МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2010 №4(10)

Свидетельство о регистрации: ПИ №ФС 77-33762
от 16 октября 2008 г.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Адрес редакции: 634050, г. Томск, пр. Ленина, 36
E-mail: vestnik_pdm@mail.tsu.ru

В журнале публикуются результаты фундаментальных и прикладных научных исследований отечественных и зарубежных ученых, включая студентов и аспирантов, в области дискретной математики и её приложений в криптографии, компьютерной безопасности, кибернетике, информатике, программировании, теории надежности, интеллектуальных системах.

Периодичность выхода журнала: 4 номера в год.

ООО «Издательство научно-технической литературы»
634050, Томск, пл. Ново-Соборная, 1, тел. (3822) 533-335
Редактор *Н. И. Шидловская*
Верстка *И. А. Панкратовой*

Изд. лиц. ИД. №04000 от 12.02.2001. Подписано к печати 10.12.2010.
Формат $60 \times 84\frac{1}{8}$. Бумага офсетная. Печать офсетная. Гарнитура «Таймс».
Усл. п. л. 13,8. Уч.-изд. л. 15,47. Тираж 300 экз. Заказ №18.

Отпечатано в типографии «М-Принт», г. Томск, ул. Пролетарская, 38/1

СОДЕРЖАНИЕ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

DOI 10.17223/20710410/10/1

УДК 519.1

ЧИСЛО НЕУПОРЯДОЧЕННЫХ ПОКРЫТИЙ КОНЕЧНОГО МНОЖЕСТВА ПОДМНОЖЕСТВАМИ ФИКСИРОВАННОГО РАЗМЕРА

Р. М. Ганопольский

Тюменский государственный университет, г. Тюмень, Россия

E-mail: rodion@utmn.ru

Рассматривается новый вид комбинаторных чисел, исчисляющих количество покрытий конечного множества подмножествами с заданными мощностями. Доказывается ряд соотношений и тождеств. Вычисляются некоторые суммы этих чисел. Приводятся частные случаи новых комбинаторных чисел при определенных значениях коэффициентов и интерпретация этих чисел.

Ключевые слова: *покрытие, конечное множество, комбинаторные числа.*

1. Основные понятия

Пусть дано конечное множество $\mathbf{X}$ мощности n и семейство $\mathbf{S}$ его непустых несовпадающих подмножеств, являющееся покрытием изначального множества:

$$\begin{gathered} |\mathbf{X}| = n, \\ \mathbf{S} \subset \mathfrak{P}'(\mathbf{X}), \\ \bigcup_{U_\alpha \in \mathbf{S}} U_\alpha = \mathbf{X}, \end{gathered} \tag{1}$$

где $\mathfrak{P}'(\mathbf{X})$ — семейство всех непустых подмножеств множества $\mathbf{X}$ [1]. Множество всех покрытий $\mathbf{S}$, удовлетворяющих условию (1), обозначим $\mathbf{C}(\mathbf{X})$:

$$\mathbf{C}(\mathbf{X}) = \left\{ \mathbf{S} \subset \mathfrak{P}'(\mathbf{X}) : \bigcup_{U_\alpha \in \mathbf{S}} U_\alpha = \mathbf{X} \right\}, \tag{2}$$

а множество покрытий, содержащих только собственные подмножества $\mathbf{X}$, обозначим $\mathbf{C}'(\mathbf{X})$:

$$\mathbf{C}'(\mathbf{X}) = \{ \mathbf{S} \in \mathbf{C}(\mathbf{X}) : \mathbf{X} \notin \mathbf{S} \}. \tag{3}$$

Для множества покрытий, содержащих ровно k подмножеств, где $1 \leqslant k \leqslant 2^n - 1$, используем обозначение $\mathbf{C}_k(\mathbf{X})$:

$$\mathbf{C}_k(\mathbf{X}) = \{ \mathbf{S} \in \mathbf{C}(\mathbf{X}) : |\mathbf{S}| = k \}. \tag{4}$$

В качестве примера конечного множества $\mathbf{X}$ можно взять множество целых чисел от 1 до n:

$$\mathbf{X}_n = \{1, 2, \ldots, n\}. \tag{5}$$

Согласно работам [1, 2], мощность множества $\mathbf{C}(\mathbf{X})$, равную количеству всех покрытий конечного множества, можно вычислить по формуле (для пустого множества по определению мощность множества покрытий равна 1):

$$|\mathbf{C}(\mathbf{X})| = \frac{1}{2}\sum_{i=0}^{n}(-1)^i C_n^i 2^{2^{n-i}}, \tag{6}$$

мощность множества $\mathbf{C}'(\mathbf{X})$ — по формуле

$$|\mathbf{C}'(\mathbf{X})| = \frac{1}{2}\sum_{i=0}^{n}(-1)^i C_n^i 2^{2^{n-i}} - \frac{2^{2^n}}{4}, \tag{7}$$

а мощность множества $\mathbf{C}_k(\mathbf{X})$ — по формуле

$$|\mathbf{C}_k(\mathbf{X})| = \sum_{i=0}^{n}(-1)^i C_n^i C_{2^{n-i}-1}^k, \tag{8}$$

где C_n^i — биномиальный коэффициент:

$$C_n^i = \frac{n!}{i!(n-i)!} = C_n^{n-i}. \tag{9}$$

Зафиксируем мощности подмножеств, составляющих семейство $\mathbf{S}$ (1), и введем обозначение

$$\mathbf{S}(k_1, k_2, \ldots, k_n) \tag{10}$$

для покрытия, содержащего k_i подмножеств мощности i (для всех i). Мощность такого покрытия равна

$$|\mathbf{S}(k_1, k_2, \ldots, k_n)| = \sum_{i=1}^{n} k_i.$$

Аналогично обозначениям (2)–(4) введем обозначение для множества семейств (10):

$$\mathbf{C}(k_1, k_2, \ldots, k_n)(\mathbf{X}) = \{\mathbf{S}(k_1, k_2, \ldots, k_n)\}. \tag{11}$$

Мощность введенного множества обозначим следующим образом:

$$|\mathbf{C}(k_1, k_2, \ldots, k_n)(\mathbf{X})| = {}_nN(k_1, k_2, \ldots, k_n). \tag{12}$$

В случае, когда не равны нулю только несколько k_i, будем пользоваться другим обозначением:

$${}_nN_{l_1 l_2 \cdots l_m}^{k_1 k_2 \cdots k_m}, \tag{13}$$

где k_i — это количество подмножеств мощности l_i в покрытии. Например:

- ${}_4N_2^3 = {}_4N(0,3,0,0)$ — количество покрытий множества мощности 4, состоящих из трех подмножеств мощности 2;
- ${}_5N_{23}^{31} = {}_5N(0,3,1,0,0)$ — количество покрытий множества мощности 5, состоящих из трех подмножеств мощности 2 и одного подмножества мощности 3.

Подобное обозначение введем и для множества семейств (11):

$${}_n\mathbf{C}_{l_1 l_2 \cdots l_m}^{k_1 k_2 \cdots k_m}(\mathbf{X}). \tag{14}$$

Для фиксирующих коэффициентов в (12) существуют следующие ограничения:

$$0 \leqslant k_i \leqslant C_n^i,$$

а для коэффициентов в (13), (14) — следующие:

$$\begin{gathered} 1 \leqslant m \leqslant n, \\ 1 \leqslant l_1 < l_2 < \cdots < l_m \leqslant n, \\ 1 \leqslant k_i \leqslant C_n^{l_i}. \end{gathered}$$

Рассмотрим множество $\mathbf{X}$ мощности 3 в виде набора чисел $\mathbf{X}_3 = \{1, 2, 3\}$, а также все различные покрытия $\mathbf{X}_3$, состоящие только из подмножеств мощности 2:

$$\begin{gathered} \{1,2\} \cup \{2,3\} = \mathbf{X}_3, \\ \{1,3\} \cup \{2,3\} = \mathbf{X}_3, \\ \{1,2\} \cup \{1,3\} = \mathbf{X}_3, \\ \{1,2\} \cup \{2,3\} \cup \{1,3\} = \mathbf{X}_3. \end{gathered}$$

Таким образом, комбинаторные числа вида ${}_3N_2^k = {}_3N(0,k,0)$ равны

$${}_3N(0,1,0) = 0; \quad {}_3N(0,2,0) = 3; \quad {}_3N(0,3,0) = 1.$$

Рассмотрим теперь все покрытия $\mathbf{X}_3$, состоящие из подмножеств мощности 2 и 3:

$$\begin{gathered} \{1,2\} \cup \{1,2,3\} = \mathbf{X}_3, \\ \{1,3\} \cup \{1,2,3\} = \mathbf{X}_3, \\ \{2,3\} \cup \{1,2,3\} = \mathbf{X}_3, \\ \{1,2\} \cup \{2,3\} \cup \{1,2,3\} = \mathbf{X}_3, \\ \{1,3\} \cup \{2,3\} \cup \{1,2,3\} = \mathbf{X}_3, \\ \{1,2\} \cup \{1,3\} \cup \{1,2,3\} = \mathbf{X}_3, \\ \{1,2\} \cup \{2,3\} \cup \{1,3\} \cup \{1,2,3\} = \mathbf{X}_3. \end{gathered}$$

Следовательно, комбинаторные числа вида ${}_3N_{23}^{kl} = {}_3N(0,k,l)$, где $k \neq 0$ и $l \neq 0$, равны

$${}_3N(0,1,1) = 3; \quad {}_3N(0,2,1) = 3; \quad {}_3N(0,3,1) = 1.$$

С помощью перебора всех вариантов покрытий можно получить и остальные коэффициенты вида ${}_3N(m,k,l)$:

$$\begin{gathered} {}_3N(1,0,0) = 0, \quad {}_3N(2,0,0) = 0, \quad {}_3N(3,0,0) = 1, \quad {}_3N(1,1,0) = 3, \\ {}_3N(2,1,0) = 6, \quad {}_3N(3,1,0) = 3, \quad {}_3N(1,2,0) = 9, \quad {}_3N(2,2,0) = 9, \\ {}_3N(3,2,0) = 3, \quad {}_3N(1,3,0) = 3, \quad {}_3N(2,3,0) = 3, \quad {}_3N(3,3,0) = 1, \\ {}_3N(1,0,1) = 3, \quad {}_3N(2,0,1) = 3, \quad {}_3N(3,0,1) = 1, \quad {}_3N(1,1,1) = 9, \\ {}_3N(2,1,1) = 9, \quad {}_3N(3,1,1) = 3, \quad {}_3N(1,2,1) = 9, \quad {}_3N(2,2,1) = 9, \\ {}_3N(3,2,1) = 3, \quad {}_3N(1,3,1) = 3, \quad {}_3N(2,3,1) = 3, \quad {}_3N(3,3,1) = 1. \end{gathered}$$

2. Соотношения и тождества

Все соотношения и тождества для комбинаторных чисел (12), (13) сначала будем получать для простого вида чисел, когда все подмножества покрытия имеют одинаковую мощность (то есть для комбинаторных чисел ${}_nN_k^l$). В этом случае фиксируем количество и мощности множеств.

Найдем зависимость комбинаторных чисел ${}_nN_l^k$ от биномиальных коэффициентов C_j^i и комбинаторных чисел предыдущих порядков вида ${}_{n-i}N_l^k$. Множество $\mathbf{X}$ мощности n имеет C_n^l подмножеств мощности l. Из этих подмножеств можно выбрать $C_{C_n^l}^k$ способами k подмножеств. Не все эти наборы подмножеств будут являться покрытием множества $\mathbf{X}$. Некоторые из них являются покрытиями собственных подмножеств исходного множества $\mathbf{X}$. Следовательно, число покрытий множества $\mathbf{X}$ k подмножествами мощности l будет равно $C_{C_n^l}^k$ минус число всех покрытий собственных подмножеств тем же количеством подмножеств с такими же мощностями:

$$\left|{}_n\mathbf{C}_l^k\left(\mathbf{X}\right)\right| = C_{C_n^l}^k - \sum_{U_\alpha\subset\mathbf{X}}\left|{}_{|U_\alpha|}\mathbf{C}_l^k\left(U_\alpha\right)\right|. \tag{15}$$

У множества мощности n существует C_n^i подмножеств мощности $(n-i)$. Количество покрытий множества мощности $n-i$ — это комбинаторное число ${}_{n-i}N_l^k$. Окончательно получаем рекуррентное соотношение для числа покрытий множества:

$${}_nN_l^k = C_{C_n^l}^k - {}_{n-1}N_l^kC_n^1 - {}_{n-2}N_l^kC_n^2 - {}_{n-3}N_l^kC_n^3 - \cdots \tag{16}$$

Вычитаемые в (16) берутся пока ${}_{n-i}N_l^k > 0$, то есть пока $C_{n-i}^l \geqslant k$. Таким образом, формулу (16) можно записать в компактном виде

$${}_nN_l^k = C_{C_n^l}^k - \sum_{i\geqslant 1}^{C_{n-i}^l\geqslant k} {}_{n-i}N_l^kC_n^i. \tag{17}$$

Соберем в соотношении (17) все слагаемые с комбинаторными числами N_l^k с одной стороны, а биномиальный коэффициент $C_{C_n^l}^k$ — с другой:

$$C_{C_n^l}^k = {}_nN_l^k + \sum_{i\geqslant 1}^{C_{n-i}^l\geqslant k} {}_{n-i}N_l^kC_n^i = \sum_{i\geqslant 0}^{C_{n-i}^l\geqslant k} {}_{n-i}N_l^kC_n^i. \tag{18}$$

Выражение (18) представляет собой биномиальное преобразование [3]. Используя обратное биномиальное преобразование, получим

$${}_nN_l^k = \sum_{i\geqslant 0}(-1)^iC_n^iC_{C_{n-i}^l}^k = C_{C_n^l}^k + \sum_{i\geqslant 1}(-1)^iC_n^iC_{C_{n-i}^l}^k. \tag{19}$$

Выведем аналогичное (19) выражение для общего случая. Из множества $\mathbf{X}$ мощности n можно выбрать $C_n^{l_i}$ способами подмножество мощности l_i. Из этого числа подмножеств выбрать k_i подмножеств можно $C_{C_n^{l_i}}^{k_i}$ способами. Так как все задаваемые мощности l_i не равны между собой, то выбрать семейство, содержащее k_1 подмножеств мощности l_1, k_2 подмножеств мощности l_2 и т. д., можно следующим числом способов:

$$\prod_{i=1}^{m} C_{C_n^{l_i}}^{k_i}.$$

Таким образом, в общем случае выражение (15) будет иметь следующий вид:

$$\left|{}_n\mathbf{C}^{k_1k_2\cdots k_m}_{l_1\,l_2\cdots l_m}(\mathbf{X})\right| = \prod_{i=1}^{k=m} C^{k_i}_{C^{l_i}_n} - \sum_{U_\alpha\subset\mathbf{X}} \left|{}_{|U_\alpha|}\mathbf{C}^{k_1k_2\cdots k_m}_{l_1\,l_2\cdots l_m}(U_\alpha)\right|.$$

У множества мощности n существует C^i_n подмножеств мощности $n-i$. Количество покрытий множества мощности $n-i$ подмножествами с фиксированными мощностями есть комбинаторное число ${}_{n-i}N^{k_1k_2\cdots k_m}_{l_1\,l_2\cdots l_m}$. Получаем рекуррентное соотношение для числа покрытий множества:

$${}_nN^{k_1k_2\cdots k_m}_{l_1\,l_2\cdots l_m} = \prod_{i=1}^{m} C^{k_i}_{C^{l_i}_n} - \sum_{i\geqslant 1} C^i_n\,{}_{n-i}N^{k_1k_2\cdots k_m}_{l_1\,l_2\cdots l_m}, \tag{20}$$

где параметр суммы i ограничен сверху соотношениями

$$\forall j\in\{1,2,\ldots,m\}\left(C^{l_j}_{n-i}\geqslant k_j\right).$$

Избавившись в правой части (20) от комбинаторных чисел N с помощью биномиального преобразования, получим нерекуррентное выражение для ${}_nN^{k_1k_2\cdots k_m}_{l_1\,l_2\cdots l_m}$, содержащее только биномиальные коэффициенты:

$${}_nN^{k_1k_2\cdots k_m}_{l_1\,l_2\cdots l_m} = \prod_{i=1}^{m} C^{k_i}_{C^{l_i}_n} + \sum_{i\geqslant 1}(-1)^i C^i_n \prod_{j=1}^{m} C^{k_j}_{C^{l_j}_{n-i}}. \tag{21}$$

Найдем зависимость комбинаторных чисел ${}_nN^k_l$ от комбинаторных чисел предыдущих порядков вида ${}_{n-i}N^{k-1}_l$, где $0\leqslant i\leqslant l$. Рассмотрим произвольное покрытие множества мощности $n-i$, состоящее из $k-1$ подмножеств мощности l. Теперь добавим в него еще одно множество мощности l таким образом, чтобы получившееся семейство стало покрытием для множества мощности n. Для этого добавляемое множество должно содержать i элементов, не входящих в изначальное множество мощности $n-i$, и, следовательно, $l-i$ элементов, входящих в него.

Рассмотрим случай, когда $i=0$. У множества мощности n существует C^l_n подмножеств мощности l. Из них $k-1$ уже входят в покрытие. Значит, в это покрытие можно добавить одно из C^l_n-k+1 оставшихся подмножеств. Так как у множества мощности n существует ${}_nN^{k-1}_l$ покрытий, состоящих из $k-1$ подмножеств мощности l, то добавление еще одного подмножества дает

$$\left(C^l_n-k+1\right){}_nN^{k-1}_l \tag{22}$$

покрытий этого множества k подмножествами мощности l.

В случае, когда $i>0$, существует ${}_{n-i}N^{k-1}_l$ покрытий множества мощности $n-i$ подмножествами мощности l в количестве $k-1$. У множества мощности n существует C^{n-i}_n подмножеств мощности $n-i$. Способов выбрать $l-i$ элементов из множества мощности $n-i$ равно C^{l-i}_{n-i}. Таким образом, количество возможных вариантов получения из покрытий множества мощности $n-i$ покрытий множества мощности n равно

$$C^{l-i}_{n-i}C^{n-i}_n\,{}_{n-i}N^{k-1}_l = C^{l-i}_{n-i}C^i_n{}_{n-i}N^{k-1}_l. \tag{23}$$

Так как добавить в покрытие можно любое из k подмножеств мощности l, то, просуммировав (22) и (23) для всех i, окончательно получаем

$${}_nN^k_l = \frac{1}{k}\left[\left(C^l_n-k+1\right){}_nN^{k-1}_l + \sum_{i=1}^{l} C^{l-i}_{n-i}C^i_n{}_{n-i}N^{k-1}_l\right]. \tag{24}$$

Вынося одно слагаемое, получаем другой вид выражения (24):

$$ {}_nN_l^k = \frac{1}{k}\sum_{i=0}^{l} C_{n-i}^{l-i} C_n^i {}_nN_l^{k-1} - \frac{k-1}{k}{}_nN_l^{k-1}. \tag{25} $$

С помощью аналогичных рассуждений можно вывести соотношение для общего случая

$$ {}_nN_{l_1\cdots l_j\cdots l_m}^{k_1\cdots k_j\cdots k_m} = \frac{1}{k_j}\left[\left(C_n^{l_j} - k_j + 1\right){}_nN_{l_1\cdots l_j\cdots l_m}^{k_1\cdots(k_j-1)\cdots k_m} + \sum_{i=1}^{l_j} C_{n-i}^{l_j-i} C_n^i N_{l_1\cdots l_j\cdots l_m}^{k_1\cdots(k_j-1)\cdots k_m}\right], \tag{26} $$

где $1 \leqslant j \leqslant m$.

Получим выражение (24) с помощью ряда преобразований выражения (19). Подставив тождество

$$ C_{C_n^l}^k = \frac{C_n^l - k + 1}{k} C_{C_n^l}^{k-1} $$

в (19), получим

$$ {}_nN_l^k = \frac{1}{k}\left[\sum_{i\geqslant 0}(-1)^i C_n^i \left(C_{n-i}^l - k + 1\right) C_{C_{n-i}^l}^{k-1}\right]. $$

Теперь подставим вместо $C_{C_{n-i}^l}^{k-1}$ правую часть соотношения (18):

$$ {}_nN_l^k = \frac{1}{k}\left[\sum_{i\geqslant 0}(-1)^i C_n^i \left(C_{n-i}^l - k + 1\right)\sum_{j\geqslant 0} {}_{n-j-i}N_l^{k-1} C_{n-i}^j\right]. $$

Произведем замены $i + j \to i$ и $i \to j$ и поменяем порядок суммирования:

$$ {}_nN_l^k = \frac{1}{k}\left[\sum_{i\geqslant 0} {}_{n-i}N_l^{k-1}\sum_{j=0}^{i}(-1)^j C_{n-j}^{i-j} C_n^j \left(C_{n-j}^l - k + 1\right)\right]. $$

Разложим выражение под знаком суммы на два слагаемых:

$$ \sum_{j=0}^{i}(-1)^j C_{n-j}^{i-j} C_n^j C_{n-j}^l - (k-1)\sum_{j=0}^{i}(-1)^j C_{n-j}^{i-j} C_n^j. \tag{27} $$

При $i = 0$ вторая сумма равна 1. Для вычисления сумм воспользуемся тождествами для биномиальных коэффициентов [3]:

$$ \begin{aligned} C_s^r &= C_s^{s-r}, \\ C_s^r C_{s-r}^t &= C_{r+t}^r C_s^{r+t}, \\ C_s^r &= (-1)^r C_{r-s-1}^r, \\ \sum_r C_s^r C_p^{t-r} &= C_{s+p}^t. \end{aligned} \tag{28} $$

Преобразуем первую сумму в (27):

$$ \begin{aligned} \sum_{j=0}^{i}(-1)^j C_{n-j}^{i-j} C_n^j C_{n-j}^l &= C_n^i \sum_{j=0}^{i}(-1)^j C_i^j C_{n-j}^{n-l-j} = C_n^i \sum_{j=0}^{i}(-1)^j C_i^j (-1)^{n-l-j} C_{-l-1}^{n-l-j} = \\ &= C_n^i (-1)^{n-l} C_{i-l-1}^{n-l} = C_n^i C_{n-i}^{n-l} = C_n^i C_{n-i}^{l-i}. \end{aligned} \tag{29} $$

Вторую сумму вычислим для $i > 0$:

$$\sum_{j=0}^{i}(-1)^j C_{n-j}^{i-j} C_n^j = \sum_{j=0}^{i}(-1)^j C_{n-j}^{n-i} C_n^j = (-1)^i \sum_{j=0}^{i} C_{i-n-1}^{i-j} C_n^j = (-1)^i C_{i-1}^i = 0. \tag{30}$$

Воспользовавшись (29) и (30), а также учитывая значение второй суммы при $i = 0$, получаем выражение (24).

Рекуррентное соотношение (24) при $k = 1$ и $n = l$ дает

$${}_l N_l^1 = {}_0 N_l^0.$$

Таким образом, получаем

$${}_0 N_l^0 = 1,$$

что можно интерпретировать следующим образом: число покрытий пустого множества семейством подмножеств, не содержащим ни одного подмножества, равно 1.

3. Суммы

Вычислим сумму всех комбинаторных чисел ${}_n N(k_1 k_2 \cdots k_n)$ для определенного n. Для это воспользуемся выражением (21)

$$\sum_{k \geqslant 0} {}_n N(k_1, k_2, \ldots, k_n) = \sum_{k \geqslant 0} \sum_{i}^{n} (-1)^i C_n^i \prod_{j=1}^{n} C_{C_{n-i}^j}^{k_j},$$

где $k \geqslant 0$ — общее условие для всех чисел из набора $(k_1, k_2, \ldots, k_n)$. Поменяем порядок суммирования, вынесем один множитель из произведения и воспользуемся тождеством $\sum\limits_i C_n^i = 2^n$:

$$\sum_{k \geqslant 0} {}_n N(k_1, k_2, \ldots, k_n) = \sum_{i}^{n} (-1)^i C_n^i \sum_{k_s \geqslant 0} C_{C_{n-i}^s}^{k_s} \sum_{k \geqslant 0} \prod_{j \neq s} C_{C_{n-i}^j}^{k_j} = \sum_{i}^{n} (-1)^i C_n^i 2^{C_{n-i}^s} \sum_{k \geqslant 0} \prod_{j \neq s} C_{C_{n-i}^j}^{k_j}.$$

Аналогично, вынося по одному множителю, получим

$$\sum_{k \geqslant 0} {}_n N(k_1, k_2, \ldots, k_n) = \sum_{i}^{n} (-1)^i C_n^i 2^{\sum\limits_j C_{n-i}^j} = \sum_{i}^{n} (-1)^i C_n^i 2^{2^{n-i}-1},$$

что равно числу всех покрытий множества мощности n (6).

