УДК 519.876.5

С.С. ЛЕВИН

Национальный аэрокосмический университет им Н.Е. Жуковского «ХАИ», Украина

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ БИНАРНОЙ АВТОМАТНОЙ МОДЕЛИ

Предложена БА-модель, предназначенная для применения в имитационном моделирования поведения систем с большим количеством взаимодействующих объектов. Введены понятия систем и подсистем конечных автоматов. Рассмотрены теоретические аспекты поведения систем конечных автоматов. Предложена алгоритмизация решения проблемы синхронизатора с применением RB-деревьев. Описан событийный подход к моделированию при помощи систем конечных автоматов. Рассмотрены ограничения подхода. Приведено несколько базовых доказательств. Описано применение БА-модели для метода твердых сфер. Приведены иллюстрации к модельному примеру.

имитационное моделирование, система конечных автоматов, бинарная автоматная модель

Введение

Имитационное моделирование [1] используют в случаях, когда задача имеет слишком большую размерность или не поддается решению в явном (аналитическом) виде. В отличие от большинства технических методов, которые могут быть классифицированы в соответствии с научными дисциплинами, в которые они уходят своими корнями (например, с физикой или химией), имитационное моделирование применимо в любой отрасли науки. Для моделирования системы необходимо поставить искусственный эксперимент, отражающий основные условия моделируемой ситуации. Для этого необходим способ имитации искусственной последовательности происходящих в системе событий [2].

Целью статьи является определение системы связанных конечных автоматов и бинарной автоматной (БА) модели, разработанной специально для моделирования взаимодействия большего количества объектов, а также ее применение в имитационном моделировании различных явлений в тех случаях, когда:

1) элементарный объект моделирования можно представить в виде конечного автомата [3].

2) взаимодействие между объектами можно свести к парным (бинарным) взаимодействиям.

1. Прямое произведение *n* конечных автоматов

Введем в рассмотрение конечный автомат

$$L = <X, Y, S, s_0, \psi, \delta>,$$

где X – входные сигналы; Y – выходные сигналы; S – множество внутренних состояний; s_0 – начальное состояние ($s_0 \in S$); ψ – функция выходов $S \times X \rightarrow Y$; δ – функция переходов $S \times X \rightarrow S$.

Рассматриваются конечные автоматы, где $Y \equiv S$, т.е. конечные автоматы Мура [3].

Введем понятие *прямого произведения n* конечных автоматов с *одинаковым входным алфавитом*:

$$P = L_1 \times L_2 \times \ldots \times L_n : \begin{Bmatrix} A \\ B \end{Bmatrix}, \text{ где :}$$

$$\begin{aligned} &A \Rightarrow (\forall x \in X)(\forall s_1 \in S_1)(\forall s_2 \in S_2)\ldots(\forall s_n \in S_n): \\ &: \delta_{L_1 \times \ldots \times L_n}((s_1 \ldots s_n), x) = (\delta_1(s_1, x), \ldots, \delta_n(s_n, x)); \\ &B \Rightarrow (\forall x \in X)(\forall s_1 \in S_1)(\forall s_2 \in S_2)\ldots(\forall s_n \in S_n): \\ &: \psi_{L_1 \times \ldots \times L_n}((s_1 \ldots s_n), x) = (\psi_1(s_1, x), \ldots, \psi_n(s_n, x)). \end{aligned} \quad (1)$$

Иными словами, если конечный автомат P является прямым произведением n конечных автоматов $L_1, \ldots, L_n$, то:

– состояниями автомата P являются комбинации состояний исходных автоматов $L_1, \ldots, L_n$,;

– начальное состояние автомата P есть множество, состоящее из начальных состояний автоматов $L_1, \ldots, L_n$,;

– выходной алфавит автомата P – множество комбинаций выходных символов автоматов $L_1, \ldots, L_n$;

– функции переходов и выходов автомата P определены *покомпонентно, т.е. матрицей* ψ.

2. Система (коллектив) конечных автоматов

Под системой Σ из n конечных автоматов будем понимать прямое произведение $L_1 \times L_2 \times \ldots \times L_n$, при котором выход одного конечного автомата является входом в другой конечный автомат:

$$\Sigma(L) \Rightarrow \begin{pmatrix} (\forall L_i \in \Sigma)(\forall L_j \in \Sigma): \\ : \forall L_i \exists L_j \Rightarrow (X_j \equiv Y_i)(X_j = Y_i) \end{pmatrix}. \qquad (2)$$

Примечание. Под знаком "=" будем понимать равенство значений; под знаком "$\equiv$" будем понимать соответствие входов, выходов, параметров или множеств.

