четвертую сторону обложки) фрагменты кода, реализующие конкретную математическую модель, объединяются в один файл C++ кода, который затем компилируется и запускается. В программе также предусмотрены опции распределения вычислений по компьютерам локальной сети.

Ниже приведены две диаграммы классов для МКА SoftCAM, выполненные средствами программы StarUML. На рис. 3 показана архитектура в целом, а на рис. 4 — структура ядра симуляции МКА.

Заключение

Компромисс между широким спектром свободно распространяемых и немногочисленными академическими САПР в области клеточных автоматов, доступ к которым затруднен, позволит глубже их интегрировать в информационные и нанотехнологии. На основе обзора существующих решений в статье проанализированы перспективные пути развития МКА. Предложен новый подход к проектированию МКА, отличающийся большей гибкостью и открытостью по отношению к конечному пользователю, занимающемуся математическим моделированием распределенных физических, биологических или социальных объектов.

Список литературы

1. **Аладьев В. З.** Классические однородные структуры. Клеточные автоматы — CA. Palo Alto: Fultus Corporation, 2009. 536 с.

2. **Тоффоли Т., Марголус Н.** Машины клеточных автоматов. М.: Мир, 1991. 280 с.

3. **Wolfram S.** A New Kind of Science. N. Y.: Wolfram Media, 2002. 1197 с.

4. **Porod W.** Quantum-Dot Cellular Automata: Emerging Nanoelectronic Device Technologies // Proc. of Nano Engineering World Forum. Boston, Massachusetts, June 2003.

5. **Than O. and Buttgenbach S.** Simulation of anisotropic chemical etching of crystalline silicon using a cellular automata model // Sensors and actuators. Part a. October. 1994.

6. **Hopkins D.** Fun with Cellular Automata. URL: http://www.art.net/~hopkins/Don/art/cell.html.

7. **Legendi T.** Cellprocessors in Computer Architecture // Comp. Linguist. and Comp. Languages. 1976. Vol. 11. N 2. P. 147—167.

8. **Cannataro M.** et al. A parallel cellular automata environment on multicomputers for computational science // Parallel Computing. 1995. Vol. 21. P. 803—823.

9. **Наумов Л.** Метод введения обобщенных координат и инструментальное средство для автоматизации проектирования программного обеспечения вычислительных экспериментов с использованием клеточных автоматов // Дисс. на соиск. уч. ст. канд. техн. наук. СПбГУ ИТМО. 2007. 283 с.