Получим значения следующих сумм:

$$\sum_{i \geqslant 0} C_n^i \, {}_{n-i} N_{l_1 l_2 \cdots l_m}^{k_1 k_2 \cdots k_m}; \tag{31}$$

$$\sum_{n \geqslant 1} (-1)^n \, {}_n N_{l_1 l_2 \cdots l_m}^{k_1 k_2 \cdots k_m}; \tag{32}$$

$$\sum_{n \geqslant 1} \frac{(-1)^n}{n} \, {}_n N_{l_1 l_2 \cdots l_m}^{k_1 k_2 \cdots k_m}; \tag{33}$$

$$\sum_{n \geqslant 1} \frac{(-1)^n}{n(n-1)} \, {}_n N_{l_1 l_2 \cdots l_m}^{k_1 k_2 \cdots k_m}. \tag{34}$$

В выражении (20) переносом всех комбинаторных чисел ${}_n N_{l_1 l_2 \cdots l_m}^{k_1 k_2 \cdots k_m}$ из правой части получаем сумму (31):

$$\sum_{i \geqslant 0} C_n^i \, {}_{n-i} N_{l_1 l_2 \cdots l_m}^{k_1 k_2 \cdots k_m} = \prod_{i=1}^{m} C_{C_n^{l_i}}^{k_i}.$$

Для нахождения других сумм воспользуемся соотношениями (25) и (26). Подставим в (32) вместо ${}_nN_l^k$ правую часть из (25) и преобразуем выражение:

$$\sum_{n\geqslant 1}(-1)^n{}_nN_l^k=\sum_{n\geqslant 1}(-1)^n\left(\frac{1}{k}\sum_{i=0}^{l}C_{n-i}^{l-i}C_n^{n-i}N_l^{k-1}-\frac{k-1}{k}{}_nN_l^{k-1}\right)=$$
$$=\frac{1}{k}\sum_{n\geqslant 1}(-1)^n\sum_{i=0}^{l}C_{n-i}^{l-i}C_n^{n-i}N_l^{k-1}-\frac{k-1}{k}\sum_{n\geqslant 1}(-1)^n{}_nN_l^{k-1}. \tag{35}$$

В первой сумме в (3) произведем замену $n-i\to n$:

$$\sum_{n\geqslant 1}(-1)^n\sum_{i=0}^{l}C_{n-i}^{l-i}C_n^{n-i}N_l^{k-1}=\sum_{n}(-1)^n{}_nN_l^{k-1}\sum_{i=0}^{l}(-1)^iC_{n+i}^iC_n^{l-i}.$$

Воспользуемся тождествами (28):

$$\sum_{i=0}^{l}(-1)^iC_{n+i}^iC_n^{l-i}=\sum_{i=0}^{l}C_{-n-1}^iC_n^{l-i}=C_{-1}^l=(-1)^lC_l^l=(-1)^l.$$

Подставив значение первой суммы в (3), получим рекуррентное соотношение

$$\sum_{n\geqslant 1}(-1)^n{}_nN_l^k=\frac{(-1)^l-k+1}{k}\sum_{n\geqslant 1}(-1)^n{}_nN_l^{k-1}. \tag{36}$$

Для $k=1$ значение суммы (32) равно

$$\sum_{n\geqslant 1}(-1)^n{}_nN_l^1=(-1)^l{}_lN_l^1=(-1)^l.$$

Коэффициент в выражении (36) при четных и нечетных значениях l соответственно равен

$$\frac{(-1)^l-k+1}{k}=\begin{cases}-1, & \text{если } l \text{ нечетное},\\ \dfrac{2-k}{k}, & \text{если } l \text{ четное}.\end{cases}$$

При $k=2$ коэффициент для четных l равен 0. Окончательно получаем

$$\sum_{n\geqslant 1}(-1)^n{}_nN_l^k=\begin{cases}(-1)^k, & \text{если } l \text{ нечетное},\\ 1, & \text{если } l \text{ четное и } k=1,\\ 0, & \text{если } l \text{ четное и } k>1.\end{cases}$$

В общем случае сумма (32) равна

$$\sum_{n\geqslant 1}(-1)^n{}_nN_{l_1\,l_2\cdots l_m}^{k_1k_2\cdots k_m}=\begin{cases}(-1)^{\sum k_i}, & \text{если все } l_i \text{ нечетные},\\ (-1)^{\sum k_i}, & \text{если } k_i=1 \text{ для всех четных } l_i\text{; сумма идет}\\ & \text{только по } i\text{, для которых } l_i \text{ нечетно},\\ 0, & \text{если } k_i>1 \text{ хотя бы для одного четного } l_i.\end{cases}$$

Подставим ${}_nN_l^k$ из (25) в (33) и преобразуем получившееся выражение:

$$\sum_{n\geqslant 1}\frac{(-1)^n}{n}{}_nN_l^k=\sum_{n\geqslant 1}\frac{(-1)^n}{n}\left(\frac{1}{k}\sum_{i=0}^{l}C_{n-i}^{l-i}C_n^{n-i}N_l^{k-1}-\frac{k-1}{k}{}_nN_l^{k-1}\right)=$$
$$=\frac{1}{k}\sum_{n\geqslant 1}\frac{(-1)^n}{n}\sum_{i=0}^{l}C_{n-i}^{l-i}C_n^{n-i}N_l^{k-1}-\frac{k-1}{k}\sum_{n\geqslant 1}\frac{(-1)^n}{n}{}_nN_l^{k-1}.$$

Воспользуемся тождеством

$$\frac{1}{n}C_n^i = \frac{1}{n-i}C_{n-1}^i$$

и произведем замену $n - i \to n$ в первой сумме:

$$\frac{1}{k}\sum_{n\geqslant 1}\frac{(-1)^n}{n}\sum_{i=0}^{l}C_{n-i}^{l-i}C_{nn-i}^{i}N_l^{k-1} = \frac{1}{k}\sum_{n}\frac{(-1)^n}{n}\,{}_nN_l^{k-1}\sum_{i=0}^{l}(-1)^iC_n^{l-i}C_{n+i-1}^{i}.$$

Для вычисления суммы произведений биномиальных коэффициентов воспользуемся тождествами (28):

$$\sum_{i=0}^{l}(-1)^iC_n^{l-i}C_{n+i-1}^{i} = \sum_{i=0}^{l}C_n^{l-i}C_{-n}^{i} = C_0^l = 0.$$

Таким образом, получаем

$$\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_l^k = -\frac{k-1}{k}\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_l^{k-1} = \frac{(-1)^{k-1}}{k}\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_l^1. \tag{37}$$

Для $k = 1$ значение суммы (33) равно

$$\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_l^1 = \frac{(-1)^l}{l}\,{}_lN_l^1 = \frac{(-1)^l}{l}.$$

Окончательно получаем значение суммы комбинаторных коэффициентов:

$$\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_l^k = \frac{(-1)^{k+l-1}}{kl}.$$

В общем случае для $m > 1$ сумма (33) равна

$$\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_{l_1\cdots l_i\cdots l_m}^{k_1\cdots k_i\cdots k_m} = \frac{(-1)^{k_i+l_i-1}}{k_il_i}\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_{l_1\cdots l_i\cdots l_m}^{k_1\cdots 1\cdots k_m} = 0,$$

так как, согласно (37),

$$\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_{l_1\cdots l_i\cdots l_m}^{k_1\cdots 1\cdots k_m} = -\sum_{n\geqslant 1}\frac{(-1)^n}{n}\,{}_nN_{l_1\cdots l_i\cdots l_m}^{k_1\cdots 0\cdots k_m} = 0. \tag{38}$$

С помощью подобных же выкладок вычислим сумму (34):

$$\sum_{n\geqslant 1}\frac{(-1)^n}{n(n-1)}\,{}_nN_l^k = \frac{1}{k}\sum_{n\geqslant 1}\frac{(-1)^n}{n(n-1)}\sum_{i=0}^{l}C_{n-i}^{l-i}C_{nn-i}^{i}N_l^{k-1} - \frac{k-1}{k}\sum_{n\geqslant 1}\frac{(-1)^n}{n(n-1)}\,{}_nN_l^{k-1};$$

$$\frac{1}{n(n-1)}C_n^i = \frac{1}{(n-i)(n-i-1)}C_{n-2}^i;$$

$$\sum_{n\geqslant 1}\frac{(-1)^n}{n(n-1)}\sum_{i=0}^{l}C_{n-i}^{l-i}C_{nn-i}^{i}N_l^{k-1} = \sum_{n}\frac{(-1)^n}{n(n-1)}\,{}_nN_l^{k-1}\sum_{i=0}^{l}(-1)^iC_n^{l-i}C_{n+i-2}^{i};$$

$$\sum_{i=0}^{l}(-1)^i C_n^{l-i} C_{n+i-2}^{i} = \sum_{i=0}^{l} C_n^{l-i} C_{-n+1}^{i} = C_1^l = \begin{cases} 1, & \text{если } l=1, \\ 0, & \text{если } l>1. \end{cases}$$

Для $k=1$ и $l>1$

$$\sum_{n\geqslant 1} \frac{(-1)^n}{n(n-1)}\, {}_nN_l^1 = \frac{(-1)^l}{l(l-1)}\, {}_lN_l^1 = \frac{(-1)^l}{l(l-1)}.$$

Для $l=1$

$$\sum_{n\geqslant 1} \frac{(-1)^n}{n(n-1)}\, {}_nN_1^k = -\frac{k-2}{k} \sum_{n\geqslant 1} \frac{(-1)^n}{n(n-1)}\, {}_nN_1^{k-1} = \frac{(-1)^k}{k(k-1)}.$$

Окончательно:

$$\sum_{n\geqslant 1} \frac{(-1)^n}{n(n-1)}\, {}_nN_l^k = \begin{cases} \dfrac{(-1)^k}{k(k-1)}, & \text{если } l=1, \\ \dfrac{(-1)^{k+l-1}}{kl(l-1)}, & \text{если } l>1. \end{cases}$$

Используя выкладки, аналогичные (38), получим, что для $m>1$ сумма (34) равна 0:

$$\sum_{n\geqslant 1} \frac{(-1)^n}{n(n-1)}\, {}_nN_{l_1\, l_2 \cdots l_m}^{k_1 k_2 \cdots k_m} = 0.$$

4. Частные случаи

Рассмотрим частные случаи комбинаторных чисел (13), когда их коэффициенты заданы или связаны определенными соотношениями.

При $k > C_{n-1}^l$ в выражении (19) остается только одно слагаемое:

$$k > C_{n-1}^l \Rightarrow {}_nN_l^k = C_{C_n^l}^k.$$

То есть k подмножеств мощности l не могут являться покрытием ни одного собственного подмножества исходного множества (при условии вхождения этих k подмножеств в собственное подмножество). Аналогично в выражении (21): если $k_j > C_{n-1}^{l_j}$ хотя бы для одного из коэффициентов k_j, то

$${}_nN_{l_1 \cdots l_j \cdots l_m}^{k_1 \cdots k_j \cdots k_m} = \prod_{i=1}^{m} C_{C_n^{l_i}}^{k_i}.$$

Из этого тождества следует, что если в комбинаторном числе ${}_nN(k_1, k_2, \ldots, k_n)$ последний коэффициент равен 1, то

$${}_nN(k_1, k_2, \ldots, k_{n-1}, 1) = \prod_{i=1}^{n-1} C_{C_n^i}^{k_i}.$$

Просуммировав все комбинаторные числа вида ${}_nN(k_1, k_2, \ldots, k_{n-1}, 1)$, получим

$$\sum_{k\geqslant 0} {}_nN(k_1, k_2, \ldots, k_{n-1}, 1) = 2^{2^n-2},$$

что равно количеству всех покрытий множества мощности n, включающих в себя исходное множество. Вычтя это число из общего числа покрытий множества мощности n, получим выражение (7).

Используя выражение (8) и сумму комбинаторных чисел с одинаковым количеством подмножеств в покрытии, можно вывести соотношение

$$\sum_{\sum k_i=k} {}_nN_{l_1l_2\cdots l_m}^{k_1k_2\cdots k_m} = \frac{1}{2}\sum_{i=0}^{n}(-1)^iC_n^iC_{2^{n-i}-1}^k.$$

Для простого вида комбинаторных чисел при $k = C_n^l$ из (19) получаем

$$k = C_n^l \Rightarrow {}_nN_l^k = 1.$$

Если $n = \sum k_il_i$, то каждый элемент множества $\mathbf{X}$ принадлежит только одному из подмножеств покрытия. Таким образом, все покрытия представляют собой неупорядоченные разбиения множества $\mathbf{X}$ на подмножества фиксированного размера [3–5]:

$$n = \sum_i^m k_il_i \Rightarrow {}_nN_{l_1l_2\cdots l_m}^{k_1k_2\cdots k_m} = \frac{n!}{(l_1!)^{k_1}(l_2!)^{k_2}\cdots(l_m!)^{k_m}\,k_1!k_2!\cdots k_m!}. \tag{39}$$

При $m = 1$ получим

$${}_{kl}N_l^k = \frac{(kl)!}{(l!)^k k!}.$$

Количество всех неупорядоченных разбиений n-элементного множества задается числом Белла B_n [3]. Таким образом, сумма всех новых комбинаторных чисел, для которых $n = \sum k_il_i$, равна B_n. Вследствие этого получаем соотношения между различными комбинаторными числами:

$$B_n = \sum_{\sum k_il_i=n} {}_nN_{l_1l_2\cdots l_m}^{k_1k_2\cdots k_m},$$

$$B_n = n!\sum_{\sum k_il_i=n}\left((l_1!)^{k_1}(l_2!)^{k_2}\cdots(l_m!)^{k_m}\,k_1!k_2!\cdots k_m!\right)^{-1}.$$

Если $n = \sum k_il_i - 1$, то один и только один элемент множества $\mathbf{X}$ принадлежит двум подмножествам покрытия. Исключим одно из них из покрытия. Пусть мощность этого покрытия l_i. Тогда оставшееся семейство подмножеств будет являться покрытием для подмножества мощности $n - l_i + 1$, каждый элемент которого будет принадлежать только одному подмножеству покрытия. Число таких подмножеств в множестве $\mathbf{X}$ равно

$$C_n^{n-l_i+1} = C_n^{l_i-1}.$$

Любой из $n - l_i + 1$ элементов подмножества может быть в добавляемом подмножестве. Таким образом, количество вариантов построения из покрытия подмножества мощности $n - l_i + 1$ покрытия множества $\mathbf{X}$ равно

$$(n - l_i + 1)C_n^{l_i-1}\,{}_{n-l_i+1}N_{l_1\cdots l_i\cdots l_m}^{k_1\cdots k_i-1\cdots k_m}.$$

Общее количество вариантов построения покрытия множества $\mathbf{X}$ из покрытий подмножеств равно

$$\sum_{i=1}^{m}(n - l_i + 1)C_n^{l_i-1}\,{}_{n-l_i+1}N_{l_1\cdots l_i\cdots l_m}^{k_1\cdots k_i-1\cdots k_m}.$$

Так как один элемент принадлежит двум подмножествам в покрытии, то каждое покрытие посчитано два раза. Таким образом, окончательно получаем

$${}_nN_{l_1\cdots l_m}^{k_1\cdots k_m} = \frac{1}{2}\sum_{i=1}^{m}(n - l_i + 1)C_n^{l_i-1}\,{}_{n-l_i+1}N_{l_1\cdots l_i\cdots l_m}^{k_1\cdots k_i-1\cdots k_m}. \tag{40}$$

Упростим выражение (40), учитывая, что $n = \sum k_j l_j - 1$, а

$$n - l_i + 1 = \sum_{j \neq i} k_j l_j + (k_i - 1) l_i.$$

То есть для комбинаторных чисел, стоящих под знаком суммы в (40), можно применить выражение (39):

$$\begin{aligned}
{}_nN_{l_1 \cdots l_m}^{k_1 \cdots k_m} &= \sum_{i=1}^{m} \frac{(n - l_i + 1)n!}{2(l_i - 1)!(n - l_i + 1)!} \frac{(n - l_i + 1)!}{(l_1!)^{k_1} \cdots (l_i!)^{k_i - 1} \cdots (l_m!)_m^k k_1! \cdots (k_i - 1)! \cdots k_m!} = \\
&= \sum_{i=1}^{m} \frac{(n - l_i + 1)k_i l_i}{2(n+1)} \frac{(n+1)!}{(l_1!)^{k_1} \cdots (l_i!)^{k_i} \cdots (l_m!)_m^k k_1! \cdots k_i! \cdots k_m!} = \\
&= \frac{1}{2(n+1)} {}_{n+1}N_{l_1 \cdots l_m}^{k_1 \cdots k_m} \left((n+1) \sum_{i=1}^{m} k_i l_i - \sum_{i=1}^{m} k_i (l_i)^2 \right) = \\
&= {}_{n+1}N_{l_1 \cdots l_m}^{k_1 \cdots k_m} \frac{1}{2(n+1)} \left((n+1)^2 - \sum_{i=1}^{m} k_i (l_i)^2 \right).
\end{aligned}$$

Для $m = 1$ получим

$${}_{kl-1}N_l^k = \frac{l(k-1)}{2} {}_{kl}N_l^k = \frac{l(k-1)}{2} \frac{(kl)!}{(l!)^k k!}.$$

Найдем общее количество покрытий, в котором все подмножества одной мощности, то есть сумму

$$\sum_k {}_nN_l^k.$$

Подставим выражение для простых комбинаторных чисел (19) и преобразуем:

$$\sum_k {}_nN_l^k = \sum_k \sum_i C_{C_{n-i}^l}^k C_n^i (-1)^i = \sum_i C_n^i (-1)^i \sum_k C_{C_{n-i}^l}^k = \sum_i C_n^i (-1)^i \left(2^{C_{n-i}^l} - 1 \right).$$

Раскрывая скобки и учитывая, что $\sum_i C_n^i (-1)^i = 0$, окончательно получаем

$$\sum_k {}_nN_l^k = \sum_{i=0}^{n} C_n^i (-1)^i 2^{C_{n-i}^l}.$$

При $k = 2$ получаем целочисленную последовательность A006129 в OEIS [6].

5. Интерпретация

Новые комбинаторные числа, исчисляющие количество неупорядоченных покрытий конечного множества подмножествами с фиксированными мощностями, можно применять в следующих областях:

1) Конечные множества, покрытия множеств. С помощью новых комбинаторных чисел (12) и (13) можно вычислять количества покрытий конечных множеств с различными ограничениями: фиксирование мощностей подмножеств покрытия, ограничивающие соотношения для количества подмножеств в покрытии, общие условия на мощности и количество подмножеств. Вместо способа подсчета ограниченных условиями покрытий множеств, использующего метод исключения из общего числа покрытий, не удовлетворяющих заданным условиям [5], можно вычислять покрытия с помощью суммирования новых комбинаторных чисел, ограничивая параметры сумм.

2) Распределение элементов по классам. Покрытие конечного множества заданным числом подмножеств с определенными мощностями можно интерпретировать как распределение элементов множества по классам с условиями, что элемент может принадлежать нескольким классам, при этом он должен принадлежать хотя бы одному классу, а также нет идентичных друг другу классов. Новые комбинаторные числа можно применять для вычисления количества распределений по классам с заданными характеристиками (количество и мощности классов) [7, 8].
3) Диаграммы Эйлера — Венна. В качестве наглядного представления, с помощью которого можно изобразить отношения между подмножествами или классами элементов, обширно используются диаграммы Эйлера — Венна. Для вычисления вариантов распределения элементов по диаграммам могут служить новые комбинаторные числа [9].
4) Двудольные графы [10]. Покрытие конечного множества семейством несовпадающих подмножеств можно интерпретировать как частный случай двудольного графа $G(\mathbf{X} \cup \mathbf{S}, E)$, в котором каждый элемент (вершина) из $\mathbf{X}$ связан ребром хотя бы с одним элементом (вершиной) из $\mathbf{S}$ и обратно: для каждой вершины $\mathbf{S}$ есть хотя бы одно ребро, связывающее вершину из $\mathbf{X}$ с данной вершиной. Кроме того, любую пару вершин из $\mathbf{X}$ и $\mathbf{S}$ связывает только одно ребро. При этом есть условие, что для любых двух вершин из $\mathbf{S}$ наборы вершин из $\mathbf{X}$ не должны совпадать. Для подсчета таких двудольных графов с зафиксированным количеством ребер можно использовать комбинаторные числа (12) и (13).

ЛИТЕРАТУРА

1. *Comtet L.* Advanced Combinatorics. The Art of Finite and Infinate Expansions. Dordrecht, Holland: D. Reidel Publishing Company, 1974.
2. *Macula A. J.* Covers of a finite set // Mathematics Magazine. 1994. V. 67. No. 2. P. 141–144.
3. *Кнут Д., Грэхем Ф., Поташник О.* Конкретная математика. Основание информатики. М.: Мир, 2006.
4. *Эндрюс Г.* Теория разбиений. М.: Наука, 1982.
5. *Stanley R. P.* Enumerative Combinatorics. V. I. Cambridge University Press, 2002.
6. `http://oeis.org/classic/A006129` — On-Line Encyclopedia of Integer Sequences — энциклопедии целочисленных последовательностей.
7. *Холл М.* Комбинаторика. М.: Мир, 1970.
8. *Риордан Д.* Введение в комбинаторный анализ. М.: ИЛ, 1963.
9. *Burger A. P., van Vuuren J. H.* Balanced minimum covers of a finite set // Discrete Mathematics. 2007. V. 307. No. 22. P. 2853–2860.
10. *Айгнер М.* Комбинаторная теория. М.: Мир, 1982.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/10/2

УДК 004.056

АУТЕНТИФИКАЦИЯ В МНОГОУРОВНЕВЫХ СИСТЕМАХ НА ОСНОВЕ ДОВЕРЕННОЙ ПОДПИСИ[1]

П. А. Паутов

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: __Pavel__@mail.ru

Предлагаются два протокола аутентификации в многоуровневых системах на основе доверенной подписи и их реализация с помощью сертификатов. В одном протоколе доверенная подпись применяется для идентификации только клиента, а в другом — ещё и для проверки подлинности запросов внутри системы. Второй протокол обладает более сильными гарантиями безопасности, вытекающими из невозможности злоумышленником в отсутствие закрытых ключей сторон сформировать аутентичный набор запросов.

Ключевые слова: *многоуровневые системы, аутентификация, доверенная подпись.*

Введение

Рассмотрим систему, состоящую из трёх взаимодействующих подсистем: клиент, внешний сервер, внутренний сервер. Клиент взаимодействует только с внешним сервером, внешний сервер взаимодействует как с клиентом, так и с внутренним сервером (внешний сервер является клиентом внутреннего сервера). Внешний сервер взаимодействует с внутренним только для обработки запросов своих клиентов. Для взаимодействия с внутренним сервером внешний использует фиксированный набор учётных записей, т. е. одна учётная запись внутреннего сервера соответствует нескольким клиентам внешнего сервера.

Для взаимодействия с внутренним сервером используется учётная запись, соответствующая привилегиям клиента внешнего сервера. Например, на внешнем сервере клиенты делятся на группы по привилегиям: «гости», «операторы», «администраторы». Тогда для взаимодействия с внутренним сервером можно использовать три учётных записи, соответствующих группам клиентов. Если внешний сервер сам выбирает учётную запись для взаимодействия с внутренним сервером, то в случае компрометации первого злоумышленник сможет использовать учётную запись с максимальными привилегиями. Возникает задача разработки такой схемы аутентификации, при которой внешний сервер смог бы использовать для взаимодействия с внутренним сервером только ту учётную запись, которая соответствует клиенту, и только тогда, когда клиент взаимодействует с внешним сервером. Например, если клиент, обращающийся

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

к внешнему серверу, относится к группе «гости», то внешний сервер должен иметь возможность пройти аутентификацию перед внутренним сервером только от имени учётной записи «гость». Кроме того, внешний сервер не должен иметь возможности пройти аутентификацию перед внутренним сервером от имени учётной записи «гость» без помощи клиента, относящегося к группе «гости».

Искомая схема аутентификации должна удовлетворять следующим двум условиям:

- С1. При взаимодействии с клиентом внешний сервер может пройти аутентификацию перед внутренним сервером только от имени учётной записи, соответствующей данному клиенту.
- С2. Внешний сервер не может пройти аутентификацию перед внутренним сервером от имени какой-либо учётной записи без помощи клиента, соответствующего этой записи.

В работе автора [1] предложено несколько схем аутентификации на основе коммутативного шифрования, удовлетворяющих этим условиям. В данной работе предлагаются два протокола двухуровневой аутентификации на основе доверенной подписи и их реализации с помощью сертификатов. В одном протоколе (описанном в п. 1) используется доверенная подпись без полномочий и осуществляется идентификация только клиента (по его ключу), а в другом (п. 2) — применяется доверенная подпись с полномочиями и, кроме идентификации собственно клиента, производится ещё и проверка подлинности запросов, пересылаемых внешним сервером внутреннему. Второй протокол обладает более сильными гарантиями безопасности, вытекающими из невозможности злоумышленником в отсутствие закрытых ключей клиента и внутреннего сервера сформировать на внешнем сервере аутентичный набор запросов к внутреннему серверу.

1. Протокол на основе доверенной подписи

Механизм доверенных подписей (proxy signatures) позволяет одному пользователю (доверителю) делегировать возможность подписывать сообщения другому пользователю (доверенному подписчику). Доверенная подпись ставится доверенным подписчиком от имени доверителя. Концепция доверенных подписей была представлена в работе [2].

Далее будем использовать следующие обозначения: $PS_{a,b}(m)$ — доверенная подпись под сообщением m, поставленная пользователем b от имени a; C — клиент, F — внешний сервер, B — внутренний сервер; $f(C)$ — учётная запись внутреннего сервера, соответствующая клиенту C. Администратор системы делегирует каждому клиенту системы право подписи от имени соответствующей учётной записи внутреннего сервера. Для делегации права подписи необходимо выполнить некоторый протокол, который определяется выбранной схемой доверенной подписи. Он должен быть выполнен перед началом работы предлагаемого протокола аутентификации, который можно записать в виде следующей схемы, где r — случайное число:

$$\begin{array}{ll} C \to F: & C \\ F \to B: & f(C) \\ F \leftarrow B: & r \\ C \leftarrow F: & r \\ C \to F: & PS_{f(C),C}(r) \\ F \to B: & PS_{f(C),C}(r) \end{array}$$

Внутреннему серверу достаточно убедиться, что подпись поставлена от имени учётной записи $f(C)$. Идентификация доверенного подписчика (клиента) перед внутрен-

ним сервером в данном протоколе не требуется. Условия С1, С2 выполняются благодаря свойствам доверенной подписи. Корректную подпись $PS_{f(C),C}(r)$ может вычислить только клиент C, которому было делегировано право подписи от имени $f(C)$.

Схему доверенной подписи для этого протокола можно реализовать с помощью сертификата. В такой схеме доверитель подписывает сертификат, содержащий информацию, позволяющую идентифицировать доверенного подписчика (например, его открытый ключ). Сертификат отправляется доверенному подписчику. Для того чтобы сформировать доверенную подпись, доверенный подписчик подписывает сообщение своим закрытым ключом и прилагает полученный сертификат. Проверяющий проверяет подпись под сертификатом и подпись под сообщением. Следовательно, для проверки такой доверенной подписи требуется в два раза больше времени, чем при проверке одной обычной подписи. Большинство работ по доверенным подписям посвящены тому, чтобы создать схему, требующую при проверке меньше времени, чем проверка двух обычных подписей. Схемы доверенной подписи, построенные с помощью сертификатов, рассматриваются в работах [3, 4]. Такие схемы представляют интерес, так как строятся на основе проверенных технологий, их проще реализовать, используя готовые решения для обычной схемы цифровой подписи.

Для реализации предложенного протокола аутентификации на основе доверенной подписи с сертификатом будем использовать сначала вариант без полномочий, когда в сертификат входит только открытый ключ доверенного подписчика. Каждому клиенту системы и каждой учётной записи внутреннего сервера ставится в соответствие пара ключей (x, y) — соответственно закрытый (для подписания) и открытый (для проверки подписи). На внешнем сервере для каждого клиента хранится значение $(y_C, S_{x_{f(C)}}(y_C))$, где y_C — открытый ключ клиента C; S_k — некоторый алгоритм цифровой подписи на ключе k; $x_{f(C)}$ — закрытый ключ учётной записи внутреннего сервера, соответствующей клиенту C. Протокол аутентификации в этом случае записывается как

$$\begin{aligned} C \to F:&\quad C \\ F \to B:&\quad f(C) \\ F \leftarrow B:&\quad r \\ C \leftarrow F:&\quad r \\ C \to F:&\quad S_{x_C}(r) \\ F \to B:&\quad S_{x_C}(r), y_C, S_{x_{f(C)}}(y_C) \end{aligned}$$

В конце протокола внутренний сервер проверяет подпись $S_{x_C}(r)$ с помощью ключа y_C и подпись $S_{x_{f(C)}}(y_C)$ с помощью ключа $y_{f(C)}$, который хранится на внутреннем сервере. Это обеспечивает выполнение условий С1, С2.

2. Аутентификация с проверкой подлинности запросов

Описанные выше протоколы обеспечивают аутентификацию клиента, и они затрагивают только начало сеанса работы клиента с системой. Если злоумышленник скомпрометирует внешний сервер (получит над ним полный контроль), то после аутентификации клиента злоумышленник сможет подменять запросы, отправляемые на внутренний сервер в рамках данного сеанса работы. Для предотвращения этой угрозы нужно сопоставить запросы, отправляемые клиентом, с запросами, отправляемыми внешним сервером на внутренний, таким образом, чтобы злоумышленник не смог отправить на внутренний сервер запрос, не соответствующий клиентскому.

Будем считать, что взаимодействие между клиентом и внешним сервером, внешним сервером и внутренним происходит с помощью запросов вида $(req, (a_1, ..., a_n))$,

состоящих из имени req и набора аргументов $(a_1, ..., a_n)$. В ответ на любой такой запрос клиента внешний сервер посылает один или несколько запросов внутреннему серверу. Для каждого запроса клиента набор отправляемых на внутренний сервер запросов фиксирован, и в каждом запросе для внутреннего сервера используется подмножество параметров запроса клиента. Иначе говоря, для каждого запроса $Freq$ с n параметрами, посылаемого на внешний сервер, можно записать набор $((Breq_1, (i_1, ..., i_{k_1})), ..., (Breq_m, (i_1, ..., i_{k_m})))$, где $Breq_j$ — запросы, посылаемые на внутренний сервер, а i_j — номера параметров запроса $Freq$. Далее данный набор будем обозначать $g(Freq)$. В рамках такой модели можно сформулировать ещё одно требование безопасности:

С3. При обработке запроса клиента внешний сервер может использовать только определённые для него запросы внутреннего сервера с параметрами, переданными клиентом.

Данное требование можно выполнить, применив схему доверенной подписи с полномочиями. В таких схемах доверенный подписчик может поставить подпись от имени доверителя только под сообщением, которое соответствует полномочиям, делегированным ему доверителем. Доверенную подпись под сообщением m, поставленную пользователем b от имени a в соответствии с набором полномочий w, будем обозначать как $PS_{a,b,w}(m)$.

При добавлении нового клиента системы C администратор должен делегировать ему право подписи от имени $f(C)$ для каждого запроса $Freq$, посылаемого клиентом, использовав пару $(Freq, g(Freq))$ в качестве набора полномочий.