Под соответствием входа и выхода будем понимать *связь* между конечными автоматами. Исходя из введенных определений данная связь является однонаправленной, т.е. если L_i связан с L_j, то это не означает, что L_j связан с L_i. Введем обозначение связи:

$$(\forall L_i \in \Sigma)(\forall L_j \in \Sigma): \theta(L_i, L_J) \Rightarrow (X_j \equiv Y_i)(X_j = Y_i). \qquad (3)$$

3. Моделирование поведения системы Σ

Согласно определению конечного автомата [4, стр. 188], конечный автомат – это автомат [4, стр. 34], который обрабатывает входную цепочку и в любой момент времени находится в каком-то состоянии. Для моделирования поведения этого автомата необходимо определить этот момент времени.

Введем дополнительный скаляр t_{ai}, являющийся дополнительным внутренним состоянием автомата L_i. Изменение состояния t_{ai} будем осуществлять, используя дополнительный вход $t_{вхi}$. Данные на вход $t_{вхi}$ поступают от общего генератора T:

$$(\forall L_i \in \Sigma): t_{вхi} \equiv T. \qquad (4)$$

Принимая во внимание новое внутреннее состояние и новый вход, наложим на функции ψ и δ дополнительное ограничение.

Ограничение 1. Автомат L_i осуществляет переход в новое состояние $s_k \in S_i$ тогда и только тогда, когда $t_{ai} = T$.

Утверждение. Состояния S_i и выходные параметры Y_i системы автоматов Σ не зависят от введения параметра t_{ai}, и ограничения 1.

Доказательство. Доказательство удобно разбить на две части. Сначала рассмотрим поведение отдельно взятого автомата, а затем поведение системы Σ.

1. Покажем, что введение нового параметра t_{ai} никак не отразится на выходе $Y_i \in \psi_i$ конечного автомата L_i:

а) ψ – функция, зависящая от s, x;

б) δ – функция, зависящая от s, x.

При этом s, x не зависят от t_{ai}. Следовательно, ψ и δ не зависят от t_{ai}. Таким образом, состояние s_k, в которое перейдет автомат L_i в момент времени t_{ai}, не зависит от t_{ai}.

2. Покажем что поведение системы не изменится.

Прямое произведение автоматов $L_1 \times L_2 \times \ldots \times L_n$ не изменится, так как функции ψ и δ для каждого автомата не изменились. Следовательно, поведение системы в каждый момент времени t_{ai} не изменится относительно поведения этой же системы в тот же самый момент времени, однако без введения параметра t_{ai}.

4. Проблема синхронизатора

При моделировании поведения системы на цифровых устройствах невозможно добиться непрерывности изменения сигнала T [5]. Всегда существует минимальное изменение сигнала h, которое является характеристикой точности вычислений на конкретно взятой цифровой вычислительной системе. Следовательно,

$$T_{k+1} = T_k + h. \qquad (5)$$

Очевидно, что могут существовать такие T, при которых не будет осуществляться переход ни одного из автоматов L_i системы $\sum$ из одного состояния в другое, так как не найдется ни одного L_i, для кото-

рого выполнялось бы условие перехода (4):

$$\forall L_i \in \Sigma : t_{ai} \neq t_{вхi}. \quad (6)$$

Таким образом, при моделировании возникает проблема синхронизатора. Для обработки потоков возникающих взаимодействий необходимо на каждом шаге моделирования выбирать такое h, чтобы на каждом T_{k+1} происходил переход из одного состояния в другое хотя бы одного из автоматов системы. Следовательно, будем выбирать h исходя из следующей формулы:

$$\begin{cases} h_k = \min\limits_{i=1..n}(t_{ai} \in L_i) - T_{k-1}; \\ T_k = T_{k-1} + h_k. \end{cases} \quad (7)$$

Реализация поиска h_k связана с определением автомата L_i с минимальным t_{ai}. Если представлять поток событий в виде списка, то для систем, состоящих из большого количества автоматов, этот поиск на каждом шаге моделирования может занять значительное время, что с ростом количества событий приведет к существенному увеличению времени поиска, удаления и вставки событий в процессе работы алгоритма.

Одним из путей решения этой проблемы является замена прямого перебора событий в списке направленным поиском по дереву при вставке, удалении и поиске события [9]. Для этого каждому событию необходимо сопоставить уникальный номер – ключ. Наиболее удобным ключом является время наступления события. Такой ключ имеет и ряд недостатков:

1) он не целочисленный, что приводит к возрастанию времени проведения операции сравнения при поиске по сравнению с целочисленным ключом;

2) он не уникален, одновременно в системе могут находиться два события и более с одинаковым временем наступления.

Первый недостаток можно не принимать во внимание, так как возрастание времени сравнения незначительно по сравнению с выигрышем по скорости, получаемым при замене полного перебора направленным поиском. Второй же недостаток заставляет ввести не только первичный ключ, но и вторичный. Комбинация этих ключей должна абсолютно точно определять событие в потоке. В качестве вторичного ключа можно использовать уникальный номер события или, что эквивалентно, его адрес в памяти ЭВМ.

Для поиска события по первичному и вторичному ключу необходимо определенная организация памяти для хранения событий. При этом с точки зрения программной реализации наиболее простым методом является применение хэш-таблиц [7]. Но при этом возникает проблема выбора хэш-функции. При произвольном расположении событий в памяти ЭВМ выбрать хэш-функцию практически невозможно, т.е. при применении хэш-таблицы все события должны располагаться в памяти последовательно. Это приводит к существенным ограничениям не только со стороны количества событий (дефрагментация ОЗУ и, как следствие, невозможность последовательного выделения длинного непрерывного участка памяти), но и со стороны проведения операций вставки и удаления. При этом необходимо передвигать существенные объемы памяти с места на место, а также изменять вторичные ключи событий.

Альтернативой хэш-таблице является применение деревьев сортировок [7]. Однако эти деревья без применения дополнительных ухищрений через определенное время работы алгоритма просто вырождаются в линейный список. Т.е. необходима быстрая перестройка дерева при операциях удаления и вставки элементов. Такими возможностями обладают AVL-дерево, которое предложили два русских математика Адельсон-Вельский и Ландис, а также RB-дерево (или красно-черное дерево), разработанное Байером (Bayer) в 1972 г. При сравнении этих деревьев можно выделить следующие характерные особенности [6]:

1. Операции вставки и удаления для AVL-дерева и RB-дерева являются $O(log_2 n)$-операциями.

2. Вставка в AVL-дерево требует не более одной операции вращения, но удаление может потребовать до $log_2 n$ вращений. С другой стороны вставка в

RB-дерево может требовать двух вращений, но удаление – не более трех.

3. Максимальная высота AVL-дерева для дерева из *n* узлов меньше максимальной высоты RB-дерева для одного и того же количества узлов.

Исходя из вышеизложенного, учитывая разницу в количестве вращений для удаления элемента, *предлагается хранить дерево событий в виде RB-дерева*.

Под RB-деревом понимается двоичное дерево поиска, вершины которого разделены на красные (red) и черные (black). Таким образом, каждая вершина дерева хранит дополнительную информацию о ней – цвет [6] .

Дополнительно это дерево должно обладать следующими свойствами:

1) красные узлы могут иметь только черные дочерние узлы;

2) все пути от узла до любого листа, расположенного ниже в дереве, содержат одно и то же количество черных узлов.

При проведении операций удаления и вставки неизбежно нарушение указанных свойств дерева. Для их восстановления применяют операции вращения и перекрашивают некоторые вершины.

Операция вращения вправо преобразует левое дерево в правое заменой нескольких указателей. Правое дерево можно преобразовать в левое обратной операцией – вращением влево. Вершины ***x*** и ***y*** могут находиться в любом месте дерева. Буквы ***a***, ***b***, и ***c*** обозначают поддеревья.