Для краткости изложения протокола будем предполагать, что по любой из подписей $PS_{a,b,w}(m)$ и $S_x(m)$ можно восстановить сообщение m. Тогда работа системы происходит по следующему протоколу, поясняемому ниже:

$$\begin{aligned}
C \to F: &\quad C, PS_{f(C),C,(Freq,g(Freq))}(Freq, A_{1,n}, t) \\
F \to B: &\quad f(C), (Breq_1, A_{1,k_1}), PS_{f(C),C,(Freq,g(Freq))}(Freq, A_{1,n}, t) \\
&\quad \dots \\
F \to B: &\quad f(C), (Breq_m, A_{1,k_m}), PS_{f(C),C,(Freq,g(Freq))}(Freq, A_{1,n}, t)
\end{aligned}$$

В нём клиент посылает внешнему серверу подписанный запрос с набором аргументов $A_{1,n}$ и временно́й меткой t. Внешний сервер посылает серию соответствующих запросов внутреннему. Значения, возвращаемые в качестве результата исполнения запросов, опущены для краткости. Внутренний сервер при обработке каждого запроса проверяет полученную подпись, а также соответствие запроса $(Breq_i, A_{1,k_i})$ описанию $g(Freq)$ и исходному запросу $(Freq, A_{1,n})$. Вместо временной метки можно использовать случайные числа, генерируемые внутренним сервером.

Доверенная подпись с полномочиями может быть реализована посредством сертификатов следующим образом: $PS_{a,b,w}(m) = (S_{x_b}(m), y_b, w, S_{x_a}(y_b, w))$. Рассмотрим протокол аутентификации, использующий данную реализацию. Каждому клиенту системы и каждой учётной записи внутреннего сервера, как обычно, ставится в соответствие пара ключей (x, y) (закрытый и открытый). На внешнем сервере для каждого запроса $Freq$, посылаемого клиентом C на внешний сервер, хранится сертификат $S_{x_{f(C)}}(y_C, Freq, g(Freq))$. Аутентификация в системе происходит по следующему протоколу:

$$
\begin{aligned}
C \to F: &\quad C, S_{x_C}(Freq, A_{1,n}, t)\\
F \to B: &\quad f(C), (Breq_1, A_{1,k_1}), S_{x_C}(Freq, A_{1,n}, t), S_{x_{f(C)}}(y_C, Freq, g(Freq))\\
&\quad \dots\\
F \to B: &\quad f(C), (Breq_m, A_{1,k_m}), S_{x_C}(Freq, A_{1,n}, t), S_{x_{f(C)}}(y_C, Freq, g(Freq))
\end{aligned}
$$

В данной реализации клиент подписывает свои запросы, используя лишь свой закрытый ключ, в то время как в предыдущем варианте (без сертификата) каждый запрос должен быть подписан с использованием соответствующего набора полномочий.

Заключение

В работе предложены протоколы аутентификации в многоуровневой системе, ограничивающие возможности злоумышленника, получившего контроль над внешним сервером системы. Аутентификация в них основывается на доверенной подписи. Использование последней с полномочиями позволяет не только идентифицировать клиента, но и проверить подлинность запросов от его имени с внешнего сервера к внутреннему, что усиливает защиту системы от злоумышленника, скомпрометировавшего внешний сервер. Реализация же доверенной подписи с помощью сертификатов позволяет реализовать протоколы аутентификации исключительно на базе алгоритмов обычной цифровой подписи, что ввиду развитости инфраструктуры цифровых подписей в мире делает предложенные протоколы аутентификации совершенно практическими.

ЛИТЕРАТУРА

1. *Паутов П. А.* Аутентификация в модели доверенной подсистемы на основе коммутативного шифрования // Прикладная дискретная математика. 2010. № 3. С. 90–95.
2. *Mambo M. M., Usuda K., Okamoto E.* Proxy Signatures: Delegation of the Power to Sign Message // IEICE Transaction Fundamentals. 1996. V. E79-A. No. 9. P. 1338–1353.
3. *Boldyreva A., Palacio A., Warinschi B.* Secure Proxy Signature Schemes for Delegation of Signing Rights. 2003. `http://eprint.iacr.org/2003/096.pdf`
4. *Tan Z., Liu Z.* Provably Secure Delegation-by-Certification Proxy Signature Schemes // InfoSecu'04. Proceedings of the 3rd International Conference on Information security. 2004. P. 38–43.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

DOI 10.17223/20710410/10/3

УДК 681.3

МОДЕЛЬ ЗАЩИЩЁННОЙ АРХИТЕКТУРЫ ЭВМ И ЕЁ ВИРТУАЛЬНАЯ РЕАЛИЗАЦИЯ[1]

В. В. Горелов

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: skylark@mail.tsu.ru

Предлагается модель защищённой вычислительной архитектуры со сквозной защитой типов данных и ссылок. Сквозная защита достигается за счёт контроля границ значений типов данных и ссылок и допустимых операций для них. Такой подход делает возможной единообразную защиту ресурсов на всех уровнях: машинных инструкций, прикладных программ, операционной системы. Текущая реализация данной архитектуры произведена через разработку виртуальной машины, работающей поверх существующей операционной системы общего назначения.

Ключевые слова: *программное обеспечение, вычислительная архитектура, безопасность, виртуализация, изоляция ошибок.*

Введение

На протяжении последних десятилетий развитие массовых электронных вычислительных систем общего назначения происходит скачкообразными темпами. Если аппаратная составляющая развивается в основном экстенсивным методом (увеличение объёма оперативной и постоянной памяти, разрядности процессора, частоты и количества ядер), то архитектурная составляющая меняется значительно медленнее. Пожалуй, единственным серьёзным архитектурным изменением, с точки зрения безопасности, в архитектуре процессоров (Intel-совместимых и с аналогичной архитектурой) явилось использование защищённого режима процессора. В данном режиме адресное пространство процесса и ядра ОС защищено от случайного ошибочного или умышленного злонамеренного поведения другого процесса. Компоненты вычислительной системы могут находиться в защищённых друг от друга областях памяти. Однако на этом изоляция адресных пространств заканчивается. Внутри этих разделённых областей никакой защиты нет. Ни ядро операционной системы, ни прикладные программы не защищены внутри себя от непреднамеренных ошибок или умышленного «склонения» поведения кода в нужную злоумышленнику сторону.

С целью повышения надёжности и безопасности разработчики современных языков программирования (Ада, Ява, Си#) предусмотрели ряд мер, предотвращающих ненадёжное и небезопасное поведение программ, однако все они направлены исключительно на программы, написанные на этих языках, а не на вычислительную архитектуру

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

в целом. Кроме того, для обеспечения проверки целостности типов и указателей в код внедряются программные проверки разнообразных границ, что неизбежным образом замедляет работу программ. В двух последних языках программирования обязательным компонентом является сборщик мусора, лишающий программу важной характеристики: работа в реальном времени. Данный режим работы гарантирует максимальное время реакции компонентов вычислительной системы на определённое событие, что при работе сборщика мусора не представляется возможным. Последние два языка также изначально спроектированы как языки прикладные, т. е. написание на них кода ядра операционной системы крайне затруднительно, если вообще возможно.

Язык программирования существующих широко распространённых ядер — Си. Если часть ядра, которая не зависит от внешних устройств, относительно проработана, то большое количество плат и устройств расширений ЭВМ требует драйверов, которые динамически подключаются в адресное пространство ядра для обеспечения связи между ядром и «железом». Драйверы зачастую разрабатываются сторонними (не основными) разработчиками ядра. Это влияет как на количество разнообразных ошибок, так и на сам формат драйвера, которым они снабжают пользователей своих устройств. Часть драйверов поставляются без исходных текстов, т. е. в машинном виде, что делает их анализ, нахождение и устранение ошибок максимально трудоёмким. Самостоятельная разработка драйверов на системном языке Си не всегда возможна ввиду того, что производитель не раскрывает спецификацию протокола взаимодействия со своим устройством. В частности, производители видеокарт не открывают исходные коды своих драйверов ещё и по причине наличия в них сомнительных приёмов для:

— подгонки производительности под конкретное ПО (для тестов производительности) с целью получения видимости конкурентного преимущества перед аналогичными показателями тестов видеокарт других производителей;
— трюков, позволяющих повысить производительность за счёт качества изображения;
— сокрытия механизмов постепенного ухудшения производительности и функциональности «старых» видеокарт.

Подключение таких закрытых[2] производителем модулей в настоящее время почти всегда является единственной возможностью использовать устройство, но в то же время позволяет работать на уровне ядра ошибочным и потенциально злонамеренным драйверам. Недоступность исходного кода драйверов, кроме вышеописанного, влечёт за собой затруднённое обнаружение не только ошибок внутри самого драйвера, но и его пагубного воздействия на другие компоненты ядра.

Текущее положение вещей представляется весьма неблагоприятным для надёжности и безопасности как отдельных компонентов вычислительной системы, так и для всей системы в целом. Далее предлагается экспериментальная модель архитектуры (называемая «композит») со сквозной защитой данных и ссылок, достигаемой путём контроля границ значений типов данных и ссылок, а также допустимых операций для них. Тот факт, что такой контроль заложен непосредственно в архитектуре, даёт возможность работы всей вычислительной системы с высокой степенью изоляции от внешних и внутренних ошибок на всех уровнях: машинные коды, ядро операционной системы, прикладные программы, т. е. защита распростаняется на все компоненты: от самого низкого уровня «доступа к железу» до уровня прикладных программ. Це-

[2]Продолжая Л. Н. Гумилёва: «А там, где тайна, там дьявол, ложь, гордыня и злоба».

лью архитектуры является выявление и, как следствие, ликвидация широкого класса дефектов в программном обеспечении.

Статья состоит из двух частей. В первой части (п. 1 – 6) описана модель защищённой архитектуры ЭВМ. Представлены типы данных архитектуры, операции над ними, вычислительные модули, смена контекста выполнения вычислительных модулей и поддержка исключений. Во второй части (п. 7 – 11) описаны разные подходы виртуализации, процессы загрузки и выполнения ОС для защищённой архитектуры, а также текущее состояние реализации и перспективы разработки.

1. Типы данных архитектуры

Типы данных делятся на два принципиально различных класса:

1) непосредственно данные;
2) ссылки на данные.

Первые представляют собой величины для хранения и вычисления целых и вещественных значений. Размерность этих базовых значений может быть 1, 2, 4 или 8 байт для целых и 4 или 8 байт для вещественных типов.

1.1. Целые типы данных архитектуры

Целые делятся на два класса:

1) с представлением неотрицательных чисел;
2) с представлением отрицательных и неотрицательных чисел.

Допустимый диапазон значений для первого класса $[0; 2^n - 1]$; для второго — $[-2^{n-1}; 2^{n-1} - 1]$, где n — разрядность в битах.

В свою очередь, каждый из классов целых чисел разделён на два других: тип по модулю и без модуля. Непосредственно поддерживаются следующие типы переменных: `uchar`, `ushort`, `uint`, `schar`, `sshort`, `sint`, `mod8`, `mod16`, `mod32`. Первые три — беззнаковые типы с размерностью 8, 16, 32 бит соответственно; за ними следуют знаковые типы тех же размерностей и закрывают перечисление модульные типы.

Для типов по модулю операции над ними проводятся по модулю 2^8, 2^{16} или 2^{32} соответственно. Для безмодульных типов операции, приводящие к результату, лежащему за пределами допустимого диапазона, вызывают диагностическое прерывание.

1.2. Подтипы

Допускается введение подтипов для базовых типов данных. Подтипы — новые типы с суженными для них границами диапазонов допустимых значений. В новом подтипе допустимы такие же операции, как и для родительского типа, но с учётом ограничений подтипа.

В операциях присваивания родительскому типу значения его подтипа не требуется указание типа, так как заранее известно, что его диапазон шире, чем у подтипа. Обратное не верно, операция указания типа необходима, и в случае нарушения допустимого диапазона подтипа вызывается диагностическое прерывание.

1.3. Теги

Каждая переменная, кроме типа данных и самих данных, снабжена тегом, указывающим допустимые для неё операции — чтение и/или запись. Поскольку теги и другие вспомогательные информационные блоки необходимы только для архитектуры, то видны они только «процессору» архитектуры, оставаясь недоступными на уровне кода программы. Платой за это является бóльший расход физической памяти.

2. Операции

Представленные здесь операции являются лишь минимальной основой для работы разработанного прототипа [1] виртуальной машины, реализующей архитектуру. С развитием архитектуры и более широком её использовании на практике их множество, очевидно, будет расширяться. Коды основных операций следующие:

— `op_call`: вызов функции;
— `op_jmp`: переход к именованной метке;
— `op_ret`: возврат из функции;
— `op_nop`: пустая операция;
— `op_mov`: операция пересылки;
— `op_add`: операция сложения;
— `op_sub`: операция вычитания;
— `op_mul`: операция умножения;
— `op_div`: операция деления;
— `op_out`: операция печати (на «принтер» — фактически байты выводятся на стандартный поток вывода эмулятора).

Каждая операция состоит из:

— номера строки (необходим при выводе диагностической информации);
— метки (необязательная метка для перехода);
— кода операции;
— списка аргументов.

3. Вычислительный модуль

Этот модуль, называемый иногда сокращённо ВМ, представляет собой базовый блок архитектуры, защищённый от недружественного воздействия извне. Он имеет доступ непосредственно к переменным и ссылкам, переданным ему как параметры, и больше ни к чему доступа не имеет. Создать произвольные ссылки, указывающие на не принадлежащие ему величины, он также не может.

Описательная часть модуля состоит из:

— имени;
— списка входных типов и их символьного обозначения;
— списка собственных переменных, доступных для оперирования;
— списка собственных массивов переменных;
— списка операций.

4. Текущий контекст

Текущий контекст задаёт состояние вычислительного модуля с переданными ему аргументами, с его собственными аргументами и с текущим курсором порядкового номера исполняемой операции. Номер используется исключительно процессором и не доступен для выполняющегося кода.

5. Поддержка исключений

Архитектура предусматривает поддержку исключений как один из базовых механизмов по обнаружению и реагированию на внештатные ситуации. В рамках вычислительного модуля предусматривается возможность создания блоков обработки исключительных ситуаций. Для обработки исключительной ситуации в конце тела модуля создаются метки вида `:when_overflow`, `:when_outofbounds`, `:when_outofmemory`,

`:when_ioerror` и т. д., в рамках которых алгоритм может должным образом среагировать на возникшую ситуацию и при возможности продолжить работу. В случае отсутствия необходимого обработчика исключения в текущем модуле контекст закрывается и выполняется операция сворачивания контекстов и поиска обработчика. Операция поиска продолжается до тех пор, пока «размотка» контекстов не доберётся до обработчика исключений по умолчанию, который закроет экземпляр ОС. Вся информация об исключении сохраняется для последующего анализа.

Ввиду поддержки исключений представляется возможным исполнение программ, написанных как на низкоуровневых языках программирования (Си и т. п.), так и на высокоуровневых (Ада, Ява, Си# и т. п.).

6. Поддержка языков программирования

Для поддержки существующего программного обеспечения необходима разработка или, скорее всего, адаптация существующих компиляторов путём добавления в них возможности создавать код для защищённой архитектуры. Для достаточно низкоуровневого языка программирования Си (и Си++, как основанного на нём) ожидаемой будет ситуация, подобная следующей. Предположим, что где-то в исходном коде есть некая переменная i типа `unsigned int`, которая изменяется во время исполнения программы. Если значение переменной максимально $(2^{32} - 1)$ и к нему прибавляется единица, возможны два случая.

— Счётчик переполнен. В этом случае далее алгоритм будет работать неверно, так как значение счётчика станет равным 0 и выражение $(i + 1 > i)$ будет ложно. Поскольку с точки зрения процессора архитектуры произошло нарушение границ типов, задача будет остановлена с протоколированием стека для последующего разбирательства.

— Возможен и другой случай, когда по логике алгоритма так задумано умышленно: прибавление 1 к максимальному допустимому значению должно выполняться по модулю 2 в степени (размер счётчика в битах), и аварийный останов программы недопустим. Автоматически отличить этот случай от первого не представляется возможным, поэтому для формального их разделения необходима более точная конкретизация языка программирования, например через введение в него типа `mod int`.

При применении адаптированного компилятора для платформы будут получены два очевидных плюса:

— автоматическое обнаружение ошибочных и потенциально ошибочных мест в программах;

— возможность запуска на данной архитектуре существующего программного обеспечения.

7. Системы виртуализации

7.1. Типы виртуализации

Современные системы виртуализации весьма обширны по своей природе и чаще всего бывают следующих типов:

1) Полная виртуализация архитектуры. Для запускаемой ОС создаётся впечатление, что она работает поверх реального «железа», и для ОС зачастую нет никакой возможности понять, что она виртуализирована. Данный тип подходит для виртуализации практически всего, что работает на эмуляторе соответствующей

архитектуры. Обычно различают архитектуру машины-хозяина (Host) и архитектуру машины-гостя (Guest). Под машиной-хозяином имеется в виду главная (физическая) машина, на которой запущено средство виртуализации, а под гостевой — система, запущенная внутри средства виртуализации. Их архитектуры могут не совпадать. Например, в эмуляторе, работающем в ОС на архитектуре x86, можно запустить ОС для архитектуры PowerPC и наоборот. Типичным представителем данного типа виртуализации является проект QEMU [2]. Этот тип виртуализации низкоскоростной, однако существуют различные эффективные оптимизации, значительно ускоряющие процесс. Проникновение злоумышленного кода из гостевой системы в систему хозяина маловероятно.

2) Частичная виртуализация. В этом случае, как правило, архитектуры гостевой и хозяйской систем совпадают. Часть гостевых команд запускается прямо на процессоре хозяина, другая часть (привилегированные команды) динамически преобразуется в этот тип. Хорошо подходит для виртуализации произвольных ОС на данной архитектуре. Отличается довольно высокой скоростью работы. Типичные представители этого типа — VMWare [3], VirtualBox [4]. Проникновение злоумышленного кода из гостевой системы в систему хозяина маловероятно.
3) Паравиртуализация. Ядро гостевой системы должно быть определённым образом подготовлено для такой работы. Подготовка несколько напоминает перенос на другую платформу. Заключается это в перенастройке ядра на использование специализированных обращений к гипервизору Xen для работы в привилегированном состоянии процессора, для захвата и управления физической памятью, устройствами ввода-вывода. Этот тип отлично подходит для ядер с открытым кодом. Данная технология успешно применена для Linux [5], FreeBSD [6]. Для закрытых систем требуется модификация с разрешения компании-владельца. Запуск таких систем возможен и без их модификации, однако в данном случае выигрыш в быстродействии не наблюдается. Типичным представителем этого направления является технология Xen [7]. Паравиртуализация отличается ещё бóльшим быстродействием. Проникновение злоумышленного кода из гостевой системы в систему хозяина зависит от многочисленных факторов.
4) Виртуализация уровня ОС. В данном подходе изменяется ОС машины-хозяина, чтобы создать внутри неё изолированное окружение. В этом случае скорость работы программ ещё выше, поскольку инструкции работают напрямую — поверх самой ОС. Архитектуры гостя и хозяина совпадают. Защита осуществляется силами самой ОС. Взлом таких систем наиболее вероятен. Типичные представители этого: Solaris Zones [8], FreeBSD jail [9], OpenVZ [10], chroot [11].

7.2. Взаимовлияние хозяина и гостевых систем

Общим свойством описанных выше систем виртуализации является тот факт, что для полноценной работы гостевой системы в неё необходимо добавление драйверов или других изменений ядра, реализующих каналы информационного обмена с хозяином. С точки зрения производительности это безусловный плюс. С точки зрения безопасности появляется возможность совершить поднятие полномочий с гостевых до уровня хозяина. Кроме этого, если гостевые ОС более-менее изолированы друг от друга, то главная ОС имеет полный доступ к любой из гостевых систем.

В разрабатываемой архитектуре изначально предполагается сделать реальным и естественным безопасное и эффективное сосуществование разных ОС для данной архитектуры. Механизмом для этого служит системный монитор. В его основные задачи

входит запуск, остановка и переключение между запущенными ОС. Таким образом, все запускаемые ОС изолированы друг от друга и над ними не довлеет хозяйская система.

7.3. Дополнительные плюсы виртуализации

Кроме прямых плюсов виртуализации — возможности запускать несколько ОС на одной физической ЭВМ, у этой технологии есть и относительно недавно открытая возможность миграции гостевых систем между системами хозяев. Это находит широкое применение при обновлении аппаратного обеспечения, когда в режиме реального времени практически без прерывания работы гостевая ОС «переезжает» на новую машину. Кроме этого, такой механизм зачастую применяется при перераспределении нагрузки в вычислительных центрах и для повышения отказоустойчивости систем. Такие технологии входят в так называемые облачные вычисления на стороне вычислительных центров.

8. Виртуальная ОС защищённой архитектуры

Она состоит из набора вычислительных модулей, векторов для оперирования с контекстами вычислительных модулей, индекса текущего контекста.

Экспериментальный вариант виртуальной машины, реализующей защищённую архитектуру, реализован для загрузки мнемонического представления программ на этапе загрузки. Далее он преобразуется во внутренние структуры процессора. Часть проверок, которые можно отследить при трансляции кода, реализована на этапе загрузки. В аппаратной же реализации проверки при трансляции следует отдать транслятору, а проверки на этапе выполнения — процессору. Ввиду программной реализации архитектуры (в виде эмулятора), все проверки будут описаны без явного разделения на фазу трансляции и фазу проверки во время исполнения кода.

9. Процесс загрузки экземпляра (виртуальной) ОС

ОС представляет собой набор вычислительных модулей. Её загрузка состоит из трёх фаз. До выполнения первой создаётся пустая запись, идентифицирующая очередной загружаемый вычислительный модуль.

9.1. Фаза 1. Загрузка описания (заголовка) вычислительного модуля

В данной фазе производится проверка синтаксиса и семантики описания вычислительного модуля. Каждый модуль должен начинаться с ключевого слова `.name` с указанием имени модуля и следующим за ним списком пар вида (тип, имя). В семантическую составляющую входит проверка существования модуля с таким же именем, корректности названий типов, а также уникальности названий переменных в рамках данного вычислительного модуля. Разрешённые операции для аргументов устанавливаются истинными для записи и чтения, если не определено обратное. Если проверка прошла успешно, то в запись о ВМ вносятся соответствующие поля и осуществляется переход к следующей фазе. В случае нарушения правил синтаксиса или семантики выдаётся сообщение об ошибке и номер её строки.

9.2. Фаза 2. Загрузка локальных переменных вычислительного модуля

В данной фазе также проводится проверка синтаксических и семантических конструкций. Сигнальным элементом начала определения блока переменных является

ключевое слово `.var`. Для определения одной переменной построчно должны быть считаны пары (тип, имя).

Синтаксически верными именами переменных являются имена, начинающиеся с буквы, далее может располагаться произвольная комбинация цифр, букв и символов подчёркивания.

Для неинициализированных переменных допустимость операции чтения устанавливается как ложь, операции записи — как истина. Это не касается случая, когда задаётся константа через элемент `const`, который следует после описания типа. В этом случае допустимость чтения — истина, записи — ложь.

Кроме одиночных переменных, существует понятие массива — адресуемого множества с переменными одного типа. В блоке описания переменных `.var` массивы задаются следующим образом:

```
array TYPE NAME SIZE VALUE
```

или как

```
array TYPE NAME SIZE VALUE1 VALUE2 ...
```

В первом случае ключевое слово `array` указывает, что описывается массив; затем идёт его тип, размер и значение, которым он будет инициализирован. Во втором случае значения `VALUE1`, `VALUE2`, ... будут циклически записаны в элементы массива. Значения для инициализации массива проверяются на границы допустимости для типа массива. Неотъемлемым атрибутом массива является его размер. Все операции по индексированию проверяются на соблюдение границ.

Для задания строк допустима такая форма:

```
array mod8 My_String "Здравствуй, мир!".
```

Данная форма — аналог байтного массива. Размер массива определяется на этапе трансляции. В текущей реализации для простоты все символы кодируются однобайтной кодировкой KOI8-R. Такая форма задания строк допускается как для удобства программиста, так и для повышения читаемости кода.

Так же, как и в предыдущей фазе, любое нарушение синтаксиса или семантики выводится как ошибка с пояснением и номером строки.

9.3. Фаза 3. Загрузка кодов операций вычислительного модуля

Ключевым словом к разбору тела модуля является `.begin`. В данной фазе, как и в двух предыдущих, проводится синтаксическая проверка соответствия названий имён переменных, операций и др. Семантическая проверка сверяет область видимости вызываемых функций, совместимость типов и их количество, передаваемых при вызове других модулей. Все операции, прошедшие проверку, записываются в кортеж операций текущего вычислительного модуля с заполнением всех полей. Для модулей не допускается отсутствие операций. Сигналом окончания текущего модуля является ключевое слово `.end`. После этого текущий вычислительный модуль помещается в кортеж вычислительных модулей для текущей ОС. На этом фаза загрузки вычислительных модулей завершена.

9.4. Ссылки

Ссылку можно создать специальной командой, аргументом которой является имя переменной или массива, на который она должна указывать. При создании ссылки на массив необязательным, но полезным является параметр, конкретизирующий границы, доступные для индексации через создаваемую ссылку. Границей служит как нижний индексируемый диапазон массива, так и верхний. Часто бывает полезно умышлен-

но «заузить» адресуемый диапазон, чтобы предоставить доступ только к необходимому подмножеству значений.

10. Процесс выполнения экземпляра (виртуальной) ОС

На начальном этапе создаётся пустой контекст, в который помещается контекст главного вычислительного модуля. По традиции главным модулем должен быть модуль с именем `main`, хотя это вопрос реализации, и первым запускаемым модулем может быть просто первый модуль. Курсор текущей операции устанавливается на начало и запускается выполнение команд из текущего контекста.

10.1. Выполнение операций в контексте (текущего) вычислительного модуля

Берётся очередная операция, и в зависимости от её кода выполняются действия. Рассмотрим варианты работы для ключевых операций.

`op_call`: вызов другого вычислительного модуля. Для нового ВМ создаётся новый контекст, в который помещается отображение вызываемой функции с установкой курсора на начало операций. Кроме этого, передаются параметры в контекст вызываемого модуля. Типы и совпадение количества аргументов уже проверены на этапе загрузки кода, так что остаётся только проверить правомочность доступа и скопировать значения передаваемых аргументов. Далее осуществляются запись нового контекста во внутренний стек процессора, сдвиг курсора контекстов на единицу и переход к выполнению вновь созданного контекста. По сути, здесь идёт рекурсивный вызов, однако для нового контекста, который является подмножеством родительского.

`op_jmp`: переход к именованной метке внутри текущего вычислительного модуля. Смены контекста не происходит, следующая выполняемая операция — операция, помеченная соответствующей меткой.

`op_ret`: окончание выполнения модуля и возвращение в предыдущий контекст. Если курсор контекстов установлен на начальное положение, то происходит завершение работы данной виртуальной ОС. В противном случае уничтожается текущий контекст и сдвигается курсор на начало. Далее следует выход из рекурсии к предыдущему контексту.

`op_XXX`: какая-то операция, которую выполняет процессор. В процессе её выполнения процессор проверяет необходимые условия и ограничения. Если эта операция не последняя и не операция перехода или вызова другого ВМ, то происходит переход к выполнению следующей операции. Если эта операция последняя в ВМ, то её действие аналогично операции `op_ret`.

11. Текущее состояние и перспективы разработки

На данный момент в «композите» реализована существенная часть безопасной архитектуры. Все исходные материалы проекта доступны на странице проекта [1]. Поддерживается запуск небольших программ, успешное выполнение всех существующих на данный момент регрессионных тестов. Планируется проверить на практике работу всей системы с привлечением широкого спектра существующего прикладного обеспечения на высокоуровневых языках программирования. Для этого потребуется адаптировать существующие или разработать новые компиляторы для перевода программ с языков высокого уровня на уровень кодов архитектуры. Применение компиляторов для существующих программ, их обкатка и успешная работа в рамках предложенной архитектуры может дать толчок для реализации архитектуры в аппаратуре. В этом случае возможен отказ от использования текущей операционной системы, служащей

для запуска виртуальной машины. Кроме этого, производительность системы в целом должна существенно возрасти за счёт реализации примитивов защиты на аппаратном уровне. В остальном все положительные черты безопасной архитектуры сохраняются.

ЛИТЕРАТУРА

1. `http://skylark.tsu.ru/compozit/` — Домашняя страница экспериментальной реализации проекта Compozit. 2010.
2. `http://www.qemu.org/` — Домашняя страница проекта QEMU. 2010.
3. `http://www.vmware.com/` — Домашняя страница проекта VMWare. 2010.
4. `http://www.virtualbox.org/` — Домашняя страница проекта VirtualBox. 2010.
5. `http://www.linux.org/` — Домашняя страница проекта Linux. 2010.
6. `http://www.freebsd.org/` — Домашняя страница проекта FreeBSD. 2010.
7. `http://www.xen.org/` — Домашняя страница проекта Xen. 2010.
8. `http://www.sun.com/bigadmin/content/zones/` — Домашняя страница проекта Solaris Zones. 2010.
9. `http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/jails.html` — Документация про технологию FreeBSD jail. 2010.
10. `http://wiki.openvz.org/` — Домашняя страница проекта OpenVZ. 2010.
11. `http://www.freebsd.org/cgi/man.cgi?query=chroot&apropos=0&sektion=0&manpath=FreeBSD+8.1-RELEASE&format=html` — Руковоство по функции `chroot()` из проекта FreeBSD. 2010.

2010 Математические основы информатики и программирования №4(10)

DOI 10.17223/20710410/10/4

УДК 004.032.26(06)

О ВЛОЖЕНИИ ГРАФОВ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ В ГРАФЫ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ РЕКУРРЕНТНЫМИ НЕЙРОННЫМИ СЕТЯМИ

М. С. Тарков

Институт физики полупроводников им. А. В. Ржанова СО РАН, г. Новосибирск, Россия

E-mail: tarkov@isp.nsc.ru

Сформулирована задача вложения графов параллельных программ в графы распределённых вычислительных систем рекуррентными нейронными сетями. Экспериментально получены значения параметров, обеспечивающие отсутствие некорректных решений. Благодаря введению в функцию Ляпунова коэффициента штрафа для рёбер графа программы, не совпадающих с рёбрами графа ВС, при вложении «линейки» в двумерный тор получены оптимальные решения. Для увеличения вероятности оптимальных вложений предложен метод расщепления вложения, суть которого заключается в приведении матрицы решения к блочно-диагональному виду. Для исключения некорректных решений при вложении линейки в трехмерный тор использована рекуррентная сеть Вана, обладающая более быстрой сходимостью, чем сеть Хопфилда.

Ключевые слова: *вложение, графы параллельных программ, распределенные вычислительные системы, нейрон, сеть Хопфилда, рекуррентная сеть Вана.*

Введение

Распределённая вычислительная система (ВС) [1 – 5] представляет собой множество элементарных машин (ЭМ), связанных сетью, программно управляемой из этих машин. Каждая элементарная машина (ЭМ) включает вычислительный модуль (ВМ) и системное устройство (СУ) (маршрутизатор сообщений). Системное устройство функционирует под управлением ВМ и имеет входные и выходные полюсы, связанные соответственно с выходными и входными полюсами соседних ЭМ. Структура ВС описывается графом $G_s(V_s, E_s)$, где V_s — множество ЭМ и E_s — множество связей между ЭМ. Для максимальной эффективности информационных взаимодействий современные высокопроизводительные ВС используют регулярные графы $G_s(V_s, E_s)$ межмашинных соединений («гиперкуб», «двумерный тор», «трёхмерный тор») [1 – 5].

Гиперкубическая структура описывается графом, известным как m-мерный булевский куб с числом вершин $n = 2^m$. Тороидальные структуры представляют собой m-мерные евклидовы решетки с замкнутыми границами. Группа автоморфизмов E_m такой структуры есть прямое произведение циклических подгрупп C_{N_k}: $E_m = \bigotimes_{k=1}^{m} C_{N_k}$, где N_k — порядок подгруппы C_{N_k}; $\bigotimes$ — символ прямого произведения. При $m = 2$ получаем двумерный тор (2D-тор) (рис. 1), при $m = 3$ — 3D-тор.

Для распределённых ВС граф $G_p(V_p, E_p)$ параллельной программы обычно определяется как множество V_p ветвей программы (виртуальных элементарных машин), которые взаимодействуют друг с другом по принципу «точка — точка» посредством передачи сообщений по логическим (виртуальным) каналам (одно- и двунаправленным) множества $E_p \subseteq V_p \times V_p$. Для большинства параллельных прикладных программ харак-

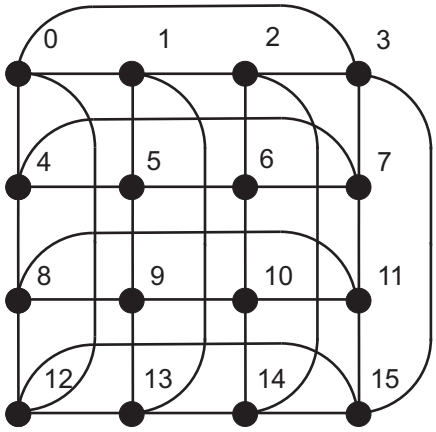

Рис. 1. Пример 2D-тора

терны упорядоченные во времени и регулярные в пространстве схемы взаимодействий между обрабатывающими модулями (линейка, кольцо, решётка и др.) (рис. 2) [1, 2].

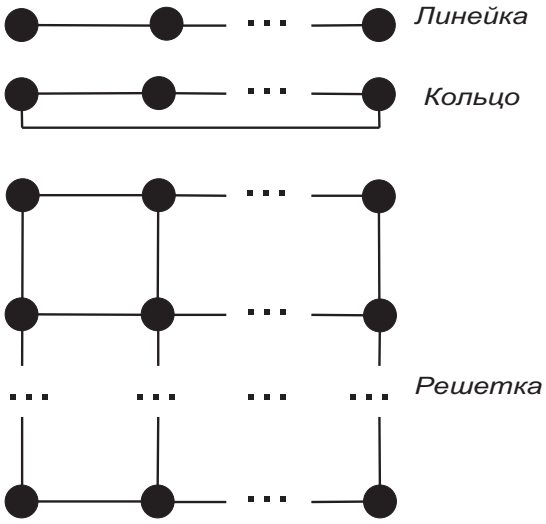

Рис. 2. Типовые графы параллельных программ

В работе рассматривается задача вложения графа параллельной программы $G_p(V_p, E_p)$ в граф распределённой вычислительной системы $G_s(V_s, E_s)$, $n = |V_p| = |V_s|$ — число ветвей параллельной программы (число ЭМ ВС). Цель вложения: взаимно однозначно отобразить вершины графа программы G_p в вершины графа системы G_s так, чтобы при этом ребра G_p совпали с ребрами G_s (установить изоморфизм графа программы с остовным подграфом графа ВС).

Массовый параллелизм обработки информации в нейронных сетях позволяет рассматривать их как перспективное высокопроизводительное и надёжное средство для решения сложных, в том числе и оптимизационных, задач. Наибольший интерес для решения задач дискретной оптимизации представляют рекуррентные нейронные сети [6 – 11]. Модель глобально сходящихся рекуррентных нейронных сетей Хопфилда хорошо согласуется с парадигмой самостабилизации Дейкстры [12]. Это означает, что реализуемые сетями Хопфилда вложения графов параллельных программ в графы распределённых вычислительных систем являются самостабилизирующимися. Важность использования самостабилизирующихся вложений обусловлена возможностью

нарушения регулярности графов ВС вследствие выхода из строя элементарных машин и межмашинных соединений.

1. Сеть Хопфилда для задачи вложения

Рассмотрим матрицу нейронов v размером $n \times n$, где каждая строка матрицы соответствует ветви параллельной программы, каждый столбец матрицы соответствует ЭМ. Каждая строка и каждый столбец матрицы содержат один и только один ненулевой элемент, равный единице, остальные элементы равны нулю. Энергия соответствующей нейронной сети Хопфилда описывается функцией Ляпунова

$$E = \frac{C}{2}\left[\sum_x\left(\sum_j v_{xj} - 1\right)^2 + \sum_i\left(\sum_y v_{yi} - 1\right)^2\right] + \frac{D}{2}\sum_x\sum_i\sum_{y\in Nb_p(x)}\sum_{j\neq i} v_{xi}v_{yj}d_{ij}. \quad (1)$$

Здесь v_{xi} — состояние нейрона строки x и столбца i матрицы v; C и D — параметры функции Ляпунова. В квадратных скобках минимум первого слагаемого соответствует наличию одного единичного элемента в каждой строке матрицы v, а минимум второго означает наличие одного единичного элемента в каждом столбце этой матрицы. Минимум суммы первых двух слагаемых обеспечивается в случае, когда каждая строка и каждый столбец матрицы содержат один и только один элемент, равный 1, остальные элементы — нули (корректное решение). Минимум третьего слагаемого обеспечивает минимум суммы расстояний между соседними ветвями параллельной программы при их размещении на графе системы. Здесь d_{ij} — расстояние между вершинами графа ВС, содержащими соседние ветви параллельной программы («растяжение» ребра графа программы на графе ВС); $Nb_p(x)$ — множество вершин, смежных с вершиной x на графе программы.

Динамика сети Хопфилда с непрерывными состояниями и непрерывным временем [6], минимизирующей функцию (1), описывается системой уравнений

$$\frac{\partial u_{xi}}{\partial t} = -\frac{\partial E}{\partial v_{xi}}, \quad (2)$$

где u_{xi} — активация нейрона с индексами $x, i = 1, \ldots, n$; $v_{xi} = (1 + \exp(-\beta u_{xi}))^{-1}$ — состояние (выходной сигнал) нейрона; β — параметр функции активации. Из (1) и (2) получаем

$$\frac{\partial u_{xi}}{\partial t} = -C\left(\sum_j v_{xj} + \sum_y v_{yi} - 2\right) - D\sum_{y\in Nb_p(x)}\sum_{j\neq i} v_{yj}d_{ij}. \quad (3)$$

Разностный вариант дифференциального уравнения (3) имеет вид

$$u_{xi}^{t+1} = u_{xi}^t - \Delta t\left[C\left(\sum_j v_{xj} + \sum_y v_{yi} - 2\right) + D\sum_{y\in Nb_p(x)}\sum_{j\neq i} v_{yj}d_{ij}\right], \quad (4)$$

где Δt — величина временного шага. Начальные значения u_{xi}^0, $x, i = 1, \ldots, n$, задаются случайным образом.

Выбор параметров β, Δt, C, D [6 – 11] определяет качество решения v системы уравнений (4). В [9] получена формула взаимосвязи между параметрами C и D, которая для задачи (1)–(4) приобретает вид

$$C \approx 100 \cdot D. \quad (5)$$

Из (4) и (5) следует, что изменения параметров Δt и D влияют на решение системы уравнений (4) одинаково. Поэтому положим $\Delta t = 1$ и получим уравнение

$$u_{xi}^{t+1} = u_{xi}^{t} - C\left(\sum_j v_{xj} + \sum_y v_{yi} - 2\right) - D \sum_{y \in Nb_p(x)} \sum_{j \neq i} v_{yj} d_{ij}. \qquad (6)$$

Положим $\beta = 0{,}1$ (это значение выбрано в работе [10]) и будем подбирать D таким образом, чтобы обеспечить отсутствие некорректных решений. Следует отметить, что в экспериментах зачастую некорректные решения получаются, если при достижении заданного максимального числа итераций, например $t_{\max} = 10000$, не достигнуто условие сходимости $\sum\limits_{xi} |u_{xi}^{t} - u_{xi}^{t-1}| < \varepsilon$ при заданном $\varepsilon = 0{,}01$.

2. Вложение сетью Хопфилда

Качество вложения будем оценивать числом совпадений рёбер графа программы с ребрами графа ВС. Это число назовем рангом вложения. Ранг вложения является приближённой оценкой качества вложения, поскольку одному и тому же рангу могут соответствовать вложения с различными растяжениями рёбер графа программы на графе системы. Однако максимальное значение ранга, равное числу рёбер графа программы, соответствует оптимальному вложению и минимуму суммы $S_d = \sum\limits_x \sum\limits_i \sum\limits_{y \in Nb_p(x)} \sum\limits_{j \neq i} v_{xi} v_{yj} d_{ij}$ в (1). Цель исследования — определить значения параметров алгоритма вложения, при которых достигается максимум вероятности оптимального вложения. В качестве примера рассмотрим вложение графа программы типа «линейка» в 2D-тор. Максимальное значение ранга вложения для линейки с n вершинами равно $n - 1$.

Для экспериментального исследования качества вложения использованы гистограммы распределения частоты ранга вложения при заданном числе экспериментов, равном 100. Выполнены эксперименты по вложению графа программы типа «линейка» в граф системы типа «двумерный тор» при числе вершин $n = l^2$, $l \in \{3, 4\}$, где l — порядок циклической подгруппы.

При $D \geqslant 8$ получены корректные решения для $n = 9$ (рис. 3,*а*) и 16 (рис. 3,*б*). Но, как видно из рис. 3, для $D = 8$ количество решений с оптимальным вложением, соответствующим максимуму ранга вложения, мало.

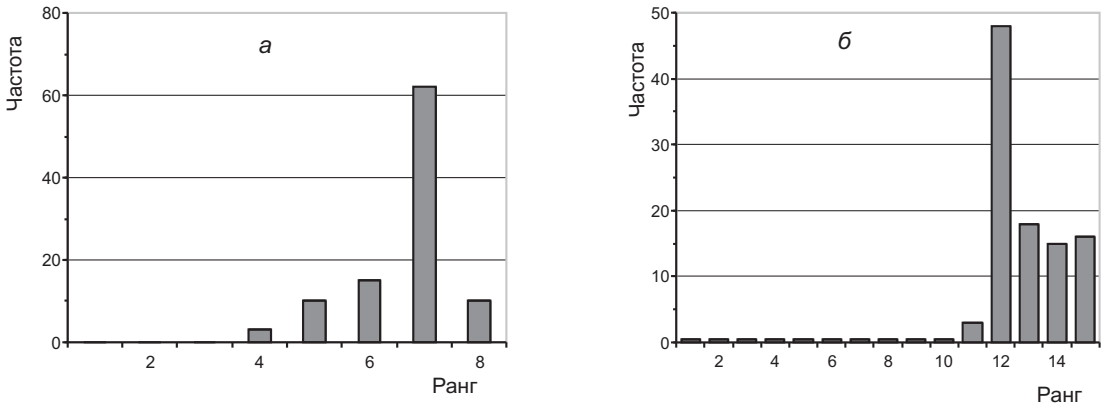

Рис. 3. Гистограммы вложений для нейронной сети (6)

Для повышения частоты оптимальных решений в уравнении (6) величины расстояний d_{ij} заменим величинами

$$c_{ij} = \begin{cases} d_{ij}, & d_{ij} = 1, \\ p \cdot d_{ij}, & d_{ij} > 1, \end{cases} \tag{7}$$

где p — коэффициент штрафа за превышение расстоянием d_{ij} величины 1, то есть за несовпадение ребра графа программы с ребром графа системы. Получаем уравнение

$$u_{xi}^{t+1} = u_{xi}^{t} - C\left(\sum_j v_{xj} + \sum_y v_{yi} - 2\right) - D \sum_{y \in Nb_p(x)} \sum_{j \neq i} v_{yj} c_{ij}. \tag{8}$$

При $p = n$ для вышеописанных вложений получены гистограммы для $n = 9$ (рис. 4,*а*) и 16 (рис. 4,*б*), свидетельствующие о заметном улучшении качества вложения, но для $n = 16$ максимум частоты приходится на субоптимальные решения с рангом 13.

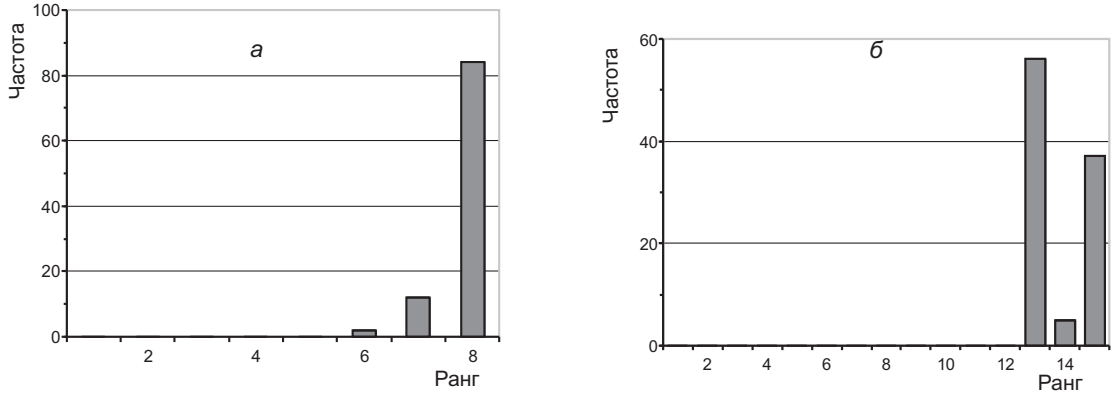

Рис. 4. Гистограммы вложений для нейронной сети (8)

3. Метод расщепления

С целью уменьшения числа локальных экстремумов функции (1) произведём разбиение множества $\{1, 2, \ldots, n\}$ значений индексов x и i переменных v_{xi} на $1 < K < n$ подмножеств I_k, $k = 1, 2, \ldots, K$, таких, что множества вершин с индексами из одного и того же I_k образуют связные подграфы как на графе G_p параллельной программы, так и на графе G_s вычислительной системы. В рассматриваемой здесь задаче возьмем $I_k = \{(k-1)q+1, \ldots, k \cdot q\}$, $q = n/K$, и индексы $x \in I_k$ будем отображать только в множество индексов I_k, то есть приведём матрицу решений v к блочно-диагональному виду. Тогда с учетом (7) функция Ляпунова (1) приобретает следующий вид:

$$E = \frac{C}{2} \sum_{k=1}^{K} \left[\sum_{x \in I_k} \left(\sum_{j \in I_k} v_{xj} - 1 \right)^2 + \sum_{i \in I_k} \left(\sum_{y \in I_k} v_{yi} - 1 \right)^2 \right] + \frac{D}{2} \sum_{k=1}^{K} \sum_{x \in I_k} \sum_{i \in I_k} \sum_{y \in Nb_p(x)} \sum_{j \neq i} v_{xi} v_{yj} c_{ij},$$

и динамика сети Хопфилда описывается уравнением

$$\begin{gathered} u_{xi}^{t+1} = u_{xi}^{t} - C\left(\sum_{j \in I_k} v_{xj} + \sum_{y \in I_k} v_{yi} - 2\right) - D \sum_{y \in Nb_p(x)} \sum_{j \neq i} v_{yj} c_{ij}, \\ v_{xi} = \left(1 + \exp(-\beta u_{xi})\right)^{-1}, \quad x, i \in I_k, \quad k = 1, 2, \ldots, K. \end{gathered} \tag{9}$$

При этом $v_{xi} = 0$ для $x \in I_k, i \notin I_k, k = 1, 2, ..., n$. При использовании этого подхода, который мы назовем методом расщепления, для линейки с числом вершин $n = 16$ при $K = 2$ получена гистограмма вложения в 2D-тор (рис. 5). Сравнивая рис. 4,*б* и рис. 5, убеждаемся в том, что использование метода расщепления позволило существенно повысить частоту оптимальных вложений. Увеличение параметра D до значения $D = 32$ приводит к дополнительному росту частоты оптимальных вложений (рис. 6).

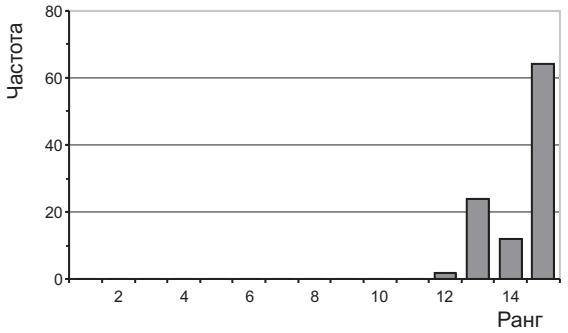

Рис. 5. Гистограмма вложений для нейронной сети (9):
$n = 16$, $K = 2$, $D = 8$

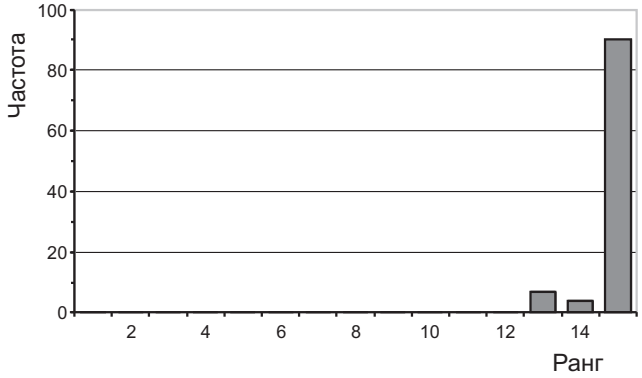

Рис. 6. Гистограмма вложений для нейронной сети (9):
$n = 16$, $K = 2$, $D = 32$

4. Вложение сетью Вана

В рекуррентной сети Вана [11] сумма $S_d = \sum\limits_x \sum\limits_i \sum\limits_{y \in Nb_p(x)} \sum\limits_{j \neq i} v_{xi} v_{yj} d_{ij}$ в (1) умножается на величину $\exp(-t/\tau)$, где τ — параметр. Для сети Вана уравнение (9) приобретает следующий вид:

$$u_{xi}^{t+1} = u_{xi}^t - C\left(\sum_{j \in I_k} v_{xj} + \sum_{y \in I_k} v_{yi} - 2\right) - D \sum_{y \in Nb_p(x)} \sum_{j \neq i} v_{yj} c_{ij} \exp(-t/\tau),$$
$$v_{xi} = \left(1 + \exp(-\beta u_{xi})\right)^{-1}, \quad x, i \in I_k, \quad k = 1, 2, \ldots, K. \tag{10}$$

Введение множителя $\exp(-t/\tau)$ ускоряет сходимость рекуррентной сети, и количество некорректных решений сокращается. Так, для трехмерного тора с $n = 3^3 = 27$ вершинами при $p = n$, $K = 3$, $D = 256$, $\beta = 0{,}1$ для 100 решений получены такие результаты:

1) на сети Хопфилда получено 32 некорректных решения и 36 оптимальных решений (ранг 26);
2) на сети Вана при тех же значениях вышеуказанных параметров и $\tau = 500$ все 100 полученных решений корректны, причем 72 решения оптимальны. Оптимальные решения, как правило, достигались за несколько десятков шагов.

Заключение

Сформулирована задача вложения графов параллельных программ в графы распределённых вычислительных систем рекуррентными нейронными сетями. Экспериментально получены значения параметров, обеспечивающие отсутствие некорректных решений. Для вложения графа параллельной программы типа «линейка» в граф ВС типа 2D-тор показано, что благодаря введению в функцию Ляпунова параметра штрафа для рёбер графа программы, не совпадающих с рёбрами графа ВС, удаётся получить оптимальные решения при вложении линейки в двумерный тор для числа вершин $n \in \{9, 16\}$. Для увеличения вероятности (частоты) оптимальных вложений:

1) предложен метод расщепления, суть которого заключается в приведении матрицы решения к блочно-диагональному виду;
2) выполнен переход от сети Хопфилда к сети Вана, что ускорило сходимость.

В результате получены высокие частоты оптимальных решений (при числе экспериментов, равном 100):

1) порядка 80–90 % для двумерных торов ($n = 3^2 = 9$ и $n = 4^2 = 16$);
2) порядка 70 % для трехмерного тора ($n = 3^3 = 27$).

Дальнейшие исследования следует проводить в направлении увеличения вероятности получения оптимальных решений при росте числа вершин графа программы.

ЛИТЕРАТУРА

1. *Корнеев В. В.* Архитектура вычислительных систем с программируемой структурой. Новосибирск: Наука, 1985. 166 с.
2. *Тарков М. С.* Вложение структур параллельных программ в структуры живучих распределенных вычислительных систем // Автометрия. 2003. Т. 39. № 3. С. 84–96.
3. Cray T3E: `http://www.cray.com/products/systems/crayt3e/1200e.html`
4. *Yu H., Chung I.-H., Moreira J.* Topology Mapping for Blue Gene/L Supercomputer // Proc. of the ACM/IEEE SC2006 Conf. on High Performance Networking and Computing, November 11–17, 2006, Tampa, FL, USA. ACM Press, 2006. P. 52–64.
5. *Абрамов С. М., Заднепровский В. Ф., Шмелев А. Б., Московский А. А.* СуперЭВМ ряда 4 семейства «СКИФ»: Штурм вершины суперкомпьютерных технологий // Вестник Нижегородского университета им. Н. И. Лобачевского. 2009. № 5. С. 200–210.
6. *Меламед И. И.* Нейронные сети и комбинаторная оптимизация // Автоматика и телемеханика. 1994. № 11. С. 3–40.
7. *Kate A. S.* Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research // INFORMS J. Computing. 1999. V. 11. No. 1. P. 15–34.
8. *Tarkov M. S., Lapukhov S. A.* Mapping a parallel program structure onto distributed computer system structure by the Hopfield neural network with fuzzy scheduling parameters // Bulletin of the Novosibirsk Computing Center, Comp. Science. 2003. No. 19. P. 83–88.
9. *Feng G., Douligeris C.* The Convergence and Parameter Relationship for Discrete-Time Continuous-State Hopfield Networks // Proc. of Intern. Joint Conference on Neural Networks. 2001. P. 376–381.

10. *Тарков М. С.* Построение гамильтоновых циклов рекуррентной нейронной сетью в тороидальных графах с дефектами ребер // Труды XII Всерос. науч.-технич. конф. «НЕЙРОИНФОРМАТИКА-2010». Ч. 2. М., 2010. С. 26–34.
11. *Wang J.* Analysis and Design of a Recurrent Neural Network for Linear Programming // IEEE Trans. On Circuits and Systems-I: Fundamental Theory and Applications. 1993. V. 40. No. 9. P. 613–618.
12. *Dijkstra E. W.* Self-stabilizing systems in spite of distributed control // Commun. ACM. 1974. V. 17. No. 11. P. 643–644.

ПРИКЛАДНАЯ ТЕОРИЯ АВТОМАТОВ

DOI 10.17223/20710410/10/5

УДК 519.217

МНОГОПАРАМЕТРИЧЕСКАЯ КЛАССИФИКАЦИЯ АВТОМАТНЫХ МАРКОВСКИХ МОДЕЛЕЙ НА ОСНОВЕ ГЕНЕРИРУЕМЫХ ИМИ ПОСЛЕДОВАТЕЛЬНОСТЕЙ СОСТОЯНИЙ

А. Р. Нурутдинова, С. В. Шалагин

Казанский государственный технический университет им. А. Н. Туполева, г. Казань, Россия

E-mail: sshalagin@mail.ru, Alsu124@mail.ru

Предложена методика многопараметрической классификации автоматных марковских моделей (АММ) на основе генерируемых ими последовательностей состояний при использовании метода дискриминантного анализа. При этом АММ заданы стохастическими матрицами, принадлежащими к априори заданным подклассам. Определено множество классифицирующих признаков (МКП), позволяющее разделить АММ, заданные матрицами из различных подклассов. Предложена методика, позволяющая определить минимальную длину последовательностей, требуемых для вычисления МКП с заданной доверительной вероятностью.

Ключевые слова: *цепи Маркова, эргодические стохастические матрицы, идентификация,автономный верятностный автомат, автоматная марковская модель, статистическая обработка данных, дискриминантный анализ, линейные дискриминантные функции, информативность.*

Введение

Аппарат теории цепей Маркова (ЦМ) служит для моделирования поведения технических устройств, а также объектов из области экономики, социологии, образования и т. п. как в теоретическом, так и в прикладном аспектах [1 – 3]. В частности, автоматные марковские модели (АММ), определённые на основе аппарата теории ЦМ, позволяют описывать широкий класс устройств [4, 5].

Задача анализа ЦМ имеет важное прикладное значение, связанное с исследованием частотного распределения символов текстов по m-граммам, $m = 2, 3, \ldots$ [6, 7].

Современные информационные технологии статистической обработки данных, такие, как пакет прикладных программ Statistica 6.0 [8], позволяют ставить и решать задачи анализа АММ [9, 10]. При этом особая роль отводится отработке методики решения поставленных задач, а также исследованию множества классифицирующих признаков.