Основной проблемой при реализации указанного изменения является выбор между стандартным классом RB-дерева и реализацией этого класса собственными силами. Разработка экспериментального ПО ведется автором на MS Visual C++ .NET. Следовательно, разработчик обладает мощным инструментом, разработанным и отлаженным таким гигантом производства ПО как фирма Microsoft. Это библиотека STL версии 7.0. В эту библиотеку входит множество классов-шаблонов, реализующих хранение данных по технологии RB-деревьев. Как уже указывалось, первичный ключ события не является уникальным, что неизбежно приводит к модификации классического алгоритма организации хранения RB-дерева при данном его приложении. К счастью оказалось, что библиотека STL содержит шаблон **multimap**. Реализация этого класса позволяет хранить ключи с одинаковыми значениями. Для идентификации же возможно использование вторичного ключа – указателя на событие. Учитывая изложенное, предпочтение было отдано классу **multimap**, с внесением следующих изменений. Класс был скопирован из библиотеки в исходные коды проекта. Все ***privat*** объявления заменены объявлениями ***protected***. Эта замена позволила модифицировать класс, используя механизм наследования, что намного проще модификации исходных кодов.

Анализ стандартных функций указанного класса показал невозможность изменения ключа той вершины, которая уже добавлена, т.е. единственным способом изменения является удаление вершины и добавление ее с новым ключом, что приведет к перестройке дерева. Одним из пунктов алгоритма расчета ближайшего события системы [8] является приведение системы в состояние, соответствующее текущему времени. При этом осуществляется проход по всему дереву с вычитанием времени наступления текущего события из времен всех остальных событий. Очевидно, что эта операция будет изменять ключ, что неизбежно будет приводить к перестройке RB-дерева. Решить эту проблему можно двумя способами:

1. Добавить в класс-наследник функцию полного обхода с изменением ключей. Легко заметить, что, если все ключи RB-дерева изменить на какое-либо одно и то же число, то перестраивать RB-дерево нет необходимости.

2. Ввести понятие глобального времени системы. При этом все события содержат не время наступления относительно текущего времени системы, а время наступления относительно начала работы системы, хотя при этом подходе теряется возможность неограниченного времени работы алгоритма

ввиду возможности переполнения разрядной сетки.

Выбран второй способ, так как он примерно в два раза проще в реализации на Microsoft Visual C++.

5. Уход от NP-полной задачи. Область применимости модели

Учитывая изложенное в параграфах 3 и 4, можно сделать вывод, что связанными являются только те автоматы, у которых t_{ai} равны. В общем случае это связь 1:m. Очевидно, что с каждым новым T_k связи некоторых автоматов будут перестраиваться, так как значение t_{ai} автомата будет меняться с переходом в каждое новое состояние. Если рассматривать систему в общем случае, учитывая все возможные связи, то возникает система автоматов, в которой каждый автомат связан с каждым автоматом. Для моделирования поведения такой системы необходимо для каждого автомата рассматривать все остальные автоматы. Это приводит к NP-полной задаче, решение которой при больших n с применением современной вычислительной техники практически неосуществимо (время решения задачи растет как n^{n-1}). Возможность решения этой задачи в общем виде в будущем также ставится под сомнение.

Ограничение 2. Ограничимся рассмотрением систем $\Sigma(L)$:

$$\Sigma(L) \Rightarrow \left\{ \begin{pmatrix} i = \overline{1,n} \\ L_i \in \Sigma \end{pmatrix} : \left(\max_{\substack{i=const \\ j=\overline{1,n} \\ j \neq i}} \left[\begin{matrix} (L_j \in \Sigma), \\ \| \theta(L_i, L_j) \| \end{matrix} \right] = 1 \right) \right\}. \quad (8)$$

Другими словами, рассматриваются только те системы, в которых связи между автоматами можно свести к парным взаимодействием, т.е. каждый из автоматов в конкретный момент времени связан только с одним конечным автоматом.

Для достижения условия (8) необходимо наложить дополнительные условия на функции ψ и δ. Эти ограничения будут специфическими для каждой конкретной моделируемой системы. Например, при моделировании социальных процессов можно рассматривать связь между двумя людьми; при моделировании газовых потоков считать взаимодействие частиц парным. Данная модель неприменима, например, для моделирования плазмы, так как в плазме каждая частица взаимодействует с n другими, а эти n взаимодействуют как между собой, так и с данной частицей.

Таким образом, формула (8) описывает *область применимости* предлагаемой модели.

6. Событийный подход к моделированию

Событием системы назовем момент перехода конечного автомата $L_i \in \Sigma(L)$ из состояния s_m в состояние s_k. Очевидно, что в любой момент времени T_k может происходить любое количество событий, большее 1, причем следует учитывать, что взаимодействия парные. Характеристиками события будем считать:

- конечные автоматы, участвующие в этом событии;
- время свершения события.