В [9, 10] решены задачи классификации и идентификации эргодических стохастических матриц (ЭСМ) по функционалам, вычисленным на их базе. Что касается решения данных задач для реализаций ЦМ, то в [11] определены подходы, связанные с определением биграмм, на основе которых вычисляются определенные функционалы [11 – 13]. Подобная схема характеризуется тем, что биграммы вычисляются с некоторой погрешностью относительно ЭСМ [11]. Данное обстоятельство способствует снижению точности анализа на основе указанных функционалов. Выходом из создавшейся ситуации

служит создание интегрированных признаков, которые вычисляются непосредственно по реализации ЦМ и являются информативными в плане дискриминации марковских последовательностей, вычисленных на основе определенных подклассов ЭСМ.

В работе на основе результатов, представленных в [14 – 16], решена задача многопараметрического анализа последовательностей ЦМ, полученных на базе АММ, заданных ЭСМ. Для множества классифицирующих признаков (МКП) анализируется возможность выделения априори заданных подклассов АММ в зависимости от подклассов, к которым принадлежат ЭСМ, определяющие АММ. Анализ проведен при допущении, что имеет место сходимость по вероятности элементов МКП, вычисленных эмпирически на основе ЦМ конечной длины N, к соответствующим элементам МКП, вычисленным теоретически на базе ЭСМ. Задача решается методом дискриминантного анализа (ДА).

1. Объекты классификации

Определение 1. Простая конечная однородная ЦМ задана в виде [1]

$$(S,\ P,\ \pi_0)\,, \tag{1}$$

где $S = \{s_0, s_1, \ldots, s_{n-1}\}$ — множество состояний ЦМ; P — эргодическая [1] стохастическая матрица вида $P = (p_{ij})$, $i, j = \overline{0, n-1}$; π_0 — n-мерный вектор начального распределения вероятности появления каждого состояния ЦМ.

Определение 2. Автономным вероятностным автоматом будем называть систему [4]

$$\left(\bar{U},\ S,\ P,\ \delta(u,\ s)\right), \tag{2}$$

где $\bar{U}$ — дискретная случайная величина [4], принимающая на входе значения $u_0, u_1, \ldots, u_{l-1}$ с распределением $\bar{p} = (p_0, p_1, \ldots, p_{l-1})$; l — размер имплицирующего вектора [4]; $\sum\limits_{i=0}^{l-1} p_i = 1$, $0 \leqslant p_i \leqslant 1$; $\delta(u, s)$ — функция переходов, ставящая в соответствие паре (u, s) однозначно новое состояние $s' \in S$. При этом $P = \sum\limits_{i=0}^{l-1} p_i A_i$, где A_i — простая матрица (по терминологии [5]). Элементы A_i, обозначенные через $a_{kj}(i)$, $k, j = \overline{0, n-1}$, определены из следующего соотношения:

$$a_{kj}(i) = \begin{cases} 1, & \text{если } \delta(u_i, s_k) = s_j, \\ 0, & \text{если } \delta(u_i, s_k) \neq s_j. \end{cases}$$

Определение 3. Цепь Маркова (1) является эргодической, если она состоит из одного эргодического класса, т. е. можно из каждого состояния перейти в любое другое [1].

Определение 4. Стохастическая матрица P, задающая эргодическую цепь Маркова, называется стохастической эргодической матрицей.

Модель устройства, заданная согласно (1) и (2), определена как АММ. На АММ накладывается следующее ограничение: значения положительных элементов ЭСМ кратны определённой величине D, которую будем называть дискретностью элементов ЭСМ. АММ, заданную на основе ЭСМ P, обозначим АММ(P).

Исследуются марковские последовательности, которые являются реализациями простых конечных однородных ЦМ. Они определяются АММ, заданными на основе различных подклассов эргодических стохастических матриц: квазитреугольных (Т) и

блочно-сообщающихся (Б). Матрицы класса Т подразделяются на верхние ($\text{Т}_\text{в}$) и нижние ($\text{Т}_\text{н}$). ЭСМ подкласса Б делятся на блочные правые ($\text{Б}_\text{п}$) и блочные левые ($\text{Б}_\text{л}$). Положительными элементами матриц заданных подклассов будут следующие:

1) для подкласса $\text{Т}_\text{н}$ — элементы p_{ij}, где $i = \overline{0, n-1}$, $j = \overline{0, i}$, и элемент $p_{0(n-1)}$;
2) для подкласса $\text{Т}_\text{в}$ — элементы p_{ij}, где $i = \overline{0, n-1}$, $j = \overline{i, n-1}$, и элемент $p_{(n-1)0}$;
3) для подкласса $\text{Б}_\text{п}$ — элементы p_{ij}, где $i, j = \overline{0, k-1}$, $k = [n/2]$, и p_{ij}, где $i, j = \overline{k,\ n-1}$, а также элементы $p_{(k-1)k}$ и $p_{k(k-1)}$;
4) для подкласса $\text{Б}_\text{л}$ — элементы p_{ij}, где $i = \overline{0,\ k-1}$, $j = \overline{k, n-1}$, $k = [n/2]$, и p_{ij}, где $i = \overline{k, n-1}$, $j = \overline{0, k-1}$.

«Угловые» элементы квазитреугольных матриц добавлены для сохранения у данных ЭСМ свойства эргодичности.

Возникает вопрос — последовательность состояний какой длины требуется сгенерировать с целью классификации АММ, определенной на основе ЭСМ, с заданной доверительной вероятностью?

2. Множество классифицирующих признаков

В работе решена задача распознавания АММ(P) в зависимости от того, к какому подклассу отнесены ЭСМ P, которые определяют порождаемые ими ЦМ.

Задача решена на основе заданного набора признаков $\tilde{f}_k$, $k = \overline{1-n, n-1}$, которые характеризуют частоту перехода ЦМ из состояния s_i в состояние s_j ($s_i, s_j \in S$, $i, j = \overline{0, n-1}$) для случая, когда $i - j = k$. Вычисление $\tilde{f}_k$ эквивалентно подсчету частот по биграмме, построенной на основе реализации ЦМ [7].

На основе реализации ЦМ длины N получим последовательность Y длины $(N-1)$, такую, что $y_z = k$ для $i - j = k$ ($y_z \in Y$, $z = \overline{1, N-1}$), если на z-м шаге ЦМ переходит из состояния s_i в состояние s_j.

Частота перехода ЦМ из состояния s_i в состояние s_j для случая, когда $i - j = k$, рассчитывается по формуле

$$\tilde{f}_k = \frac{1}{N-1}\sum_{z=1}^{N-1}\hat{y}_z, \quad \hat{y}_z = \begin{cases} 1, & \text{если } y_z = k, \\ 0, & \text{если } y_z \neq k, \end{cases} \qquad k = \overline{1-n, n-1}. \tag{3}$$

Вероятности перехода ЦМ f_k ($k = \overline{1-n, n-1}$) из состояния s_i в состояние s_j для случая, когда $i - j = k$, являющиеся теоретическими оценками для величин $\tilde{f}_k$, определяются на базе ЭСМ(P), определяющих АММ(P), и рассчитываются по формуле

$$f_k = \frac{1}{n}\sum_{i=1}^{n-|k|} m(i,k), \tag{4}$$

где $m(i,k) = p_{i(i-k)}$ для $k = \overline{1-n, 0}$ и $m(i,k) = p_{(i+k)i}$ для $k = \overline{1, n-1}$. Величина (3) асимптотически стремится к (4) при бесконечном возрастании длины ЦМ. При решении практических задач величина N является конечной. Отклонение $\varepsilon_k = |\tilde{f}_k - f_k|$ ($k = \overline{1-n, n-1}$) рассчитывается исходя из того, что $\tilde{f}_k$ — частота появления события — перехода ЦМ из состояния s_i в состояние s_j для случая $i - j = k$, $i,\ j = \overline{0, n-1}$. В свою очередь, f_k, $k = \overline{1-n, n-1}$, есть вероятность данного события. Примем в качестве допущения, что значение ε_k распределено по закону Гаусса [17]. Данным допущением можно пользоваться уже при $N \geqslant 5$ [17]. При условии, что величины $\tilde{f}_k$ вычисляются на основе последовательности Y длины $(N-1)$, данное отклонение вы-

числяется по формуле [17]

$$\varepsilon_k = \left| f_k - \tilde{f}_k \right| = t_\alpha \sqrt{\frac{\tilde{f}_k \left(1 - \tilde{f}_k\right)}{N-1}}, \quad k = \overline{1-n, n-1}, \tag{5}$$

где t_α — величина, определяющая для нормального закона число среднеквадратических отклонений, которое нужно отложить относительно центра рассеивания, чтобы вероятность попадания в полученный интервал была равна α. При этом для $\tilde{f}_k$ выполняется условие [17]

$$\tilde{f}_k \in \left[\frac{5}{N-1}, \frac{N-6}{N-1} \right]. \tag{6}$$

Замечание 1. Условие (6) может быть записано, согласно (3), в виде $\sum\limits_{z=1}^{N-1} \hat{y}_z \in \in [5, N-6]$, $\hat{y}_z \in Y$, $z = \overline{1, N-1}$.

В случае невыполнения условия (6) $\tilde{f}_k$ рассматривается как число появления заданного события в некотором числе опытов, которое распределено по биномиальному закону. В этом случае ε_k вычисляется согласно

$$\varepsilon_k = \max(|\tilde{f}_k - p_{1k}|, |p_{2k} - \tilde{f}_k|), \tag{7}$$

где p_{1k} и p_{2k} — верхняя и нижняя границы доверительного интервала, рассчитанные при заданном значении доверительной вероятности α:

$$\sum_{m=l}^{N-1} C_{N-1}^m p_{1k}^m \left(1 - p_{1k}\right)^{N-m-1} = \frac{\alpha}{2}; \quad \sum_{m=0}^{l} C_{N-1}^m p_{2k}^m \left(1 - p_{2k}\right)^{N-m-1} = \frac{\alpha}{2},$$

где l — количество переходов ЦМ из состояния s_i в состояние s_j для случая, когда $i - j = k$, $k = \overline{1-n, n-1}$ [16]. Перепишем (5) в виде

$$N = \frac{\tilde{f}_k \left(1 - \tilde{f}_k\right)}{\left| f - \tilde{f}_k \right|^2} t_\alpha^2 + 1 \leqslant \max_{\tilde{f}_k} \left(\frac{\tilde{f}_k \left(1 - \tilde{f}_k\right)}{\varepsilon^2} t_\alpha^2 \right) = \frac{1}{4\varepsilon^2} t_\alpha^2, \quad k = \overline{1-n, n-1}. \tag{8}$$

Тогда имеет место

Утверждение 1. Верхняя оценка длины последовательности N, на основе которой вычисляются значения признаков $\tilde{f}$, достаточной для идентификации с заданной доверительной вероятностью α АММ(P_1) и АММ(P_2), где P_1 и P_2 принадлежат различным подклассам ЭСМ, определяется согласно (8).

Согласно утверждению 1, определив признаки, имеющие максимальные из минимальных отклонений,

$$\varepsilon = \max_{k=\overline{1-n,n-1}} \left(\min f_k(P_1) - \min f_k(P_2) \right), \tag{9}$$

где $f_k(P_1)$, $f_k(P_2)$ — значения k-го признака, вычисленного для ЭСМ P_1 и P_2 соответственно, возможно определение N — минимальной длины ЦМ, достаточной для классификации АММ, определенной на основе ЭСМ, с заданной доверительной вероятностью α, согласно (8).

На основе утверждения 1 предложена методика дискриминации АММ(P) на подклассы, определяемые на основе ЭСМ P.

3. Методика решения задачи многопараметрической классификации (дискриминации)

Пусть элементы $p_{ij} > 0$ ЭСМ P кратны некоторому числу D и $p_{ij} \in [D, 1-D]$, $i,\ j = \overline{0, n-1}$. На основе данного определения по формуле (4) вычисляется диапазон значений для величин f_k, $k = \overline{1-n, n-1}$.

Для многопараметрической дискриминации АММ(P), $P \in \{\text{Т}_\text{н}, \text{Т}_\text{в}, \text{Б}_\text{п}, \text{Б}_\text{л}\}$, на базе порождаемых ею последовательностей — реализаций ЦМ в зависимости от того, к какому подклассу принадлежит ЭСМ P, определяющая АММ(P), требуется решить следующие задачи.

Задача 1. Найти область определения каждого признака вида (4).

Задача 2. Выявить, какие признаки $\tilde{f}_k$ являются значимыми при выделении того или иного подкласса ЭСМ P, задающих АММ(P) (определить рабочее множество).

Задача 3. Для заданной ЦМ вычислить элементы рабочего множества $\tilde{f}_k$ вида (3) с априори указанной точностью. При этом определяется минимальная длина N последовательности, снимаемой с выхода АММ, которая требуется для достижения заданной точности ε_k, вычисленной согласно (5).

Задача 4. Разделить множество реализаций ЦМ, снимаемых с выхода АММ(P), на группы, определяемые подклассами $\text{Т}_\text{н}$, $\text{Т}_\text{в}$, $\text{Б}_\text{п}$, $\text{Б}_\text{л}$, используя метод дискриминантного анализа.

Последовательное решение задач 1 – 4 определяет методику многопараметрической дискриминации ЦМ на базе признаков вида (3).

Решение задачи 1 основано на следующем утверждении.

Утверждение 2. Для признаков f_k, $k = \overline{1-n, n-1}$, вычисляемых на основе множества V_k положительных элементов матрицы P согласно (4), их минимальные значения, $\min f_k$, достигаются при условии, что $h = D$ для всех $h \in V_k$, а максимальные значения, $\max f_k$, — при условии, что $h = 1-D$ для всех $h \in V_k$.

Далее в формате $(\min f_k, \max f_k)$ приведены минимальные и максимальные значения признаков для рассматриваемых подклассов ЭСМ.

Минимальные и максимальные значения признаков для $\text{Т}_\text{н}$ равны

$$\begin{gathered} f_{1-n} = (D, 1-D), \quad f_k = (0,0),\ k = \overline{2-n, -1}, \\ f_0 = \left(D, 1-(2+n(n-1))0{,}5Dn^{-1}\right), \\ f_k = \left((1-kn^{-1})D, 1-(n+k-1)(n-k)0{,}5Dn^{-1}\right),\ k = \overline{1, n-1}. \end{gathered} \tag{10}$$

Минимальные и максимальные значения признаков для $\text{Т}_\text{в}$:

$$\begin{gathered} f_k = \left((1-kn^{-1})D, 1-(n+k-1)(n-k)0{,}5Dn^{-1}\right),\ k = \overline{1-n, -1}, \\ f_0 = \left(D, 1-(2+n(n-1))0{,}5Dn^{-1}\right), \\ f_k = (0,0),\ k = \overline{1, n-2}, \quad f_{n-1} = (D, 1-D). \end{gathered} \tag{11}$$

Найдем минимальные и максимальные значения признаков для $\text{Б}_\text{п}$, приняв $w = 0{,}5n$ и $v = [w] + 1$, где $n \geqslant 4$:

$$\begin{gathered} f_k = (0,0) \text{ при } k = \overline{1-n, -v} \text{ и } k = \overline{v, n-1}, \\ f_0 = \left(D, 1-2Dn^{-1}((v-1)^2-1)\right). \end{gathered} \tag{12}$$

При нечетном n:

$$f_{-1} = \left((n-1)Dn^{-1}, 1 - 2Dn^{-1}\left((v-1{,}25)^2 + \frac{23}{16}\right)\right),$$
$$f_1 = \left((n-1)Dn^{-1}, 1 - 2Dn^{-1}(v-1{,}5)^2 + 0{,}25\right),$$
$$f_k = \left((1+2k)Dn^{-1}, 1 - (v-1)(v+k) + D(v-2)(n+k-v)\right),\ k = \overline{1-v, -2}.$$

При четном n:

$$f_{-1} = f_1 = \left((n-1)Dn^{-1}, 1 - 2Dn^{-1}((w-1{,}5)^2 - 0{,}75)\right),$$
$$f_k = \left((1-2|k|)Dn^{-1}, 1 - Dn^{-1}(2w^2 - n(1+|k|) + 2|k| + 1)\right),\ k = \overline{-w, -2},\ k = \overline{2, w}.$$

Минимальные и максимальные значения признаков для Бл: $f_0 = (0, 0)$; при нечетном n:

$$\begin{gathered} f_k = \left(kDn^{-1}, 1 - k[w]D\right), \quad k = \overline{1, [w]}, \\ f_k = \left(-kDn^{-1}, 1 - k[w]D\right), \quad k = \overline{-[w], -1}, \\ f_k = \left([w]Dn^{-1}, 1 - 2[w]D\right), \quad k = [w] + 1, \\ f_k = \left((1-k)Dn^{-1}, 1 - (n-k)[w]D\right), \quad k = \overline{1-n, -[w]-2},\ k = \overline{[w]+2, n-1}; \end{gathered} \tag{13}$$

при четном n:

$$f_k = \left(|k|Dn^{-1}, 1 + |k|(1-w)D\right), \quad k = \overline{-w+1, -1},\ k = \overline{1, w-1},$$
$$f_k = \left((1 - |k|n^{-1})D, 1 - (n - |k|)(w-1)D\right), \quad k = \overline{1-n, -w},\ k = \overline{w, n-1}.$$

Результаты, полученные на основе (10)–(13), приведены в табл. 1

Решение задачи 2 требует введения следующих обозначений. Пусть Q — множество, обозначающее подкласс ЭСМ, принадлежащих одному и только одному из заданных подклассов — Тн, Тв, Бп или Бл; J_Q — множество индексов признаков f_k, для которых $f_k(Q) = 0$.

Утверждение 3. Признаки f_k вида (4), объединенные во множество R_Q, являются значимыми для дискриминации АММ($P \in Q$) и включаются в рабочее множество, если диапазоны их значений не пересекаются: $f_k(Q) = 0$, а $\min f_k(\neg Q) > 0$, $k \in J_Q$.

Утверждение 4. Объединение множеств $R_{Q^{(1)}}$ и $R_{Q^{(2)}}$ признаков f_k вида (4), значимых для дискриминации АММ($P_1 \in Q^{(1)}$) и АММ($P_2 \in Q^{(2)}$), $Q^{(1)} \cap Q^{(2)} = \varnothing$, есть искомое рабочее множество.

Согласно (10)–(13), признаки f_k равны 0 при следующих условиях:

$$f_k(\text{Т}_\text{н}) = 0, \quad k \in J_{\text{Т}_\text{н}} = \{2-n, 3-n, \ldots, -1\},$$
$$f_k(\text{Т}_\text{в}) = 0, \quad k \in J_{\text{Т}_\text{в}} = \{1, 2, \ldots, n-2\},$$
$$f_k(\text{Б}_\text{п}) = 0, \quad k \in J_{\text{Б}_\text{п}} = J_{\text{Б}_{\text{п}_1}} \cup J_{\text{Б}_{\text{п}_2}} = \{1-n, 2-n, \ldots, -v\} \cup \{v, v+1, \ldots, n-1\},$$
$$f_0(\text{Б}_\text{л}) = 0.$$

Таблица 1

Характеристики признаков f_k, $k = \overline{1-n, n-1}$

Класс P	$\min f_k$	$\max f_k$
$\text{Т}_\text{н}$	D, $k = 1-n$	$1-D$, $k = 1-n$
	D, $k = 0$	$1-(2+n(n-1))0{,}5Dn^{-1}$, $k = 0$
	$(1-kn^{-1})D$, $k = \overline{1, n-1}$	$1-(n+k-1)(n-k)0{,}5Dn^{-1}$, $k = \overline{1, n-1}$
	0, $k = \overline{2-n, -1}$	0, $k = \overline{2-n, -1}$
$\text{Т}_\text{в}$	$(1-kn^{-1})D$, $k = \overline{1-n, -1}$	$1-(n+k-1)(n-k)0{,}5Dn^{-1}$, $k = \overline{1-n, -1}$
	D, $k = 0$	$1-(n+k-1)(n-k)0{,}5Dn^{-1}$, $k = 0$
	D, $k = n-1$	$1-D$, $k = n-1$
	$0, k = \overline{1, n-2}$	$0, k = \overline{1, n-2}$
	$n \bmod 2 = 1$	
	$\min f_k$	$\max f_k$
$\text{Б}_\text{л}$	$kn^{-1}D$, $k = \overline{1, [w]}$	$1-k[w]D$, $k = \overline{1, [w]}$, $k = \overline{-[w], -1}$
	$[w]Dn^{-1}$, $k = -[w]-1$, $k = [w]+1$	$1-2[w]D$, $k = -[w]-1$, $k = [w]+1$
$\text{Б}_\text{п}$	$(1-k)n^{-1}D$, $k = \overline{1-n, -[w]-2}$, $k = \overline{[w]+2, n-1}$	$1-(n-k)[w]D$, $k = \overline{1-n, -[w]-2}$, $k = \overline{[w]+2, n-1}$
	$-kn^{-1}D$, $k = \overline{-[w], -1}$	
	$n \bmod 2 = 0$	
	$\min \lvert f_k \rvert$	$\max \lvert f_k \rvert$
$\text{Б}_\text{л}$	$\lvert k \rvert n^{-1}D$, $k = \overline{-w+1, -1}$, $k = \overline{1,\ w-1}$	$1-\lvert k \rvert\,(w-1)\,D$, $k = \overline{-w+1, -1}$, $k = \overline{1, w-1}$
$\text{Б}_\text{п}$	$(1-\lvert k \rvert n^{-1})$, $k = \overline{1-n, -w}$, $k = \overline{w, n-1}$	$1-(n-\lvert k \rvert)\,(w-1)$, $k = \overline{1-n, -w}$, $k = \overline{w, n-1}$

На основе утверждений 3 и 4 предложена методика определения рабочего множества (РМ) для дискриминации АММ(P), $P \in \{\text{Т}_\text{н}, \text{Т}_\text{в}, \text{Б}_\text{п}, \text{Б}_\text{л}\}$, включающая три этапа (по количеству подклассов ЭСМ за вычетом единицы).

Этап 1. Выделяются признаки, которые разделяют последовательности, порождаемые АММ($P\in$Т$_{\text{н}}$), от последовательностей, порождаемых АММ(M), где $M\in\{$Т$_{\text{в}}$, Б$_{\text{п}}$, Б$_{\text{л}}\}$.

Этап 2. Определяется множество признаков, которые разделяют последовательности, снимаемые с выхода АММ($P\in$Т$_{\text{в}}$), от последовательностей, порождаемых АММ($M\in$Б).

Этап 3. Вычисляются признаки, разделяющие реализации цепи Маркова, снимаемые с выхода АММ($P\in$Б$_{\text{п}}$), от реализаций цепей Маркова, порождаемых АММ($P\in$Б$_{\text{л}}$).

При выполнении этапа 1 определяются признаки, выделяющие Т$_{\text{н}}$ из множества {Т$_{\text{н}}$, Т$_{\text{в}}$, Б$_{\text{п}}$, Б$_{\text{л}}$}. Найдем максимальное отклонение для признаков, отделяющих Т$_{\text{н}}$ от Т$_{\text{в}}$. Для этого определим следующие значения:

$A_1 = \max\limits_{k\in J_{\text{Т}_{\text{н}}}}|\min f_k(\text{Т}_{\text{н}}) - \min f_k(\text{Т}_{\text{в}})| = \max\limits_{k\in J_{\text{Т}_{\text{н}}}}(\min f_k(\text{Т}_{\text{в}})) = (n-1)Dn^{-1}$ для $k=-1$,

$B_1 = \max\limits_{k\in J_{\text{Т}_{\text{в}}}}|\min f_k(\text{Т}_{\text{н}}) - \min f_k(\text{Т}_{\text{в}})| = \max\limits_{k\in J_{\text{Т}_{\text{в}}}}(\min f_k(\text{Т}_{\text{н}})) = (n-1)Dn^{-1}$ для $k=1$.

Максимальное отклонение для признаков, отделяющих Б$_{\text{п}}$ от Т$_{\text{н}}$, найдем согласно формуле

$$C_1 = \max\limits_{k\in(J_{\text{Т}_{\text{н}}}\setminus J_{\text{Б}_{\text{п}_1}})\cup J_{\text{Б}_{\text{п}_2}}}|\min f_k(\text{Т}_{\text{н}}) - \min f_k(\text{Б}_{\text{п}})| = \\ = \max\left(\max\limits_{k\in(J_{\text{Т}_{\text{н}}}\setminus J_{\text{Б}_{\text{п}_1}})}(\min f_k(\text{Б}_{\text{п}})),\ \max\limits_{k\in J_{\text{Б}_{\text{п}_2}}}(\min f_k(\text{Т}_{\text{н}}))\right). \tag{14}$$

При этом $J_{\text{Т}_{\text{н}}}\setminus J_{\text{Б}_{\text{п}_1}} = \{-[w], -[w]+1, \ldots, -1\}$. На этом интервале $\max\limits_k(\min f_k(\text{Б}_{\text{п}})) = (n-1)Dn^{-1}$ при $k=-1$, а для $k\in J_{\text{Б}_{\text{п}_2}}$ получим $\max\limits_k(\min f_k(\text{Т}_{\text{н}})) = (n-1)Dn^{-1}$ при $k=1$. В результате $C_1 = (n-1)Dn^{-1}$.

Отклонения для признаков, отделяющих Б$_{\text{л}}$ от Т$_{\text{н}}$, вычисляются, согласно (14), как $E_1 = \max\limits_{k\in J_{\text{Т}_{\text{н}}}\cup\{0\}}|\min f_k(\text{Т}_{\text{н}}) - \min f_k(\text{Б}_{\text{л}})|$. В данном случае $J_{\text{Т}_{\text{н}}}\cup\{0\} = \{2-n, 3-n, \ldots, 0\}$, $\max\limits_{k\in J_{\text{Т}_{\text{н}}}}(\min f_k(\text{Б}_{\text{л}})) = [w]Dn^{-1}$, а $\min f_0(\text{Т}_{\text{н}}) = D$. В результате $E_1 = D$.

Из этапа 1 следует вывод: для того чтобы выделить Т$_{\text{н}}$ из множества {Т$_{\text{н}}$, Т$_{\text{в}}$, Б$_{\text{п}}$, Б$_{\text{л}}$}, достаточно вычислить значения признаков с максимальным отклонением, равным $\min(A_1, B_1, C_1, E_1) = (n-1)Dn^{-1}$. В РМ войдут признаки $\tilde{f}_k$, для которых f_k удовлетворяет условию: $|\min f_k(\text{Т}_{\text{н}}) - \min f_k(Q)| \geqslant (n-1)Dn^{-1}$, $Q\in\{$Т$_{\text{н}}$, Т$_{\text{в}}$, Б$_{\text{п}}$, Б$_{\text{л}}\}$. Определим данное РМ как $R_{\text{Т}_{\text{н}}}$.

На этапе 2 выявлены признаки, выявляющие ЭСМ подкласса Т$_{\text{в}}$ из множества Б $= \{$Б$_{\text{п}}$, Б$_{\text{л}}\}$.

Максимальное отклонение для признаков, отделяющих ЭСМ Б$_{\text{п}}$ от Т$_{\text{в}}$, согласно (13), равно $A_2 = \max\limits_{k\in J_{\text{Т}_{\text{в}}}\setminus J_{\text{Б}_{\text{п}_2}}}|\min f_k(\text{Т}_{\text{в}}) - \min f_k(\text{Б}_{\text{п}})| = (n-1)Dn^{-1}$, $k=1$.

Максимальное отклонение для признаков, отделяющих Б$_{\text{л}}$ от Т$_{\text{в}}$, равно, согласно (14), $B_2 = \max\limits_{k\in\{0\}\cup J_{\text{Т}_{\text{в}}}}|\min f_k(\text{Т}_{\text{в}}) - \min f_k(\text{Б}_{\text{л}})| = D$. При этом $\{0\}\cup J_{\text{Т}_{\text{в}}} = \{0, 1, \ldots, n-2\}$, $\min f_0(\text{Т}_{\text{в}}) = D$, а $\max\limits_{k\in J_{\text{Т}_{\text{в}}}}(\min f_k(\text{Б}_{\text{л}})) = [w]Dn^{-1}$.

Таким образом, на этапе 2, чтобы отделить Т$_{\text{в}}$ от Б, достаточно выделить признаки с максимальным отклонением, равным $\min(A_2,\ B_2) = (n-1)Dn^{-1}$. В РМ, которое определим как $R_{\text{Т}_{\text{в}}}$, войдут признаки $\tilde{f}_k$, для которых f_k удовлетворяет условию $|\min f_k(\text{Т}_{\text{в}}) - \min f_k(\text{Б})| \geqslant (n-1)Dn^{-1}$.

На третьем этапе определены признаки, которые отделяют ЭСМ Б$_{\text{п}}$ от Б$_{\text{л}}$. Максимальное отклонение для них, согласно (14), равно $A_3 = \max\limits_{k\in\{0\}\cup J_{\text{Б}_{\text{п}}}}|\min f_k(\text{Б}_{\text{п}}) - \min f_k(\text{Б}_{\text{л}})|$.

При этом $\{0\} \cup J_{\text{Бп}} = \{1-n, 2-n, \ldots, -v, 0, v, v+1, \ldots, n-1\}$, $\min f_k\,(\text{Бп}) = D$ при $k = 0$, а $\max\limits_{k \in J_{\text{Бп}}} (\min f_k(\text{Бл})) = [w]Dn^{-1}$.

В РМ, которое определено как $R_{\text{Бп}}$, войдут признаки $\tilde{f}_k$, для которых f_k удовлетворяет условию: $|\min f_k(\text{Бп}) - \min f_k(\text{Бл})| \geqslant D$.

Таким образом, согласно данным, полученных на этапах 1 – 3, имеет место

Замечание 2. РМ для дискриминации АММ(P), $P \in \{\text{Т}_\text{н}, \text{Т}_\text{в}, \text{Бп}, \text{Бл}\}$, на основе генерируемых ими ЦМ равно $R = R_{\text{Т}_\text{н}} \cup R_{\text{Т}_\text{в}} \cup R_{\text{Бп}}$. При этом максимально допустимое отклонение ε, вычисленное по формуле (9), равно $(n-1)Dn^{-1}$.

Решение задачи 3. Согласно (3), вычислим элементы РМ R, которые позволяют дискриминировать на подклассы последовательности, порождаемые АММ(P), в зависимости от того, к какому подклассу принадлежит P. Минимальное количество испытаний, необходимых для выделения подклассов реализаций ЦМ, порождаемых АММ(P), определяется на основе (5) (при выполнении условия (6) для каждого из признаков РМ) согласно неравенству

$$N = \max_{f_k \in R} \left(\tilde{f}_k \left(1 - \tilde{f}_k\right)\ t_\alpha^2 \varepsilon_k^{-2} + 1 \right) \leqslant \frac{n^2 t_\alpha^2}{4(n-1)^2 D^2} + 1, \tag{15}$$

где ε_k определяется как максимально допустимое отклонение $\tilde{f}_k \in R$ от f_k, определённое при решении второй задачи — $(n-1)Dn^{-1}$. Чем выше ε_k, тем меньше минимальное число испытаний, определенное согласно (15). Примем $\alpha = 0{,}95$, откуда следует, что $t_\alpha = 1{,}96$ [17], $D = 5 \cdot 10^{-2}$. Таким образом, (15) будет представлено в виде: $N = \max\limits_{f_k \in R} \left(1536{,}64 \cdot n^2\,(n-1)^{-2} \cdot \tilde{f}_k \left(1 - \tilde{f}_k\right) + 1\right)$. В результате, согласно (15), при $\tilde{f}_k = 0{,}5$ получим $N \leqslant 384{,}16 \cdot n^2\,(n-1)^{-2} + 1$.

Замечание 3. В случае, когда (6) не выполняется для признаков R с множеством индексов J, минимальное число испытаний для $\tilde{f}_k$, $k \in J$, определяется на базе (7) путем решения системы уравнений относительно N_k:

$$\begin{cases} (n-1)Dn^{-1} \geqslant \tilde{f}_k - p_{1k}, \\ (n-1)Dn^{-1} \geqslant p_{2k} - \tilde{f}_k, \end{cases} \quad \text{или} \quad \begin{cases} p_{1k} \geqslant \tilde{f}_k - (n-1)Dn^{-1}, \\ p_{2k} \leqslant (n-1)Dn^{-1} + \tilde{f}_k, \end{cases} \tag{16}$$

где p_{1k} и p_{2k} определяются, согласно (7), при $\alpha = 0{,}95$:

$$\sum_{m=l}^{N_k-1} C_{N_k-1}^m p_{1k}^m\,(1-p_{1k})^{N_k-m-1} = 0{,}475, \quad \sum_{m=0}^{l} C_{N_k-1}^m p_{2k}^m\,(1-p_{2k})^{N_k-m-1} = 0{,}475.$$

Здесь l — количество переходов ЦМ из состояния s_i в состояние s_j для случая, когда $i - j = k$, $k = \overline{1-n, n-1}$.

Утверждение 5. Минимальное количество испытаний, необходимых для выделения при заданном α подклассов ЦМ, реализации которых порождаются АММ(P), определяется как $N = \max(N_k, N_z)$, где N_k, $k \in J$, вычислено на базе (16), а N_z находится, согласно (15), для $z = \overline{1-n, n-1}$.

Задачи 2 и 3 решены для ЦМ, заданных ЭСМ $P \in \{\text{Т}_\text{н}, \text{Т}_\text{в}, \text{Бп}, \text{Бл}\}$, при $n = 5$. Области определения для признаков f_k приведены в табл. 2; в ячейках заданы минимальные и максимальные значения выбранных признаков.

Таблица 2

Области определения для признаков для ЭСМ при $n = 5$

Признак	$\text{Т}_\text{н}$	$\text{Т}_\text{в}$	$\text{Б}_\text{п}$	$\text{Б}_\text{л}$
f_{-4}	$(D, 1-D)/5$	$(D, 1-4D)/5$	$(0, 0)$	$(D, 1-D)/5$
f_{-3}	$(0, 0)$	$(2D, 1-7D)/5$	$(0, 0)$	$(2D, 1-2D)/5$
f_{-2}	$(0, 0)$	$(3D, 1-9D)/5$	$(D, 1-2D)/5$	$(2D, 1-2D)/5$
f_{-1}	$(0, 0)$	$(4D, 1-10D)/5$	$(4D, 1-9D)/5$	$(D, 1-D)/5$
f_0	$(5D, 1-11D)/5$	$(5D, 1-11D)/5$	$(5D, 1-10D)/5$	$(0, 0)$
f_1	$(4D, 1-10D)/5$	$(0, 0)$	$(4D, 1-8D)/5$	$(D, 1-2D)/5$
f_2	$(3D, 1-9D)/5$	$(0, 0)$	$(D, 1-3D)/5$	$(2D, 1-4D)/5$
f_3	$(2D, 1-7D)/5$	$(0, 0)$	$(0, 0)$	$(2D, 1-4D)/5$
f_4	$(D, 1-4D)/5$	$(D, 1-D)/5$	$(0, 0)$	$(D, 1-3D)/5$

Результаты этапа 1 даны в табл. 3, где показано, с каким максимальным отклонением допускается вычислять значение того или иного признака. Для выделения $\text{Т}_\text{н}$ из множества подклассов $\{\text{Т}_\text{н}, \text{Т}_\text{в}, \text{Б}_\text{п}, \text{Б}_\text{л}\}$ достаточно вычислить признаки $\tilde{f}_k$, $k = \overline{-1, 1}$, с $\varepsilon_k \leqslant D$.

Таблица 3

Отклонения для признаков, выделяющих $\mathbf{\text{Т}_\text{н}}$ из $\{\mathbf{\text{Т}_\text{н}}, \mathbf{\text{Т}_\text{в}}, \mathbf{\text{Б}_\text{п}}, \mathbf{\text{Б}_\text{л}}\}$

Подкласс	$\tilde{f}_{-4}$	$\tilde{f}_{-3}$	$\tilde{f}_{-2}$	$\tilde{f}_{-1}$	$\tilde{f}_0$	$\tilde{f}_1$	$\tilde{f}_2$	$\tilde{f}_3$	$\tilde{f}_4$	Макс. откл.
$\text{Т}_\text{в}$	—	$0{,}4D$	$0{,}6D$	$0{,}8D$	—	$0{,}8D$	$0{,}6D$	$0{,}4D$	—	$0{,}8D$
$\text{Б}_\text{п}$	$0{,}2D$	—	$0{,}2D$	$0{,}8D$	—	—	$0{,}4D$	$0{,}4D$	$0{,}2D$	$0{,}8D$
$\text{Б}_\text{л}$	—	$0{,}4D$	$0{,}4D$	$0{,}2D$	D	—	—	—	—	D

Данные, полученные на этапе 2, сведены в табл. 4 — для выделения подкласса $\text{Т}_\text{в}$ из множества подклассов $\{\text{Т}_\text{в}, \text{Б}_\text{п}, \text{Б}_\text{л}\}$ достаточно вычислить признаки $\tilde{f}_k$, $k = 0, 1$, с максимальным отклонением D.

Таблица 4

Отклонения для признаков, выделяющих $\mathbf{\text{Т}_\text{в}}$ из $\{\mathbf{\text{Т}_\text{в}}, \mathbf{\text{Б}_\text{п}}, \mathbf{\text{Б}_\text{л}}\}$

Подкласс	$\tilde{f}_{-4}$	$\tilde{f}_{-3}$	$\tilde{f}_{-2}$	$\tilde{f}_{-1}$	$\tilde{f}_0$	$\tilde{f}_1$	$\tilde{f}_2$	$\tilde{f}_3$	$\tilde{f}_4$	Макс. откл.
$\text{Б}_\text{п}$	$0{,}2D$	$0{,}4D$	$0{,}4D$	—	—	$0{,}8D$	$0{,}2D$	—	$0{,}2D$	$0{,}8D$
$\text{Б}_\text{л}$	—	—	—	—	D	$0{,}2D$	$0{,}4D$	$0{,}4D$	—	D

В табл. 5 приведены результаты выполнения этапа 3, согласно которым достаточно вычислить признак $\tilde{f}_0$ с максимальным отклонением D.

Таблица 5

Отклонения для признаков, разделяющих подклассы $\text{Б}_\text{п}$ и $\text{Б}_\text{л}$

Подкласс	$\tilde{f}_{-4}$	$\tilde{f}_{-3}$	$\tilde{f}_{-2}$	$\tilde{f}_{-1}$	$\tilde{f}_0$	$\tilde{f}_1$	$\tilde{f}_2$	$\tilde{f}_3$	$\tilde{f}_4$	Макс. откл.
$\text{Б}_\text{л}$	$0{,}2D$	$0{,}4D$	—	—	D	—	—	$0{,}4D$	$0{,}2D$	D

На основе данных из табл. 2–4 определяем РМ для дискриминации реализаций ЦМ, порождаемых АММ(P), $P \in \{\text{Т}_\text{н}, \text{Т}_\text{в}, \text{Б}_\text{п}, \text{Б}_\text{л}\}$. При этом $\varepsilon_k = D$, $k = \overline{-1,1}$.

Согласно формуле (3), вычислим элементы РМ $\tilde{f}_k$, $k = \overline{-1,1}$, которые позволяют дискриминировать на подклассы последовательности, порождаемые АММ(P), в зависимости от того, к какому подклассу принадлежит P. Минимальное количество испытаний, необходимых для однозначного выделения подклассов ЦМ, порождаемых АММ(P), определяется на основе (5) и (14):

$$N = \max_{\tilde{f}_k \in R} \left(2401 \cdot \tilde{f}_k \left(1 - \tilde{f}_k\right) + 1\right) \leqslant 602,$$

где ε_k определяется как максимально допустимое отклонение $\tilde{f}_k$ от f_k, $k = \overline{-1,1}$, определенное при решении задачи 2 при $n = 5$. Чем выше данная величина, тем меньше минимальное число испытаний, определенное согласно (15); если $\alpha = 0{,}95$, то $t_\alpha = 1{,}96$; $D = 5 \cdot 10^{-2}$.

4. Получение и интерпретация результатов

Решение задачи 4 предполагает получение последовательностей, снимаемых с выхода АММ(P), где $P \in \{\text{Т}_\text{н}, \text{Т}_\text{в}, \text{Б}_\text{п}, \text{Б}_\text{л}\}$. Длина данных последовательностей — число испытаний ЦМ — определяется на основе (15) при выполнении условия (5) для всех признаков РМ, либо на базе (16) — в противном случае.

Указанные задачи решены для ЦМ, снимаемых с выхода АММ(P). При этом P принадлежит одному из подклассов $\text{Т}_\text{н}$, $\text{Т}_\text{в}$, $\text{Б}_\text{п}$, $\text{Б}_\text{л}$ размерности $n = 5$. Объекты $\text{Т}_\text{н}$, $\text{Т}_\text{в}$, $\text{Б}_\text{п}$, $\text{Б}_\text{л}$ получены с использованием генератора псевдослучайных чисел табличного процессора MS Excel. Дискретность представления положительных элементов ЭСМ заданных подклассов $D = 5 \cdot 10^{-2}$.

Проведено по 5 серий испытаний для АММ(P), каждая из которых определена на основе 20 ЭСМ каждого из подклассов $\text{Т}_\text{н}$, $\text{Т}_\text{в}$, $\text{Б}_\text{п}$, $\text{Б}_\text{л}$ — всего 400 последовательностей. Верхняя граница количества испытаний в каждой серии определена на основе (15) и составляет 602, если для признаков из РМ, используемых при классификации полученных последовательностей, выполняется условие (5). Минимальное количество испытаний N, необходимых для выделения подклассов реализаций ЦМ, сгенерированных на основе АММ(P), не превышает 94 % от максимального значения $N_M = 602$.

Распределение значений величины N для последовательностей, генерируемых АММ, показано в табл. 6.

Таким образом, при проведении серий экспериментов для дискриминации реализаций ЦМ (при $\alpha = 0{,}95$), сгенерированных АММ(Т), в более чем половине случаев требуется провести 509 испытаний, что соответствует 85 % от значения N_M. Для дискриминации с заданной точностью реализаций ЦМ, описываемых матрицами Б, требуется в более чем половине случаев провести не менее 473 испытаний. Это составляет 79 % от N_M. Данные факты объяснимы более выраженной структурой ЭСМ Б по сравнению с ЭСМ Т. Для распознавания реализаций ЦМ, сгенерированных АММ(Б),

требуется последовательность в среднем меньшей длины, чем для реализаций ЦМ, сгенерированных АММ(Т).

Таблица 6

Распределение значений величины N для последовательностей, генерируемых АММ

Класс	Интервал изменения N	% от числа испытаний	% от значения N_M
$Т_н$	380 – 564	87	63 – 94
$Т_н$	86 – 233	13	14 – 39
$Т_в$	380 – 564	99	63 – 94
$Б_п$	417 – 564	97	69 – 94
$Б_л$	343 – 564	90	57 – 94

На основе признаков, входящих в РМ, проведено разделение полученных последовательностей на основе метода дискриминантного анализа. Результаты дискриминации представлены в табл. 7. Значение функции Фишера (F-критерия) для полученных результатов равно 221,8 при уровне значимости $p < 5 \cdot 10^{-5}$. Величина λ-статистики Уилкса равна $1{,}622 \cdot 10^{-2}$. Данные факты свидетельствуют о высоком качестве дискриминации.

Таблица 7

Характеристики признаков РМ

Признак	λ-статистика Уилкса	F-критерий (3,291)	Довер. интервал
f_{-1}	0,073929	409,5078	0,718518
f_0	0,074529	413,8994	0,619251
f_1	0,094515	560,1085	0,668676

В табл. 7 приведены для каждой из переменных РМ значения следующих вычисленных параметров: λ-статистика Уилкса, F-статистика для исключения и доверительный интервал для его значения. Значение функции Фишера, вычисленное как соотношение усредненных межгрупповых и внутригрупповых дисперсий [11, 12], будет для данных признаков больше числа 409 [4, 7], а значение λ-статистики Уилкса не превышает 0,1. Это свидетельствует о высоком вкладе в дискриминацию каждой переменной РМ.

Доля верно дискриминированных последовательностей, достигаемая при использовании заданного РМ, составляет 100 %, что подтверждают приведенные в работе предположения.

5. Обсуждение полученных результатов

Экспериментальные данные подтверждают гипотезу относительно способности признаков, характеризующих частоту перехода ЦМ из состояния s_i в состояние s_j при $i - j = k$ ($s_i, s_j \in S$, $i, j = \overline{0, n-1}$), классифицировать последовательности, генерируемые АММ(P). При этом матрицы P ЦМ принадлежат к разным подклассам, которые различаются порядком расположения положительных элементов, а следовательно, и

характером реализаций ЦМ, генерируемых АММ(P). Определена длина последовательностей, достаточная для дискриминации порождающих их АММ(P) на априори заданные группы с заданной доверительной вероятностью.

Особую роль играет методика выбора элементов РМ — признаков, позволяющих дискриминировать генерируемые последовательности. Оценка их области определения при заданной дискретности элементов ЭСМ дает возможность выявить РМ.

Следует отметить, что решение задачи 3 для ЭСМ при $n = 5$ позволяет получить реализации ЦМ, длины которых могут в общем случае быть меньше, чем верхняя граница 602 за счет варьирования значений признаков из РМ и их отклонения от величины 0,5. Максимальное значение длины генерируемой реализации ЦМ для рассматриваемой задачи ограничено сверху, при этом увеличение размерности n ЭСМ способствует снижению данной оценки. Этот факт объясняется тем, что для ЭСМ большой размерности различие между их подклассами легче идентифицировать как за счет большего количества признаков — $(2n - 2)$, так и из-за лучшего проявления принадлежности ЭСМ к определенному подклассу. Напротив, при малых значениях n различия между подклассами ЭСМ стираются. Следствием этого является увеличение верхней оценки для длины генерируемой ЦМ.

Верхняя оценка N, вычисленная на базе (11), будет пропорциональна $\left(\max\limits_{k\in J}\varepsilon_k\right)^{-1}$. Отклонение ε_k, в свою очередь, будет асимптотически стремиться к D при бесконечном возрастании размерности ЭСМ, а для матрицы размера 2×2 оно будет равно $0{,}5D$. Однако за счет особенностей структуры ЭСМ, выявляемых признаками из РМ, и за счет многомерности пространства признаков реальное значение длины ЦМ может быть уменьшено в среднем на 6–21 % относительно ее верхней оценки.

Заключение

Решена задача многопараметрической классификации АММ(P) на базе порождаемых ими последовательностей, где P принадлежит к априори заданным подклассам. Поставленная задача решается на основе вычисления частотных характеристик, а также выбора наиболее информативных из них.

Определена верхняя оценка для длины генерируемой последовательности, на базе которой с заданной точностью производится дискриминация порождающей ее АММ(P). Данная оценка может быть снижена в зависимости от величины отклонения вероятности перехода ЦМ из одного состояния в другое от значения 0,5 и от максимально допустимого отклонения данной вероятности относительно её теоретического значения. Допустимое отклонение вычисляется на основе областей определения частот перехода ЦМ из одного состояния в другое для ЭСМ, принадлежащих к априори заданным подклассам.

Предложена методика многопараметрической дискриминации реализаций ЦМ, заданных ЭСМ определённых подклассов, на базе заданного множества классифицирующих признаков.

ЛИТЕРАТУРА

1. *Кемени Дж., Снелл Дж.* Конечные цепи Маркова. М.: Наука, 1970. 272 с.
2. *Романовский В. И.* Дискретные цепи Маркова. М.; Л.: Гостехиздат, 1949. 436 с.
3. *Раскин Л. Г.* Анализ стохастических систем и элементы теории оптимального управления. М.: Сов. радио, 1976. 344 с.
4. *Бухараев Р. Г.* Основы теории вероятностных автоматов. М.: Наука, 1985. 287 с.

5. *Поспелов Д. А.* Вероятностные автоматы. М.: Энергия, 1970. 88 с.
6. *Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В.* Основы криптографии. М.: Гелиос АРВ, 2002. 480 с.
7. *Friedman W. F., Callimahos D.* Military crypto analyze. Part I. V. Z. Aegean Park Press, Laguna Hills CA, 1985. 356 p.
8. *Боровиков В. П.* Statistica: искусство анализа данных на компьютере. 2-е изд. СПб.: Питер, 2003. 700 с.
9. *Захаров В. М., Нурмеев Н. Н., Салимов Ф. И. и др.* К задаче дискриминантного анализа автоматных марковских моделей // Вестник КГТУ им. А. Н. Туполева. 2001. Т. 1. № 3. С. 37–39.
10. *Захаров В. М., Нурмеев Н. Н., Салимов Ф. И., Шалагин С. В.* Классификация стохастических эргодических матриц методами кластерного и дискриминантного анализа // Исследования по информатике. Казань: Отечество, 2000. С. 91–106.
11. *Ли И., Джадж Д., Зельнер А. М.* Оценивание параметров марковской модели по агрегированным временным рядам. М.: Статистика, 1977. 221 с.
12. *Ланкастер Л.* Теория матриц. М.: Наука, 1982. 272 с.
13. *Хинчин А. Я.* Понятие энтропии в теории вероятностей // Успехи матем. наук. 1953. № 3(55). С. 3–20.
14. *Сабитова А. Р., Шалагин С. В.* Дискриминантный анализ вероятностных моделей марковского типа // Наука. Технологии. Инновации: Материалы Всерос. конф. молодых ученых. Новосибирск: Изд-во НГТУ, 2007. С. 90–92.
15. *Сабитова А. Р., Шалагин С. В.* Многопараметрическая классификация марковских последовательностей // XV Туполевские чтения: Междунар. молодежная науч. конф. Казань: Изд-во КГТУ им. А. Н. Туполева, 2007. С. 78–79.
16. *Нурутдинова А. Р., Шалагин С. В.* Методика идентификации автоматных марковских моделей на основе порождаемых ими последовательностей // Вестник КГТУ им. А. Н. Туполева. 2010. № 1. С. 94–99.
17. *Вентцель Е. С.* Теория вероятностей. IV изд., стер. М.: Наука, 1969. 576 с.

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ДИСКРЕТНЫХ АВТОМАТОВ

DOI 10.17223/20710410/10/6

УДК 519.714 : 681.32

ПАРАЛЛЕЛЬНО-ПОСЛЕДОВАТЕЛЬНАЯ ДЕКОМПОЗИЦИЯ СИСТЕМЫ ЧАСТИЧНЫХ БУЛЕВЫХ ФУНКЦИЙ

Ю. В. Поттосин, Е. А. Шестаков

Объединенный институт проблем информатики НАН Беларуси, г. Минск, Беларусь

E-mail: pott@newman.bas-net.by

Рассматривается задача декомпозиции системы не полностью определенных булевых функций. Вводится понятие степени зависимости функции от некоторых её аргументов, и сложность функций, на которые разлагается заданная система, оценивается этим параметром. Предлагается метод параллельно-последовательной декомпозиции системы не полностью определенных булевых функций. Особенностью этого метода является то, что для него не нужно задавать подмножества аргументов компонент декомпозиции, требуемые в большинстве известных методов. Они определяются по ходу выполнения декомпозиции.

Ключевые слова: *система не полностью определенных булевых функций, декомпозиция.*

Введение

Задача декомпозиции булевых функций является одной из важных задач из области логического проектирования, что делает её объектом большого внимания со стороны многих исследователей. Как показывает обзор [1], на данную тему написано большое количество статей. В данной работе рассматривается задача декомпозиции системы булевых функций в следующей постановке: для заданной системы $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ не полностью определённых булевых функций $f_1, \ldots, f_m$ от переменных в $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ найти системы, возможно, тоже не полностью определённых булевых функций $\boldsymbol{g_1}(\boldsymbol{z^1})$, $\boldsymbol{g_2}(\boldsymbol{z^2})$ от переменных в $\boldsymbol{x}$ и $\boldsymbol{h}(\boldsymbol{g_1}, \boldsymbol{g_2})$, такие, что суперпозиция $\boldsymbol{h}(\boldsymbol{g_1}(\boldsymbol{z^1}), \boldsymbol{g_2}(\boldsymbol{z^2}))$ совпадает с функцией $\boldsymbol{f}(\boldsymbol{x})$ на области её определения.

Обычно при решении подобной задачи отыскивается суперпозиция $\boldsymbol{h}(\boldsymbol{g_1}(\boldsymbol{z^1}), \boldsymbol{g_2}(\boldsymbol{z^2}))$, в которой число булевых аргументов (компонент векторных аргументов) в каждой из систем $\boldsymbol{h}, \boldsymbol{g_1}$ и $\boldsymbol{g_2}$ должно быть меньше, чем в системе $\boldsymbol{f}$. В такой постановке задача декомпозиции не всегда имеет решение, что значительно усложняет применение методов декомпозиции при синтезе многоярусных комбинационных схем.

В настоящей работе предлагается оценивать сложность систем булевых функций $\boldsymbol{g_1}$ и $\boldsymbol{g_2}$ в искомой суперпозиции не числом их булевых аргументов, но степенью их зависимости от некоторых из этих аргументов. Это понятие будет определено ниже.

В этом случае поиск искомой суперпозиции осуществляется в три этапа. На первом этапе по исходной системе булевых функций $\boldsymbol{f}(\boldsymbol{x})$ строится тривиальная суперпозиция $\boldsymbol{h'}(g_1(\boldsymbol{z^1}), g_2(\boldsymbol{z^2}))$, где g_1 и g_2 — псевдобулевы функции с натуральными значениями от векторных переменных $\boldsymbol{z^1}$ и $\boldsymbol{z^2}$, но на этом этапе $g_1(\boldsymbol{z^1}) = g_2(\boldsymbol{z^2})$ и $\boldsymbol{z^1} = \boldsymbol{z^2} = \boldsymbol{x}$.

На втором этапе выполняется ряд последовательных итераций, представляющий собой чередование преобразований функций $g_1(\boldsymbol{z^1})$ и $g_2(\boldsymbol{z^2})$ и соответственно функции $\boldsymbol{h'}(g_1, g_2)$ с сохранением заданного отображения $\boldsymbol{f}(\boldsymbol{x})$. При каждой такой итерации удаляется одна компонента из векторной переменной $\boldsymbol{z^i}$ ($i \in \{1, 2\}$). Если на какой-либо итерации окажется невозможным преобразование суперпозиции, полученной на предыдущей итерации, с учетом установленных ограничений, то второй этап завершается.

На третьем этапе выполняется кодирование значений натуральных переменных и находится требуемая суперпозиция $\boldsymbol{h}(\boldsymbol{g_1}(\boldsymbol{z^1}), \boldsymbol{g_2}(\boldsymbol{z^2}))$.

Особенностью предлагаемой декомпозиции является то, что для её выполнения не нужно предварительно искать в $\boldsymbol{x}$ наборы аргументов $\boldsymbol{z^1}$ и $\boldsymbol{z^2}$. Их выбор осуществляется в ходе выполнения декомпозиции. Число булевых аргументов в системе $\boldsymbol{h}$ построенной декомпозиции будет удовлетворять заданным ограничениям. Число булевых аргументов в системах $\boldsymbol{g_1} = \boldsymbol{g_1}(\boldsymbol{z^1})$, $\boldsymbol{g_2} = \boldsymbol{g_2}(\boldsymbol{z^2})$ в общем случае может не уменьшиться по сравнению с исходной системой $\boldsymbol{f}$, однако степень зависимости этих систем по некоторым подмножествам аргументов будет меньше, чем зависимость по этим аргументам исходной системы.

1. Постановка задачи

Пусть система частичных булевых функций $\boldsymbol{f}(\boldsymbol{x})$ задается парой матриц $\boldsymbol{U}, \boldsymbol{V}$ размерности $l \times n$ и $l \times m$ соответственно. Матрица $\boldsymbol{U}$ является булевой. Её столбцы соответствуют переменным $x_1, x_2, \ldots, x_n$, а её элементы принимают значения из множества $\{0, 1\}$. Каждая строка с номером i, $1 \leqslant i \leqslant l$, матрицы $\boldsymbol{U}$ задает булев вектор [3], который является значением переменной $\boldsymbol{x}$. Множество векторов, представленное матрицей $\boldsymbol{U}$, задает область определения D_f системы $\boldsymbol{f}(\boldsymbol{x})$. Матрица $\boldsymbol{V}$ — троичная, её элементы принимают значения из множества $\{0, 1, -\}$. Строки матрицы $\boldsymbol{V}$ находятся во взаимно однозначном соответствии со строками матрицы $\boldsymbol{U}$. Столбцы матрицы $\boldsymbol{V}$ соответствуют функциям $f_1, f_2, \ldots, f_m$ заданной системы (переменным $y_1, y_2, \ldots, y_m$), а i-я строка матрицы $\boldsymbol{V}$ представляет значения этих функций при значении векторной переменной, представляемом i-й строкой матрицы $\boldsymbol{U}$.