Очевидно, что общее количество событий равно количеству автоматов в системе, что следует из парности и однонаправленности связей в каждый момент времени.

Введем понятие *принадлежности события*.

Из пары конечных автоматов $L_i \in \Sigma(L)$ и $L_j \in \Sigma(L)$ событие $\zeta(L_i, L_j)$ принадлежат тому конечному автомату, для которого верно:

$$\Sigma(L): \forall L_i \left(\zeta(L_i, L_j) \subset L_i \right) \Rightarrow \theta(L_i, L_j) \neq \varnothing. \quad (9)$$

7. Действительные и недействительные события

Существует класс систем $\overline{\Sigma} \in \Sigma$, в процессе моделирования которых при наступлении события возможно не только изменение состояния автомата, которому принадлежит это событие, но также и изменение состояния связанного с ним автомата. Для определения того, когда это изменение необходимо

производить, а когда нет, введем понятие *действительного* и *недействительного* событий.

Событие $\zeta(L_i, L_j)$ будем считать *действительным*, если выполнятся следующее условие:

$$\overline{\Sigma}(L): \begin{cases} \theta(L_i, L_j) \neq \varnothing; \\ \theta(L_j, L_i) \neq \varnothing. \end{cases} \quad (10)$$

Соответственно, событие $\zeta(L_i, L_j)$ является *недействительным*, если условие (10) не выполняется.

8. Теорема о недействительном событии

Рассмотрим возможную ситуацию из трех автоматов в системе $\overline{\Sigma}(L)$. Пусть три автомата $A \in \overline{\Sigma}(L)$, $B \in \overline{\Sigma}(L)$, $C \in \overline{\Sigma}(L)$ на момент времени T_k взаимодействуют, как показано на рис. 1, а.

Пусть автоматы A и B взаимодействуют в момент времени T_{k+1}, а автоматы A и C в любой другой момент времени $T_n >> T_{k+1}$. Очевидно, что после взаимодействия возможны варианты связей, представленные на рис. 1, б, в.

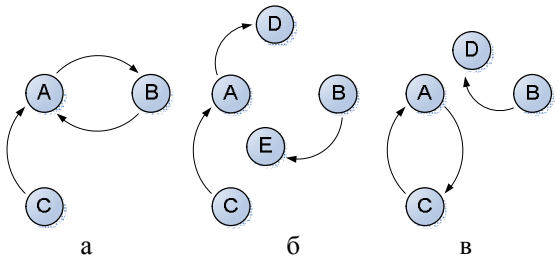

Рис. 1. Взаимодействие автоматов: а – в момент времени T_k; б – после взаимодействия автомат A не взаимодействует с автоматом C; в – после взаимодействия автомат A взаимодействует с автоматом C, а автомат B взаимодействует с каким-то еще автоматом, который не рассматривается

Для систем класса $\overline{\Sigma}$ действия при наступлении события $\zeta(C, B)$, необходимые для выполнения в ситуации, изображенной на рис. 1, а, нетривиальны. Очевидно, что новое состояние автомата C не зависит от выхода автомата A. Докажем, что оно не зависит от выходов какого-либо другого $L_i \in \Sigma(L)$.

Теорема о «*недействительном событии*». До наступления события $\zeta(C, A)$ автомат C не будет связан ни с одним другим автоматом системы $\overline{\Sigma}$.

Доказательство. Допустим, что новое внутреннее состояние зависит от какого-либо другого автомата. Для этого необходимо, чтобы

$$\zeta(C, L_j) \neq \varnothing. \quad (T1)$$

Однако выполнение T1 невозможно, так как автомат C связан с автоматом A, что говорит о том, что нет в системе такого другого события, связанного с автоматом C, которое может произойти раньше уже известного события, а именно $\zeta(C, A)$. Теорема доказана.

Следствие. При наступлении события $\zeta(C, A)$ и его недействительности необходимо рассчитать новое событие для автомата C. Если же производить изменение внутренних состояний автомата C, то эти изменения не будут зависеть ни от одного из автоматов системы $\overline{\Sigma}$.

9. Моделирование систем класса $\overline{\Sigma}$

Учитывая изложенное, процесс моделирования систем $\overline{\Sigma}$ можно свести к последовательности выполнения следующих (основных) шагов:

1. Определение события с минимальным временем наступления с использованием поиска по RB-дереву.
2. Определение действительности события по правилу (10).
3. Расчет внутренних состояний участвующих в событии конечных автоматов на базе следствия о «недействительном событии».
4. Формирование новых событий.