Пусть троичные векторы $\boldsymbol{y^*}$ и $\boldsymbol{y'^*}$ состоят из одинакового числа компонент. Будем писать $\boldsymbol{y^*} \geqslant \boldsymbol{y'^*}$, если значения всех компонент вектора $\boldsymbol{y^*}$, отличные от значения «$-$», совпадают с соответствующими компонентами вектора $\boldsymbol{y'^*}$.

Поскольку упомянутые выше векторные переменные $\boldsymbol{z^1}$ и $\boldsymbol{z^2}$ составлены из компонент векторной переменной $\boldsymbol{x}$, то компоненты их значений $\boldsymbol{z^{1*}}$ и $\boldsymbol{z^{2*}}$ совпадают с соответствующими компонентами вектора $\boldsymbol{x^*}$. В дальнейшем будем считать, что если задано значение $\boldsymbol{x^*}$ векторной переменной $\boldsymbol{x}$, то также заданы значения $\boldsymbol{z^{1*}}$ и $\boldsymbol{z^{2*}}$ векторных переменных $\boldsymbol{z^1}$ и $\boldsymbol{z^2}$ соответственно.

Говорим также, что суперпозиция систем частичных булевых функций $\boldsymbol{h}(\boldsymbol{g_1}(\boldsymbol{z^1}), \boldsymbol{g_2}(\boldsymbol{z^2}))$ *реализует* систему частичных булевых функций $\boldsymbol{f}(\boldsymbol{x})$, и пишем $\boldsymbol{f}(\boldsymbol{x}) \geqslant \boldsymbol{h}(\boldsymbol{g_1}(\boldsymbol{z^1}), \boldsymbol{g_2}(\boldsymbol{z^2}))$, если для любого значения $\boldsymbol{x^*} \in D_f$ выполняется $\boldsymbol{f}(\boldsymbol{x^*}) \geqslant \geqslant \boldsymbol{h}(\boldsymbol{g_1}(\boldsymbol{z^{1*}}), \boldsymbol{g_2}(\boldsymbol{z^{2*}}))$.

Говорят, что система $\boldsymbol{f}(\boldsymbol{x})$ не зависит от булевой переменной x_i, $1 \leqslant i \leqslant n$, если $\boldsymbol{f}(\boldsymbol{x^*}) = \boldsymbol{f}(\boldsymbol{x'^*})$ для любой пары значений $\boldsymbol{x^*}, \boldsymbol{x'^*}$ из множества D_f, отличающихся друг от друга только по i-й компоненте (значением переменной x_i). В противном случае, т. е. когда существует хотя бы одна такая пара значений $\boldsymbol{x^*}, \boldsymbol{x'^*}$ из множества D_f, отличающихся друг от друга только по i-й компоненте, что $\boldsymbol{f}(\boldsymbol{x^*}) \neq \boldsymbol{f}(\boldsymbol{x'^*})$, говорят, что система $\boldsymbol{f}(\boldsymbol{x})$ зависит от x_i. В этом случае нетрудно найти все такие различные

пары, для которых выполняется это неравенство. Число этих пар обозначим через $d(\boldsymbol{f}, x_i)$ и назовем это число *степенью зависимости* системы $\boldsymbol{f}(\boldsymbol{x})$ от переменной x_i. Очевидно, $d(\boldsymbol{f}, x_i) = 0$, если система $\boldsymbol{f}(\boldsymbol{x})$ не зависит от переменной x_i.

Пусть у систем $\boldsymbol{f}(\boldsymbol{x})$ и $\boldsymbol{f}'(\boldsymbol{x})$ совпадают области определения, т. е. $D_f = D_{f'}$, и каждая из этих систем зависит от переменной x_i. Будем говорить, что система $\boldsymbol{f}(\boldsymbol{x})$ в меньшей степени зависит от переменной x_i, чем система $\boldsymbol{f}'(\boldsymbol{x})$, если $d(\boldsymbol{f}, x_i) < d(\boldsymbol{f}', x_i)$.

Определим и степень зависимости $d(\boldsymbol{f}, X')$ системы $\boldsymbol{f}(\boldsymbol{x})$ от подмножества X' множества $X = \{x_1, \ldots, x_n\}$ булевых переменных ($X' \subset X$) как $d(\boldsymbol{f}, X') = \sum\limits_{x \in X'} d(\boldsymbol{f}, x)$. Ясно, что если система $\boldsymbol{f}(\boldsymbol{x})$ не зависит от булевых переменных, входящих в подмножество X', то $d(\boldsymbol{f}, X') = 0$.

В настоящей работе рассматривается следующая задача декомпозиции.

Для заданной системы $\boldsymbol{f}(\boldsymbol{x})$ необходимо найти суперпозицию $\boldsymbol{h}(\boldsymbol{g_1}, \boldsymbol{g_2})$, $\boldsymbol{g_1} = \boldsymbol{g_1}(\boldsymbol{z^1})$, $\boldsymbol{g_2} = \boldsymbol{g_2}(\boldsymbol{z^2})$, такую, что выполняются следующие условия:

1) $\boldsymbol{h}(\boldsymbol{g_1}(\boldsymbol{z^1}), \boldsymbol{g_2}(\boldsymbol{z^2})) \leqslant \boldsymbol{f}(\boldsymbol{x})$;
2) для заданных целых положительных чисел h_1 и h_2 при $n > h_1 + h_2 > m$ число булевых компонент в векторных переменных $\boldsymbol{g_1}$ и $\boldsymbol{g_2}$ не превышает соответственно h_1 и h_2;
3) $d(\boldsymbol{g_1}, Z_1) < d(\boldsymbol{f}, Z_1)$, $d(\boldsymbol{g_2}, Z_2) < d(\boldsymbol{f}, Z_2)$, где $Z_1 \subseteq X$ и $Z_2 \subseteq X$ — множества переменных, представляющих компоненты векторных переменных $\boldsymbol{z^1}$ и $\boldsymbol{z^2}$ соответственно ($Z_1 \neq \varnothing$, $Z_2 \neq \varnothing$).

2. Метод декомпозиции

Поиск декомпозиции системы $\boldsymbol{f}(\boldsymbol{x})$ осуществляется в три этапа.

Э т а п 1. *Формирование тривиальной декомпозиции.* По исходной системе булевых функций $\boldsymbol{f}(\boldsymbol{x})$ строится тривиальная суперпозиция $\boldsymbol{h}'(g_1, g_2)$, где $g_1 = g_1(\boldsymbol{z^1})$, $g_2 = g_2(\boldsymbol{z^2})$ — псевдобулевы функции, $g_1(\boldsymbol{z^1}) = g_2(\boldsymbol{z^2})$ и $\boldsymbol{z^1} = \boldsymbol{z^2} = \boldsymbol{x}$.

Пусть исходная система булевых функций $\boldsymbol{f}(\boldsymbol{x})$ задается матрицами $\boldsymbol{U}, \boldsymbol{V}$ с номерами строк 1, 2, …, l. Общая область определения функций $g_1(\boldsymbol{z^1})$ и $g_2(\boldsymbol{z^2})$ совпадает с областью определения исходной системы D_f, а их областью значений является множество $\{1, 2, \ldots, l\}$. Пусть перечень значений векторной переменной $\boldsymbol{z^1}$ представлен матрицей $\boldsymbol{U_1}$, а то же самое для $\boldsymbol{z^2}$ – матрицей $\boldsymbol{U_2}$. Значением функции $g_i(\boldsymbol{z^i})$ при некотором значении $\boldsymbol{z^i}$ является номер строки матрицы $\boldsymbol{U_i}$ ($i \in \{1, 2\}$). Так что функция $g_i(\boldsymbol{z^i})$ вполне задается матрицей $\boldsymbol{U_i}$. На данном этапе $\boldsymbol{U_1} = \boldsymbol{U_2} = \boldsymbol{U}$.

Область определения функции $\boldsymbol{h}'$ состоит из пар значений её аргументов $(1, 1)$, $(2, 2)$, …, (l, l). На паре значений своих аргументов (j, j) функция $\boldsymbol{h}'$ принимает значение $\boldsymbol{y^*} = \boldsymbol{h}'(j, j)$, которое задаётся j-й строкой матрицы $\boldsymbol{V}$.

Функция $\boldsymbol{h}'(g_1, g_2)$ задается квадратной матрицей размерности $l \times l$. Строкам этой матрицы приписаны значения $1, 2, \ldots, l$ переменной g_1, а столбцам — значения $1, 2, \ldots, l$ переменной g_2. Все элементы этой матрицы, расположенные не на её главной диагонали, являются неопределёнными, т. е. имеют значение «$-$». Элемент матрицы, расположенный на пересечении строки и столбца с одним и тем же номером j ($j = 1, 2, \ldots, l$), равен j-й строке матрицы $\boldsymbol{V}$.

Пример 1. Пусть исходная система булевых функций $\boldsymbol{f}(\boldsymbol{x})$, где $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5)$, $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) = (y_1, y_2)$, задана матрицами:

$$\boldsymbol{U} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \end{array}, \quad \boldsymbol{V} = \begin{array}{c} \begin{matrix} y_1 & y_2 \end{matrix} \\ \left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \end{array}.$$

На наборах значений переменных, не представленных в матрице $\boldsymbol{U}$, значения функций не определены. Справа от матриц $\boldsymbol{U}, \boldsymbol{V}$ указаны номера их строк. Построим по этой функции тривиальную суперпозицию $\boldsymbol{h'}(g_1, g_2)$, где $g_1 = g_1(\boldsymbol{z^1})$, $g_2 = g_2(\boldsymbol{z^2})$ и $\boldsymbol{z^1} = \boldsymbol{z^2} = \boldsymbol{x}$. Функции $g_1(\boldsymbol{z^1})$ и $g_2(\boldsymbol{z^2})$ в ней задаются соответственно матрицами $\boldsymbol{U_1}$ и $\boldsymbol{U_2}$, которые в данном случае совпадают с $\boldsymbol{U}$, а функция $\boldsymbol{h'}(g_1, g_2)$ — матрицей $\boldsymbol{W}$:

$$\boldsymbol{U_1} = \boldsymbol{U_2} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \end{array}, \quad \boldsymbol{W} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \end{matrix} \\ \left[\begin{array}{ccccccccccccc} 00 & - & - & - & - & - & - & - & - & - & - & - & - \\ - & 00 & - & - & - & - & - & - & - & - & - & - & - \\ - & - & 10 & - & - & - & - & - & - & - & - & - & - \\ - & - & - & 01 & - & - & - & - & - & - & - & - & - \\ - & - & - & - & 11 & - & - & - & - & - & - & - & - \\ - & - & - & - & - & 00 & - & - & - & - & - & - & - \\ - & - & - & - & - & - & 01 & - & - & - & - & - & - \\ - & - & - & - & - & - & - & 11 & - & - & - & - & - \\ - & - & - & - & - & - & - & - & 00 & - & - & - & - \\ - & - & - & - & - & - & - & - & - & 10 & - & - & - \\ - & - & - & - & - & - & - & - & - & - & 11 & - & - \\ - & - & - & - & - & - & - & - & - & - & - & 11 & - \\ - & - & - & - & - & - & - & - & - & - & - & - & 00 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \end{array}.$$

Э т а п 2. *Последовательность преобразований суперпозиции* $\boldsymbol{h'}(g_1, g_2)$. На этом этапе выполняется ряд итераций. В процессе каждой итерации осуществляется преобразование матрицы $\boldsymbol{U_i}$, представляющей псевдобулеву функцию $g_i(\boldsymbol{z^i})$ $(i \in \{1, 2\})$, и соответствующее преобразование матрицы $\boldsymbol{W}$, представляющей функцию $\boldsymbol{h'}(g_1, g_2)$. Последовательность итераций представляет собой чередование этих действий для $\boldsymbol{U_1}$ и $\boldsymbol{U_2}$.

Матрица $\boldsymbol{U_i}$ преобразуется так, чтобы снизилась степень зависимости $d(g_i, x_k)$ представляемой ею функции $g_i(\boldsymbol{z^i})$ от одного из булевых аргументов x_k. Чтобы описать упомянутые преобразования, введём некоторые понятия.

Строки и столбцы матрицы $\boldsymbol{W}$ можно рассматривать как троичные векторы. Две строки матрицы $\boldsymbol{W}$ *несовместимы*, если они представляют ортогональные троичные

векторы. В противном случае они *совместимы*. Под ортогональностью троичных векторов понимается наличие противоположных значений (0 и 1) их одноимённых компонент [3]. Такое же отношение совместимости определяется и на множестве столбцов матрицы $\boldsymbol{W}$.

Пусть булева переменная x_k является компонентой векторной переменной $\boldsymbol{z^1}$. На множестве $\{1, 2, \ldots, l_1\}$ номеров строк матрицы $\boldsymbol{U_1}$ определим разбиение $\alpha(\boldsymbol{U_1}, x_k)$ так, что r и s $(1 \leqslant r \leqslant l_1, 1 \leqslant s \leqslant l_1, r \neq s)$ попадают в один блок этого разбиения, если и только если r-я и s-я строки матрицы $\boldsymbol{U_1}$ отличаются только значением компоненты x_k, а r-я и s-я строки матрицы $\boldsymbol{W}$ совместимы. Если переменная x_k является компонентой векторной переменной $\boldsymbol{z^2}$, то аналогично определим разбиение $\alpha(\boldsymbol{U_2}, x_k)$ на множестве $\{1, 2, \ldots, l_2\}$ номеров строк матрицы $\boldsymbol{U_2}$: r и s $(1 \leqslant r \leqslant l_2, 1 \leqslant s \leqslant l_2, r \neq s)$ попадают в один блок этого разбиения, если и только если r-я и s-я строки матрицы $\boldsymbol{U_2}$ отличаются только значением компоненты x_k, а r-й и s-й столбцы матрицы $\boldsymbol{W}$ совместимы.

Преобразования матриц $\boldsymbol{U_1}$ и $\boldsymbol{W}$ выполняются по разбиению $\alpha(\boldsymbol{U_1}, x_k)$. Заметим, что любое разбиение $\alpha(\boldsymbol{U_i}, x_k)$ $(i \in \{1, 2\})$ по любой переменной x_k может содержать только одноэлементные и двухэлементные блоки. В матрице $\boldsymbol{U_1}$ каждую пару строк, номера которых содержатся в одном блоке разбиения $\alpha(\boldsymbol{U_1}, x_k)$, заменяем одной, в которой компоненте x_k приписываем значение «$-$». Это же значение приписываем тем остальным строкам, для которых это изменение не нарушает взаимной ортогональности всех строк матрицы $\boldsymbol{U_1}$. Если столбец x_k оказался при этом состоящим только из элементов со значением «$-$», то он удаляется из матрицы $\boldsymbol{U_1}$. В этом случае $d(g_1, x_k) = 0$. Преобразование матрицы $\boldsymbol{U_1}$ сопровождается преобразованием матрицы $\boldsymbol{W}$, в результате которого каждая пара строк, номера которых содержатся в одном блоке разбиения $\alpha(\boldsymbol{U_1}, x_k)$, заменяется одной строкой, являющейся результатом пересечения троичных векторов, представляемых данными строками. При этом сохраняется взаимно однозначное соответствие между строками матрицы $\boldsymbol{U_1}$ и строками матрицы $\boldsymbol{W}$, а также сохраняется согласование их нумерации.

Результатом операции пересечения двух пересекающихся троичных векторов одинаковой размерности является троичный вектор, формируемый следующим образом [4]. Если обе одноимённые компоненты исходных векторов имеют значение «$-$», то это значение приписывается соответствующей компоненте вектора-результата. Если хотя бы одна из одноимённых компонент исходных векторов имеет значение 0 или 1 (в пересекающихся векторах они не могут иметь противоположные значения), то это значение приписывается соответствующей компоненте вектора-результата.

Матрица $\boldsymbol{U_2}$ преобразуется совершенно аналогично, только прежнюю переменную x_k для этой цели использовать нельзя. Во всей последовательности итераций каждая переменная может использоваться не более одного раза. При сопровождающем преобразовании матрицы $\boldsymbol{W}$ всё, что говорилось относительно её строк, делается с её столбцами.

Пример 2. Для матрицы $\boldsymbol{U}$ из примера 1 найдем следующие разбиения:
$\alpha(\boldsymbol{U}, x_1) = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6, 12\}, \{7, 13\}, \{8\}, \{9\}, \{10\}, \{11\}\}$;
$\alpha(\boldsymbol{U}, x_2) = \{\{1\}, \{2, 6\}, \{3, 8\}, \{4\}, \{5, 9\}, \{7\}, \{10\}, \{11\}, \{12\}, \{13\}\}$;
$\alpha(\boldsymbol{U}, x_3) = \{\{1, 3\}, \{2, 5\}, \{4\}, \{6, 9\}, \{7\}, \{8\}, \{10\}, \{11\}, \{12\}, \{13\}\}$;
$\alpha(\boldsymbol{U}, x_4) = \{\{1, 2\}, \{3, 5\}, \{4\}, \{6\}, \{7\}, \{8, 9\}, \{10, 12\}, \{11, 13\}\}$;
$\alpha(\boldsymbol{U}, x_5) = \{\{1\}, \{2\}, \{3, 4\}, \{5\}, \{6, 7\}, \{8\}, \{9\}, \{10, 11\}, \{12, 13\}\}$.

Выбрав x_3 для преобразования матрицы $\boldsymbol{U_1}$ и затем x_1 для преобразования матрицы $\boldsymbol{U_2}$, получим

$$\boldsymbol{U_1} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_4 & x_5 \end{matrix} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}, \quad \boldsymbol{U_2} = \begin{array}{c} \begin{matrix} x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array}, \quad \boldsymbol{W} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{matrix} \\ \left[\begin{array}{ccccccccccc} 0 & 0 & - & 1 & 0 & - & - & - & - & - & - \\ - & 0 & 0 & - & - & 1 & 1 & - & - & - & - \\ - & - & - & 0 & 1 & - & - & - & - & - & - \\ - & - & - & - & - & 0 & 0 & - & - & 0 & 0 \\ - & - & - & - & - & - & 0 & 1 & - & - & - \\ - & - & - & - & - & - & - & 1 & 1 & - & - \\ - & - & - & - & - & - & - & - & - & 1 & 0 \\ - & - & - & - & - & - & - & - & - & - & 1 \\ - & - & - & - & - & 1 & 1 & - & - & - & - \\ - & - & - & - & - & - & 0 & 0 & - & - & - \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}.$$

Заметим, что некоторые столбцы, совместимые в матрице $\boldsymbol{W}$, могут оказаться несовместимыми после совмещения её строк. По условию решаемой задачи декомпозиции аргументы $\boldsymbol{g_1}$ и $\boldsymbol{g_2}$ системы булевых функций $\boldsymbol{h}(\boldsymbol{g_1}, \boldsymbol{g_2})$ искомой декомпозиции должны соответственно состоять не более чем из h_1 и h_2 булевых переменных. Значения векторных переменных $\boldsymbol{g_1}$ и $\boldsymbol{g_2}$ определяются при кодировании строк и столбцов матрицы $\boldsymbol{W}$ на этапе 3, и чтобы удовлетворить этому условию, любое множество взаимно ортогональных строк матрицы $\boldsymbol{W}$ должно содержать не более 2^{h_1} строк, а любое множество взаимно ортогональных столбцов — не более 2^{h_2} столбцов. Необходимым условием для этого является следующее.

Условие C. Каждый столбец матрицы $\boldsymbol{W}$ должен содержать не более 2^{h_1}, а каждая её строка — не более 2^{h_2} взаимно ортогональных значений векторной переменной $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$.

Это условие должно проверяться перед каждым преобразованием матрицы $\boldsymbol{W}$. Булеву переменную x_k, при которой преобразование матрицы $\boldsymbol{W}$ по разбиению $\alpha(\boldsymbol{U_i}, x_k)$ $(i \in \{1, 2\})$ не нарушает условие C, назовем *корректной*. При выборе аргумента x_k, по которому осуществляется данное преобразование, предпочтение отдаётся корректному аргументу. Если таких аргументов несколько, то выбирается тот, который приведёт к увеличению несовместимых пар столбцов в матрице $\boldsymbol{W}$ на минимальную величину.

Пусть корректные аргументы отсутствуют. Это означает, что для каждого аргумента не выполняется условие C. Пусть аргумент x_k не является корректным. Преобразуем разбиение $\alpha(\boldsymbol{U_i}, x_k)$ посредством деления некоторых его двухэлементных блоков так, чтобы условие C выполнялось. В этом случае степень зависимости $d(g_i, x_k)$ функции g_i от аргумента x_k останется не равной нулю. Величина $d(g_i, x_k)$ будет тем меньше, чем меньше будет выполнено делений двухэлементных блоков разбиения $\alpha(\boldsymbol{U_i}, x_k)$. Если выполнение условия C будет достигнуто лишь в том случае, когда все блоки разбиения $\alpha(\boldsymbol{U_i}, x_k)$ будут разделены до одноэлементных, то тогда $d(g_i, x_k)$ уменьшить не удастся. Назовём в этом случае аргумент x_k устойчивым. В противном случае аргумент x_k будем называть неустойчивым. Если все аргументы функции g_i устойчивы, то её преобразование прекращается. Если среди аргументов имеются неустойчивые, то выбираем из них тот, степень зависимости от которого функции g_i будет наименьшей.

Пример 3. Матрица $\boldsymbol{U_1}$, полученная в примере 2, преобразуется на основе разбиения $\alpha(\boldsymbol{U_1}, x_2) = \{\{1, 6\}, \{2, 4\}, \{3\}, \{5\}, \{7\}, \{8\}, \{9\}, \{10\}\}$, в результате чего при

сохранении прежнего значения матрицы $\boldsymbol{U}$ получаем следующее:

$$\boldsymbol{U_1} = \begin{array}{c} \begin{matrix} x_1 & x_4 & x_5 \end{matrix} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix}, \quad \boldsymbol{U_2} = \begin{array}{c} \begin{matrix} x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{matrix}, \quad \boldsymbol{W} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{matrix} \\ \left[\begin{array}{ccccccccccc} 00 & - & 10 & - & - & - & - & 11 & - & - & - \\ - & 00 & - & - & 11 & 00 & - & - & 00 & - & - \\ - & - & - & 01 & - & - & - & - & - & - & - \\ - & - & - & - & - & - & 01 & - & - & - & - \\ - & - & - & - & - & - & - & - & - & 10 & - \\ - & - & - & - & - & - & - & - & - & - & 11 \\ - & - & - & - & - & 11 & - & - & - & - & - \\ - & - & - & - & - & - & 00 & - & - & - & - \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix}.$$

Пусть в искомой суперпозиции $\boldsymbol{h}(\boldsymbol{g_1}, \boldsymbol{g_2})$ число компонент в векторной переменной $\boldsymbol{g_1}$ должно быть не более единицы, а в переменной $\boldsymbol{g_2}$ — не более двух, т. е. $h_1 = 1$ и $h_2 = 2$. Взяв любое из разбиений $\alpha(\boldsymbol{U_2}, x_k), k = 4, 5$, в качестве основы для совмещения столбцов матрицы $\boldsymbol{W}$, мы не получим окончательное значение матрицы $\boldsymbol{W}$ без нарушения указанных ограничений, т. е. ни одна из переменных x_4 и x_5 не является корректной (переменные x_2 и x_3 не могут быть взяты в данном случае, поскольку на их основе проводились преобразования матрицы $\boldsymbol{U_1}$). Тогда выбираем подходящее совмещение столбцов матрицы $\boldsymbol{W}$ — третьего с четвертым и десятого с одиннадцатым — и получаем следующие матрицы при $d(\boldsymbol{g_2}, x_5) = 1$:

$$\boldsymbol{U_1} = \begin{array}{c} \begin{matrix} x_1 & x_4 & x_5 \end{matrix} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix}, \boldsymbol{U_2} = \begin{array}{c} \begin{matrix} x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & - \\ 0 & 0 & 1 & - \\ 0 & 1 & 0 & - \\ 0 & 1 & 1 & - \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & - \\ 1 & 1 & 1 & - \\ 1 & 0 & 0 & - \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{matrix}, \boldsymbol{W} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \left[\begin{array}{ccccccccc} 00 & - & 10 & - & - & - & 11 & - & - \\ - & 00 & - & 11 & 00 & - & - & 00 & - \\ - & - & 01 & - & - & - & - & - & - \\ - & - & - & - & - & 01 & - & - & - \\ - & - & - & - & - & - & - & - & 10 \\ - & - & - & - & - & - & - & - & 11 \\ - & - & - & - & 11 & - & - & - & - \\ - & - & - & - & - & 00 & - & - & - \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix}.$$

Э т а п 3. *Получение суперпозиции* $\boldsymbol{h}(\boldsymbol{g_1}, \boldsymbol{g_2})$, *удовлетворяющей условиям задачи.* Чтобы получить векторные булевы функции $\boldsymbol{g_1}$ и $\boldsymbol{g_2}$, надо закодировать строки и столбцы матрицы $\boldsymbol{W}$ двоичными векторами, причем совместимые строки, так же как совместимые столбцы, могут иметь одинаковые коды. Те же коды присваиваются соответствующим значениям многозначных переменных g_1 и g_2. Если функция g_i задана матрицей $\boldsymbol{U_i}$ $(i = 1, 2)$, то векторная функция $\boldsymbol{g_i}$ задается следующим образом: значением функции $\boldsymbol{g_i}$ на наборе переменных, представляемым строкой матрицы $\boldsymbol{U_i}$, является код соответствующего значения функции g_i. Удобно преобразовать матрицу $\boldsymbol{W}$ в матрицу $\boldsymbol{H}$, представляющую искомую функцию $\boldsymbol{h}(\boldsymbol{g_1}, \boldsymbol{g_2})$. Для этого находится разбиение множества строк матрицы $\boldsymbol{W}$ на совместимые подмножества, т. е. на подмножества, где любые две строки совместимы, и все строки из одного подмножества заменяются одной строкой, представляющей пересечение соответствующих троичных векторов. Та же операция проделывается затем со столбцами матрицы $\boldsymbol{W}$.

При совмещении строк для получения матрицы $\boldsymbol{H}$, число строк в которой не должно превышать 2^{h_1}, следует стараться не потерять возможности сократить число столбцов в ней до 2^{h_2}. Выделим 2^{h_1} строк в матрице $\boldsymbol{W}$, которые должны войти в различные совместимые множества. Около них группируем остальные строки, формируя таким образом искомые совместимые подмножества. В результате получаем матрицу $\boldsymbol{W'}$ с 2^{h_1} строками, где каждая строка является пересечением всех строк, принадлежащих соответственному совместимому подмножеству.

Текущая ситуация в этом процессе характеризуется двумя множествами: $S = \{S_1, S_2, \ldots, S_{2^{h_1}}\}$ — совокупность совместимых подмножеств строк матрицы $\boldsymbol{W}$ (в начале процесса это одноэлементные множества) и $R = \{r_1, r_2, \ldots, r_t\}$ — множество остальных строк матрицы $\boldsymbol{W}$. Из множества R последовательно выбираются строки и переносятся в определенные подмножества из множества S. Процесс заканчивается, когда множество R оказывается пустым.

Строки матриц $\boldsymbol{W}$ и $\boldsymbol{W'}$ рассматриваются как секционированные троичные векторы [3]. Длина секции равна числу компонент векторной переменной $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$. Выбор пары (S_i, r_j) $(S_i \in S, r_j \in R)$ для включения строки r_j в множество S_i, где строка r_j должна быть совместима со всеми строками из S_i, осуществляется по следующим двум критериям: 1) минимум увеличения количества взаимно ортогональных секций в получаемой строке матрицы $\boldsymbol{W'}$; 2) минимум увеличения количества пар ортогональных секций в этой строке.