Процесс формирования нового события состоит в поиске связанных автоматов, т.е. в нахождении того автомата, с которым должен быть связан текущий. В общем случае для автомата, которому принадлежит рассматриваемое событие (9), необходимо рассмотреть все автоматы системы. С ростом количества автоматов и количества событий это процедура может занимать значительное время, что является основным недостатком приведенного под-

хода. Следовательно, для увеличения эффективности и скорости моделирования необходимо прежде всего оптимизировать процесс поиска автомата, с которым будет связан текущий. Одним из путей решения этой проблемы является *разбиение* системы с целью уменьшения количества автоматов, рассматриваемых для текущего автомата на каждом шаге итерационного процесса моделирования.

10. Разбиение системы $\overline{\Sigma}$. Конгруэнтность разбиения

По определению прямого произведения конечных автоматов (1) система $\overline{\Sigma}$ также является конечным автоматом.

Определение. Под *разбиением* π^k системы $\overline{\Sigma}$ на k групп будем понимать множество непересекающихся прямых произведений $R_i = L_i \times ... \times L_j$, объединение которых совпадает с $\overline{\Sigma}$:

$$\pi^k\left(\overline{\Sigma}\right) \Rightarrow \left\langle R_0 = L_i \times ... \times L_j \right\rangle ... \left\langle R_k = L_m \times ... \times L_n \right\rangle : \\ : R_0 \times ... \times R_k \in \overline{\Sigma}. \quad (11)$$

Минимальным элементом разбиения является элементарный конечный автомат системы, следовательно, для конечного автомата $\overline{\Sigma}$ разбиение π^k всегда будит конгруэнтным. Это говорит том, что в какое бы состояние не пришла система $\overline{\Sigma}$, оно всегда будет принадлежать только одному элементу разбиения. Назовем R_k *подсистемой* системы $\overline{\Sigma}$.

Учитывая введенное понятие разбиения на каждом новом шаге, при поиске пары для текущего автомата будем рассматривать только те автоматы, которые принадлежат одной и той же подсистеме R_k.

Очевидно, что при таком подходе возможны потери событий, связывающих автоматы различных подсистем. Следовательно, необходимо предусмотреть возможность миграции автоматов между подсистемами. Для этого необходимо:

1) добавить событие перехода между подсистемами;

2) ввести дополнительную функцию перехода между подсистемами.

11. Миграция автоматов между подсистемами. Функции переходов

Каждой возможности перехода из подсистемы $R_i \subset \overline{\Sigma}$ в подсистему $R_j \subset \overline{\Sigma}$ сопоставим функцию $P_{ij}(L)$, которая равна 1, если автомат L_i переходит из подсистемы $R_i \subset \overline{\Sigma}$ в подсистему при наступлении следующего события, и 0 в противном случае.

Подсистемы с функциями переходов удобно представить в виде графа (рис. 2). Каждой дуге графа сопоставим функцию переходов $P_{ij}(L)$. Эта функция будет различна для каждой конкретной системы и для каждого конкретного разбиения. Очевидно, что функция $P_{ij}(L)$ является характеристикой границ подсистемы. Будем считать, что количество исходящих дуг из вершины графа есть количество границ подсистемы.

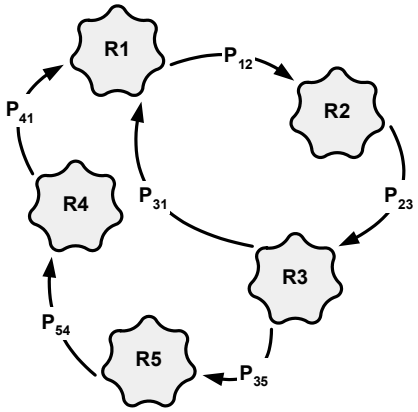

Рис. 2. Пример графа подсистем и функций переходов между ними

12. Применение БА-модели в методе твердых сфер для имитационного моделирования обтекания тела

Основной чертой предлагаемого подхода является возможность его использования для представления совокупности некоторого количества твердых сфер (частиц) в виде системы детерминированных автоматов с дискретным потоком моментов времени взаимодействия [8, 10]. Все необходимые распределения возникают как результат статистической обработки. В большей части работ по рассматриваемой тематике для упрощения расчетов обычно

предварительно принимают предположения о распределениях, после чего применяют в том или ином виде метод прямого моделирования Монте-Карло. Это позволяет резко повысить скорость расчетов, но дополнительно вносит неустранимые погрешности, связанные с априорностью статистических распределений [11, 12].

Эффективность процесса моделирования принципиально зависит от затрат машинного времени на обработку последовательности событий. Для оптимизации процесса обработки последовательности событий удобно воспользоваться БА-моделью. При использовании этой модели каждая частица интерпретируется как конечный автомат, состояния этого автомата интерпретируются как физические *скорости* и *координаты* в пространстве. Область тока газа интерпретируется как система $\overline{\Sigma}$. Под разбиением π^k системы $\overline{\Sigma}$ в этом случае будем понимать *сетку тока* – сетку непересекающихся прямоугольных ячеек, на которые разбита область тока. *Событиями* системы являются:

– парное столкновение частиц;

– перелет частиц из подсистемы в подсистему – пересечение границ ячеек сетки тока;

– столкновение частицы с обтекаемым объектом;

– влет частицы в область тока и вылет частицы из нее.

Применение описанного и обоснованного выше метода позволяет осуществить моделирование в реальном времени для количества частиц порядка 10^5 – 10^6 на одном ПК PIV 2GHz, 512 MB RAM (доступные на настоящий момент публикации по аналогичным разработкам, например, [14, 15], показывают результаты порядка 10^3 – 10^4 на существенно более мощной аппаратуре).

На базе изложенного подхода разработан пакет программ построения трехмерной сетки и трехмерного моделирования течения газа. Для влета частиц в область моделирования и вылета из нее используется модель тора [12].

На рис. 3 – 6 приведены иллюстрации к модельному примеру обтекания цилиндра.

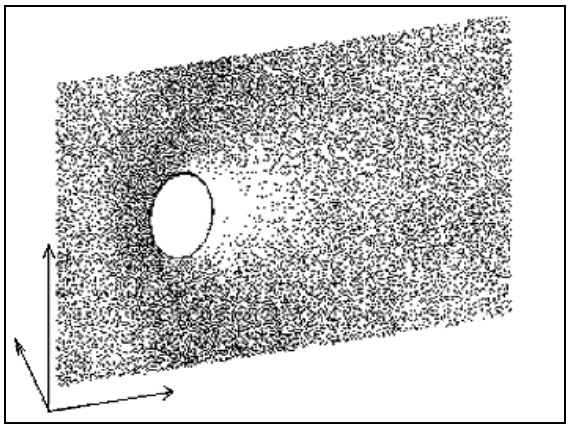

Рис. 3. Трехмерное обтекание однослойного цилиндра. Распределение модельных частиц

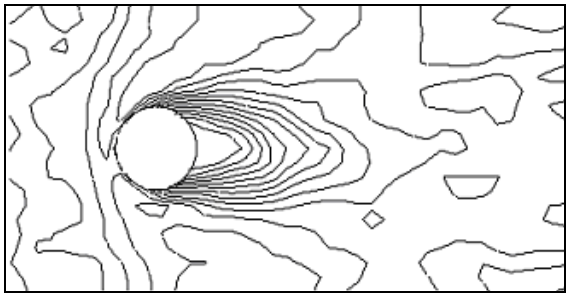

Рис. 4. Распределение энергий (характеристики температуры) модельных частиц при обтекании цилиндра

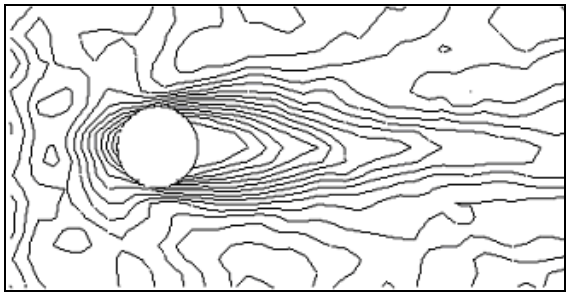

Рис. 5. Распределение проекции скоростей модельных частиц на ось абсцисс

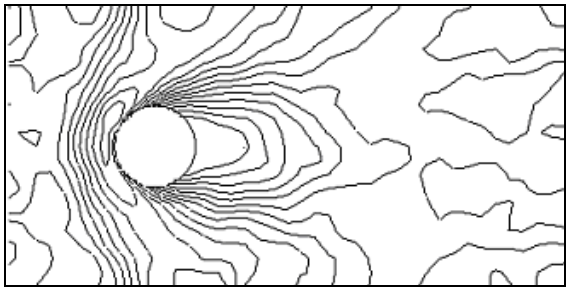

Рис. 6. Распределение концентраций модельных частиц при обтекании цилиндра

На рис. 4 – 6 заметны вихри, возникающие за обтекаемым объектом. При наблюдении за картинами в динамике можно заметить возникающие дорожки Кармана [13].

В дальнейшем планируется продвижения в трех направлениях:

– разработка методов численной интерпретации результатов обтекания и перевода полученных качественных результатов в количественные в системе СИ;

– развитие модели с целью описания имитационного моделирования химических реакций в движущихся потоках;

– рассмотрение структуры элементарного объекта в виде набора личностных качеств и запасов сведений, умений для моделирования большего коллектива субъектов – предсказания результатов социологических исследований.

Литература

1. Аверилл М. Лоу, В. Дэвид Кельтон. Имитационное моделирование. Классика CS. 3-е издание – К.: Питер, 2004 – 864 с.

2. Левин С.С. Имитационное моделирование течения газа в виде систем конечных автоматов // Открытые информационные и компьютерные интегрированные технологии. – Х: Нац. аэрокосм. ун-т «ХАИ», 2004. – Вып. 24 – С. 33 – 40.

3. Кудрявцев В.Б., Алешин С.В., Подколзин А.С. Введение в теорию автоматов. – М.: Наука, 1985. – 320 с.

4. Толковый словарь по вычислительным системам. – М: Машиностроение, 1991. – 448 с.

5. Корнеев В.В., Киселев А.В. Современные микропроцессоры. 3-е изд. – С.-Пб.: BHV, 2003. – 448 с.

6. Хэлфилд Р., Кирби Л. Искусство программирования на С. – С.-Пб.: ДиаСофт, 2001. – 728 с.

7. Кормен Т., Лейзерсон Ч., Риверст Р. Алгоритмы. Построение и анализ. – М.: МЦНМО, 2001. – 955 с.

8. Левин С.С.,Чернышев Ю.К. Алгоритмизация прямого моделирования методом частиц течения газа по каналам сложной формы при малых числах Кнудсена // Открытые информационные и компьютерные интегрированные технологии. – Х: Нац. аэрокосм. ун-т «ХАИ». – 2002. – Вып. 14. – С. 54 – 60.

9. Левин С.С., Чернышев Ю.К. Применение RB-деревьев для оптимизации алгоритма обработки потока событий при имитационном моделировании течения газа методом частиц // Открытые информационные и компьютерные интегрированные технологии. – Х: Нац. аэрокосм. ун-т «ХАИ». – 2003. – Вып. 17. – С. 77 – 82.

10. Левин С.С. Оптимизация геометрических структур, используемых при учете нежелательных краевых эффектов в подсистемах модельных частиц при имитационном моделировании течения газа // Междунар. НТК «Информационные компьютерные технологии в машиностроении, ИКТМ-2003». – Х.: Нац. аэрокосм. ун-т «ХАИ». – 2003. – С. 184.

11. Харлоу Ф.Х.. Численный метод крупных частиц в ячейках для задач гидродинамики. Вычислительные методы в гидродинамике. – М.: Мир, 1967. – 384 с.

12. Берд Г. Молекулярная газовая динамика. – М.: Мир, 1981. – 313 с.

13. Лойцянский Л.Г. Механика жидкости и газа. М.: Наука, 1978. – 676 с.

14. Берлин Ал., Балабаев Н. Имитация свойств твердых тел и жидкостей методами компьютерного моделирования // Соросовский образовательный журнал. – 1997. – Вып. 11. – С. 85 – 92.

15. Ben Tsou, Kevin Wayne. Molecular Dynamics Simulation of Hard Spheres. – 2004. – [Электрон. ресурс]. – Режим доступа: http://www.cs.princeton.edu/courses/archive/fall04/cos226/assignments/collisions.html.

Поступила в редакцию 11.05.05

Рецензент: д-р техн. наук, проф. А.Ю. Соколов, Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Харьков.