Матрица $\boldsymbol{W'}$ преобразуется в матрицу $\boldsymbol{H}$ путем аналогичного совмещения столбцов.

Пример 4. Используем данные, полученные в примере 3. Как было сказано, для получения искомой декомпозиции необходимо матрицу $\boldsymbol{W}$ преобразовать в матрицу $\boldsymbol{H}$ таким образом, чтобы матрица $\boldsymbol{H}$ имела не более двух строк и не более четырех столбцов.

В начальной ситуации имеем $S = \{\{1\}, \{3\}\}$ и $R = \{2, 4, 5, 6, 7, 8\}$, при этом матрица $\boldsymbol{W'}$ имеет следующий вид:

$$\boldsymbol{W'} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \left[\begin{matrix} 00 & - & 10 & - & - & - & 11 & - & - \\ - & - & 01 & - & - & - & - & - & - \end{matrix}\right] \end{array} \begin{matrix} 1 \\ 3 \end{matrix} .$$

Согласно приведённым выше критериям, следующим шагом является совмещение строк 1 и 2 матрицы $\boldsymbol{W}$, и матрица $\boldsymbol{W'}$ примет следующий вид, где справа покажем множества номеров строк матрицы $\boldsymbol{W}$, из которых образованы соответствующие строки матрицы $\boldsymbol{W'}$:

$$\boldsymbol{W'} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \left[\begin{matrix} 00 & 00 & 10 & 11 & 00 & - & 11 & 00 & - \\ - & - & 01 & - & - & - & - & - & - \end{matrix}\right] \end{array} \begin{matrix} 1,2 \\ 3 \end{matrix} .$$

Следующим значением матрицы $\boldsymbol{W}$ является

$$\boldsymbol{W'} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \left[\begin{matrix} 00 & 00 & 10 & 11 & 00 & - & 11 & 00 & - \\ - & - & 01 & - & - & 01 & - & - & - \end{matrix}\right] \end{array} \begin{matrix} 1,2 \\ 3,4 \end{matrix} .$$

Продолжая этот процесс, получим следующую последовательность значений матрицы $\boldsymbol{W'}$:

$$\boldsymbol{W'}=\begin{matrix}\begin{matrix}1&2&3&4&5&6&7&8&9\end{matrix}\\\begin{bmatrix}00&00&10&11&00&-&11&00&11\\-&-&01&-&-&01&-&-&-\end{bmatrix}\end{matrix}\begin{matrix}1,2,6\\3,4\end{matrix};\quad \boldsymbol{W'}=\begin{matrix}\begin{matrix}1&2&3&4&5&6&7&8&9\end{matrix}\\\begin{bmatrix}00&00&10&11&00&00&11&00&11\\-&-&01&-&-&01&-&-&-\end{bmatrix}\end{matrix}\begin{matrix}1,2,6,8\\3,4\end{matrix};$$

$$\boldsymbol{W'}=\begin{matrix}\begin{matrix}1&2&3&4&5&6&7&8&9\end{matrix}\\\begin{bmatrix}00&00&10&11&00&00&11&00&11\\-&-&01&-&-&01&-&-&10\end{bmatrix}\end{matrix}\begin{matrix}1,2,6,8\\3,4,5\end{matrix};\quad \boldsymbol{W'}=\begin{matrix}\begin{matrix}1&2&3&4&5&6&7&8&9\end{matrix}\\\begin{bmatrix}00&00&10&11&00&00&11&00&11\\-&-&01&-&11&01&-&-&10\end{bmatrix}\end{matrix}\begin{matrix}1,2,6,8\\3,4,5,7\end{matrix}.$$

В последней матрице совместимы группы столбцов: $\{3\}, \{1,2,5\}, \{6,8\}, \{4,7,9\}$. После совмещения этих столбцов получим матрицу $\boldsymbol{H}$:

$$\boldsymbol{H}=\begin{matrix}\begin{matrix}1&2&3&4\end{matrix}\\\begin{bmatrix}10&00&00&11\\01&11&01&10\end{bmatrix}\end{matrix}\begin{matrix}1\\2\end{matrix}.$$

Присвоив код 0 первой строке этой матрицы и код 1 второй строке, получим функцию $\boldsymbol{v}=\boldsymbol{g_1}(x_1,x_4,x_5)$ и, присвоив коды 00, 01, 10 и 11 соответственно столбцам 1, 2, 3 и 4, получим функцию $\boldsymbol{w}=\boldsymbol{g_2}(x_2,x_3,x_4,x_5)$, задаваемые следующими матрицами:

$$\boldsymbol{U_1}=\begin{matrix}\begin{matrix}x_1&x_4&x_5\end{matrix}\\\begin{bmatrix}0&0&0\\0&1&0\\0&0&1\\0&1&1\\1&0&0\\1&0&1\\1&1&0\\1&1&1\end{bmatrix}\end{matrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix};\ \boldsymbol{V_1}=\begin{matrix}\begin{matrix}v\end{matrix}\\\begin{bmatrix}0\\0\\1\\1\\1\\0\\1\\0\end{bmatrix}\end{matrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix};\ \boldsymbol{U_2}=\begin{matrix}\begin{matrix}x_2&x_3&x_4&x_5\end{matrix}\\\begin{bmatrix}0&0&0&-\\0&0&1&-\\0&1&0&-\\0&1&1&-\\1&0&1&0\\1&0&1&1\\1&1&0&-\\1&1&1&-\\1&0&0&-\end{bmatrix}\end{matrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\\9\end{matrix};\ \boldsymbol{V_2}=\begin{matrix}\begin{matrix}w_1&w_2\end{matrix}\\\begin{bmatrix}0&1\\0&1\\0&0\\1&1\\0&1\\1&0\\1&1\\1&0\\1&1\end{bmatrix}\end{matrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\\9\end{matrix}.$$

Искомая функция $\boldsymbol{h}(\boldsymbol{g_1},\boldsymbol{g_2})=\boldsymbol{h}(\boldsymbol{v},\boldsymbol{w})$ представится следующими матрицами:

$$\boldsymbol{U}=\begin{matrix}\begin{matrix}v&w_1&w_2\end{matrix}\\\begin{bmatrix}0&0&0\\0&0&1\\0&1&0\\0&1&1\\1&0&0\\1&0&1\\1&1&0\\1&1&1\end{bmatrix}\end{matrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix};\ \boldsymbol{V}=\begin{matrix}\begin{matrix}y_1&y_2\end{matrix}\\\begin{bmatrix}1&0\\0&0\\0&0\\1&1\\0&1\\1&0\\0&1\\1&0\end{bmatrix}\end{matrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix}.$$

ЛИТЕРАТУРА

1. *Perkowski M. A., Grygiel S.* A Survey of Literature on Functional Decomposition. Version IV (Technical Report). Portland, USA: Portland State University, Department of Electrical Engineering, 1995. 188 p.
2. *Поттосин Ю. В., Шестаков Е. А.* Табличные методы декомпозиции систем полностью определенных булевых функций. Минск: Белорус. наука, 2006. 327 с.
3. *Закревский А. Д., Поттосин Ю. В., Черемисинова Л. Д.* Логические основы проектирования дискретных устройств. М.: Физматлит, 2007. 592 с.
4. *Закревский А. Д.* Логический синтез каскадных схем. М.: Наука, 1981. 414 с.

DOI 10.17223/20710410/10/7

УДК 519.713.1

РЕГУЛЯРНАЯ ФОРМА СПЕЦИФИКАЦИИ ДЕТЕРМИНИРОВАННЫХ АВТОМАТОВ В ЯЗЫКЕ L

А. Н. Чеботарев

Институт кибернетики НАН Украины, г. Киев, Украина

E-mail: ancheb@gmail.com

Исследуются некоторые формы представления спецификаций детерминированных циклических автоматов в языке L. Использование таких форм при построении спецификации уменьшает возможность допущения ошибок.

Ключевые слова: *язык спецификации, конечный автомат, циклический автомат, сверхслово.*

Введение

Формальная спецификация автомата — это совокупность утверждений, которые определяют требования к его функционированию, т. е. его темпоральные свойства. Эти утверждения формулируются в виде формул формального языка, для чего обычно используются логические языки, такие, как темпоральные логики, исчисление предикатов первого порядка и др. В работе в качестве такого языка рассматривается простой логический язык L, являющийся подмножеством языка одноместных предикатов первого порядка. Процесс построения формальной спецификации автомата начинается с осмысления всех требований к его функционированию с последующим формулированием их в виде формул языка L. Для спецификаций, рассчитанных на синтез, т. е. на использование автоматической процедуры преобразования спецификации в процедурное представление алгоритма функционирования автомата, необходимо обеспечить их полноту, состоящую в том, что должны быть приведены все ограничения на функционирование автомата. Поскольку построение спецификации — это неформальный процесс, осуществляемый человеком, построенная спецификация может содержать ошибки, устранение которых после синтеза и на последующих этапах превращается в очень сложную задачу. Поэтому необходимо так организовать процесс построения спецификации, чтобы уменьшить вероятность допущения в ней ошибки. Этого можно достичь двумя путями: 1) придерживаться строгих правил написания спецификации в четко определенной форме; 2) использовать средства формальной верификации, т. е. проверки наличия важных свойств спецификации. Обычно используется как первый, так и второй пути уменьшения количества ошибок. Настоящая работа посвящена исследованию некоторых форм представления спецификаций детерминированных автоматов и определению их свойств, которые можно просто проверять в процессе построения спецификации. Написание спецификации в рассматриваемых формах существенно уменьшает вероятность допущения ошибок, а также возможность того, что некоторые ограничения на функционирование автомата не будут учтены.

1. Язык спецификации

Язык L [1] представляет собой фрагмент логики предикатов первого порядка с одноместными предикатами и фиксированной областью интерпретации, в качестве которой выступает множество $\mathbb{Z}$ целых чисел (моментов времени). Спецификация в языке L имеет вид формулы $\forall t F(t)$, где $F(t)$ — формула с одной переменной t, построенная

с помощью логических связок из атомарных формул (атомов) вида $p(t+j)$, где p — одноместный предикатный символ, t — переменная, принимающая значения из множества $\mathbb{Z}$, а j — целочисленная константа, называемая *рангом атома*. Разность между максимальным и минимальным рангами атомов в формуле $F(t)$ называется её *глубиной*. Поскольку $F(t)$ интерпретируется на множестве целых чисел, то для произвольного целого j имеет место эквивалентность $\forall tF(t) \Leftrightarrow \forall tF(t+j)$, где $F(t+j)$ обозначает формулу, полученную из $F(t)$ путем увеличения на j рангов всех ее атомов. Таким образом, можно ограничиться рассмотрением только таких формул $F(t)$, у которых максимальный ранг атомов равен нулю.

При определении семантики языка L он рассматривается как формализм для задания множеств сверхслов в алфавите двоичных векторов, длина которых равна количеству предикатных символов в формуле.

Пусть Σ — конечный алфавит и $\mathbb{N}^+ = \{z \in \mathbb{Z} : z > 0\}$. Отображения $u\colon \mathbb{Z} \to \Sigma$ и $l\colon \mathbb{N}^+ \to \Sigma$ называются соответственно *двусторонним сверхсловом* (обозначается $\ldots u(-2)u(-1)u(0)u(1)u(2)\ldots$) и *сверхсловом* (обозначается $l(1)l(2)\ldots$) в алфавите Σ. Для двустороннего сверхслова u и $n \in \mathbb{Z}$ определим *n-суффикс* $u(n+1,\infty)$ как сверхслово $u(n+1)u(n+2)\ldots$

Пусть $\Omega = \{p_1, p_2, \ldots, p_k\}$ — множество всех предикатных символов, которые встречаются в формуле $F(t)$ (сигнатура формулы). Областью интерпретации языка L является множество $\mathbb{Z}$, и в формулах не интерпретированы только предикатные символы. Таким образом, интерпретация формулы $\forall tF(t)$ — это набор $<\pi_1, \pi_2, \ldots, \pi_k>$ определённых на $\mathbb{Z}$ одноместных предикатов, соответствующих всем предикатным символам из множества Ω. Каждый такой предикат π_i можно рассматривать как двустороннее сверхслово в алфавите $\{0,1\}$, а набор k таких предикатов — как двустороннее сверхслово в алфавите $\Sigma = \{0,1\}^k$. Мы не будем различать интерпретации и соответствующие двусторонние сверхслова в алфавите Σ, поэтому будем говорить об истинности или ложности формулы $F(t)$ в позиции τ ($\tau \in \mathbb{Z}$) двустороннего сверхслова u, имея в виду значения формулы $F(\tau)$ при интерпретации u. Смысл понятия глубины формулы состоит в том, что значение формулы $F(t)$ глубины r в позиции τ интерпретации u определяется отрезком $u(\tau - r, \tau)$ соответствующего двустороннего сверхслова u. Интерпретация, при которой формула $\forall tF(t)$ истинна, называется *моделью* для этой формулы. С каждой формулой $F = \forall tF(t)$ ассоциируется множество $M(F)$ всех моделей для нее. Каждая формула $F = \forall tF(t)$ определяет множество $W(F)$ сверхслов над Σ, а именно множество 0-суффиксов всех двусторонних сверхслов из $M(F)$.

Определим автоматную семантику языка L.

Определение 1. Конечный $(X{-}Y)$-автомат A — это четвёрка $<X, Y, Q, \chi>$, где X и Y — соответственно входной и выходной алфавиты; Q — конечное множество состояний; $\chi\colon Q \times X \times Y \to 2^Q$ — функция переходов. Иногда отношение переходов удобно задавать в виде функции переходов-выходов $\lambda\colon Q \times X \to 2^{Q \times Y}$.

Автомат A называется *детерминированным*, если для любых $x \in X$, $q \in Q$ верно $|\lambda(q,x)| \leqslant 1$; в противном случае он называется *недетерминированным*.

Определение 2. $(X{-}Y)$-автомат $A = <X, Y, Q, \chi>$ называется *квазидетерминированным*, если для всяких $q \in Q$, $x \in X$ и $y \in Y$ справедливо $|\chi(q,x,y)| \leqslant 1$.

Квазидетерминированный $(X{-}Y)$-автомат удобно рассматривать как детерминированный частичный автомат без выходов $A = <\Sigma, Q, \delta>$ с входным алфавитом $\Sigma = X \times Y$, с множеством состояний Q и с функцией переходов δ, являющейся отображением из $Q \times \Sigma$ в Q. Будем называть его *Σ-автоматом* A.

Определение 3. Σ-автомат $A = <\Sigma, Q, \delta>$ называется *циклическим*, если для каждого $q \in Q$ существуют такие $q_1, q_2 \in Q$ и $\sigma_1, \sigma_2 \in \Sigma$, что $q_1 = \delta(q, \sigma_1)$ и $q = \delta(q_2, \sigma_2)$.

Циклический Σ-автомат можно охарактеризовать в терминах допустимых сверхслов.

Определение 4. Сверхслово $l = \sigma_1\sigma_2\ldots$ *допустимо в состоянии* q Σ-автомата A, если существует такое сверхслово состояний $q_0q_1q_2\ldots$, где $q_0 = q$, что для любого $i = 0, 1, 2, \ldots$ имеет место $\delta(q_i, \sigma_{i+1}) = q_{i+1}$. Сверхслово l *допустимо для автомата* A, если оно допустимо хотя бы в одном из его состояний.

Обозначим $W(A)$ множество всех сверхслов, допустимых для автомата A. Формула F *специфицирует* автомат A, если $W(A) = W(F)$. Как показано в [2], класс автоматов, специфицируемых формулами языка L, совпадает с классом циклических автоматов с конечной памятью [3]. Рассмотрим способ построения автомата $A(F)$, специфицируемого формулой $F = \forall tF(t))$ [4].

Пусть $\Omega = \{p_1, p_2, \ldots, p_k\}$ — сигнатура формулы $F(t)$, r — её глубина и $S_\Omega = \{0, 1\}^k$. Последовательность $s_0s_1\ldots s_r$ векторов $s_0, s_1, \ldots, s_r$ из S_Ω назовем *состоянием* глубины r, а множество $Q(r, \Omega)$ всех таких последовательностей — *пространством состояний* глубины r для формулы $F(t)$. Формулу $F(t)$ будем рассматривать как пропозициональную формулу от переменных $p_1(t)$, …, $p_k(t)$, $p_1(t-1)$, …, $p_k(t-1)$, …, $p_1(t-r)$, …, $p_k(t-r)$. Если компоненты вектора s_i в состоянии $q = s_0s_1\ldots s_r$ рассматривать как истинностные значения соответствующих атомов ранга $i-r$ при определенном упорядочении множества Ω, то можно говорить о значении формулы $F(t)$ на состоянии q.

На множестве $Q(r, \Omega)$ определим отношение N *непосредственного следования* так, что за каждым состоянием $q = s_0s_1\ldots s_r$ непосредственно следуют 2^k состояний вида $s_1\ldots s_rs$, где $s \in S_\Omega$. Множество всех состояний, которые непосредственно следуют за q, будем обозначать $N(q)$.

При использовании языка L для спецификации автоматов предикатные символы ставятся в соответствие входным и выходным двоичным каналам специфицируемого автомата. Поэтому множество предикатных символов Ω разбивается на два класса: входные и выходные, которые обозначаются соответственно U и W. Определим входной алфавит X и выходной алфавит Y автомата $A(F)$ как множества всех двоичных векторов соответственно длины $|U|$ и $|W|$. Каждый вектор из S_Ω можно рассматривать как пару $<x, y>$, где $x \in X$, $y \in Y$, поэтому наряду с обозначением $s_0\ldots s_r$ для состояния глубины r будем использовать обозначение $(<x_0, y_0>\ldots<x_r, y_r>)$.

Построим вспомогательный автомат $A'(F) =< X, Y, Q', \chi_{A'} >$. Множество состояний Q' — это все те состояния из $Q(r, \Omega)$, на которых $F(t)$ истинна. Функцию переходов этого автомата определим следующим образом. Пусть $q \in Q'$, $x \in X$ и $y \in Y$, тогда $\chi_{A'}(q, x, y)$ — это множество всех тех состояний $(<x_0, y_0>\ldots<x_r, y_r>)$ из $N(q) \cap Q'$, у которых $x_r = x$ и $y_r = y$. Если таких состояний нет, то значение $\chi_{A'}(q, x, y)$ не определено. Автомат $A(F)$ — это максимальный циклический подавтомат автомата $A'(F)$.

2. Свойства спецификаций детерминированных автоматов

Рассмотрим представление формулы $F(t)$ в пространстве состояний соответствующей глубины. Будем говорить, что формула $F(t)$ истинна на области пространства состояний, если она истинна хотя бы на одном состоянии этой области, и формула ложна на области, если она ложна на всех ее состояниях. Если q — состояние пространства состояний, то $N(q)$ называется *областью переходов* для состояния q. Часть области переходов, соответствующая символу x входного алфавита X, т. е. все такие состояния

($<x_0, y_0> \ldots <x_r, y_r>$) из $N(q)$, у которых $x_r = x$, называется *областью перехода* под действием символа x.

Пусть $F(t)$ — формула глубины r с сигнатурой $\Omega = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$, где $x_1, \ldots, x_n$ — входные переменные, а $y_1, \ldots, y_m$ — выходные. Область перехода в пространстве состояний $Q(r, \Omega)$ задается формулой, представляющей собой конституенту единицы от переменных $x_1(t)$, …, $x_n(t)$, $x_1(t-1)$, …, $x_n(t-1)$, $y_1(t-1)$, …, $y_m(t-1)$, …, $x_1(t-r)$, …, $x_n(t-r)$, $y_1(t-r)$, …, $y_m(t-r)$. Заметим, что формула ложна на области переходов, задаваемой конституентой единицы l, тогда и только тогда, когда в её разложении по указанным переменным остаточная функция, соответствующая l, равна нулю.

Свойства спецификаций, соответствующие детерминированности и всюду определённости автомата, определяются свойствами областей перехода минимальной формы представления спецификации [5]. Минимальная форма спецификации $F = \forall t F(t)$ — это такая эквивалентная спецификация $\forall t \min(F(t))$ автомата $A(F)$, что множество состояний пространства $Q(r, \Omega)$, на которых истинна формула $\min(F(t))$, совпадает с множеством состояний циклического автомата $A(F)$ в этом пространстве. В [5] показано, как по формуле $F(t)$ построить формулу $\min(F(t))$.

Утверждение 1. Формула $\forall t F(t)$ специфицирует детерминированный автомат тогда и только тогда, когда в каждой области перехода пространства состояний формула $\min(F(t))$ истинна не более чем на одном состоянии.

Утверждение 2. Формула $\forall t F(t)$ специфицирует всюду определённый автомат тогда и только тогда, когда для каждой области переходов пространства состояний, на которой истинна $\min(F(t))$, она истинна на всех входящих в нее областях перехода.

В терминах свойств булевых функций, задаваемых формулами $F(t)$ и $\min(F(t))$, эти утверждения можно переформулировать следующим образом.

Утверждение 3. Формула $\forall t F(t)$ специфицирует детерминированный автомат тогда и только тогда, когда каждая остаточная функция в разложении функции $\min(F(t))$ по переменным $x_1(t)$, …, $x_n(t)$, $x_1(t-1)$, …, $x_n(t-1)$, $y_1(t-1)$, …, $y_m(t-1)$, …, $x_1(t-r)$, …, $x_n(t-r)$, $y_1(t-r)$, …, $y_m(t-r)$ либо равна нулю, либо представляет собой конституенту единицы от переменных $y_1(t), \ldots, y_m(t)$.

Очевидно, что такое разложение представляет собой совершенную ДНФ.

Утверждение 4. Формула $\forall t F(t)$ специфицирует всюду определённый автомат, если и только если из того, что остаточная функция $f_i(t)$ в разложении $\min(F(t))$ по переменным $x_1(t-1)$, …, $x_n(t-1)$, $y_1(t-1)$, …, $y_m(t-1)$, …, $x_1(t-r)$, …, $x_n(t-r)$, $y_1(t-r)$, …, $y_m(t-r)$ не равна нулю, следует, что все остаточные функции в разложении $f_i(t)$ по переменным $x_1(t), \ldots, x_n(t)$ также не равны нулю.

Пусть $F(x_1, \ldots, x_n, y)$ — булева функция от переменных $x_1, \ldots, x_n, y$. $\exists$-проекцией функции $F(x_1, \ldots, x_n, y)$ на $\{x_1, \ldots, x_n\}$ называется функция $\exists y F(x_1, \ldots, x_n, y)$, которая задается формулой $F(x_1, \ldots, x_n, 0) \vee F(x_1, \ldots, x_n, 1)$. Аналогично, $\exists$-проекцией функции $F(x_1, \ldots, x_n, y_1, \ldots, y_m)$ на множество переменных $\{x_1, \ldots, x_n\}$ называется функция $\exists y_1 \ldots \exists y_m F(x_1, \ldots, x_n, y_1, \ldots, y_m)$.

Теорема 1. Функция $F(x_1, \ldots, x_n, y_1, \ldots, y_m)$ может быть представлена в виде

$$F = \bigwedge_{i=1}^{m} (y_i \leftrightarrow f_i(x_1, \ldots, x_n))$$

тогда и только тогда, когда каждая остаточная функция в разложении функции F по переменным $x_1, \ldots, x_n$ является конституентой единицы от переменных $y_1, \ldots, y_m$.

Доказательство. *Необходимость.* Пусть $F(x_1,\dots,x_n,y_1,\dots,y_m)$ имеет вид $(y_1\leftrightarrow f_1)\&\dots\&(y_m\leftrightarrow f_m)=(y_1f_1\vee\neg y_1\neg f_1)\&\dots\&(y_mf_m\vee\neg y_m\neg f_m)=y_1\dots y_mf_1\dots f_m\vee\vee y_1\dots\neg y_mf_1\dots\neg f_m\vee\dots\vee\neg y_1\dots\neg y_m\neg f_1\dots\neg f_m$, где $f_1,\dots,f_m$ — функции от переменных $x_1,\dots,x_n$. На каждом наборе значений переменных $x_1,\dots,x_n$ истинно только одно из 2^m произведений вида $\tilde{f}_1\dots\tilde{f}_m$, где $\tilde{f}_i\in\{f_i,\neg f_i\}$. Отсюда следует, что каждая остаточная функция в разложении $F(x_1,\dots,x_n,y_1,\dots,y_m)$ по переменным $x_1,\dots,x_n$ является конституентой единицы от переменных $y_1,\dots,y_m$.

Достаточность. Пусть $F(x_1,\dots,x_n,y_1,\dots,y_m)$ удовлетворяет условиям теоремы. Эти условия можно переформулировать в виде следующих свойств функции F.

1. Совершенная ДНФ функции $F(x_1,\dots,x_n,y_1,\dots,y_m)$ содержит ровно 2^n конституент единицы.

2. Каждая конституента единицы от переменных $x_1,\dots,x_n$ является составной частью одной и только одной из конституент единицы функции $F(x_1,\dots,x_n,y_1,\dots,y_m)$.

Этими же свойствами обладает и каждая $\exists$-проекция функции $F(x_1,\dots,x_n,y_1,\dots,y_m)$ на множество переменных $\{x_1,\dots,x_n,y_i\}$ $(i=1,2,\dots,m)$. Отсюда следует, что каждая проекция функции $F(x_1,\dots,x_n,y_1,\dots,y_m)$ на $\{x_1,\dots,x_n,y_i\}$ имеет вид $f_i(x_1,\dots,x_n)y_i\vee\neg f_i(x_1,\dots,x_n)\neg y_i=y_i\leftrightarrow f_i(x_1,\dots,x_n)$. Поскольку рассматриваемые проекции обладают указанными свойствами, то и произведение всех этих проекций также обладает ими. Кроме того, $F(x_1,\dots,x_n,y_1,\dots,y_m)$ является импликантой каждой такой проекции, а следовательно, и импликантой их произведения.

Лемма 1. Если $f(x_1,\dots,x_n)$ — импликанта функции $f'(x_1,\dots,x_n)$ и совершенные ДНФ этих функций имеют одинаковое количество конституент единицы, то $f(x_1,\dots,x_n)=f'(x_1,\dots,x_n)$.

Из этой леммы следует, что произведение всех m проекций функции $F(x_1,\dots,x_n,y_1,\dots,y_m)$ совпадает с этой функцией, что завершает вторую часть доказательства. ■

Теорема 2. Любой всюду определённый, детерминированный автомат с конечной памятью может быть специфицирован в виде

$$\forall t\bigwedge_{i=1}^{m}(y_i(t)\leftrightarrow f_i(t)),$$

где $y_1,\dots,y_m$ — все выходные предикатные символы спецификации и $f_i(t)$ не зависит от атомов $y_1(t),\dots,y_m(t)$.

Доказательство. Пусть $F(t)$ в спецификации $\forall tF(t)$ такова, что все остаточные функции в её разложении по переменным $x_1(t),\dots,x_n(t),x_1(t-1),\dots,x_n(t-1),y_1(t-1),\dots,y_m(t-1),\dots,x_1(t-r),\dots,x_n(t-r),y_1(t-r),\dots,y_m(t-r)$ равны конституентам единицы от переменных $y_1(t),\dots,y_m(t)$. В силу теоремы 1 она может быть представлена в виде $\bigwedge\limits_{i=1}^{m}(y_i(t)\leftrightarrow f_i(t))$, где формулы $f_i(t)$ не зависят от атомов $y_1(t),\dots,y_m(t)$. Таким образом, для доказательства теоремы 2 достаточно показать, что минимальная форма спецификации детерминированного, всюду определённого автомата может быть эквивалентно преобразована в формулу, удовлетворяющую указанному выше условию. Такое преобразование состоит в том, чтобы формулу $\min(F(t))$ сделать истинной на тех областях переходов, на которых она ложна, и таким образом, чтобы в каждой области перехода она была истинна не более чем на одном состоянии. Это осуществляется путем замены в разложении функции $\min(F(t))$ по переменным $x_1(t),\dots,x_n(t),x_1(t-1),\dots,x_n(t-1),y_1(t-1),\dots,y_m(t-1),\dots,x_1(t-r),\dots,x_n(t-r),y_1(t-r),$

$\ldots$, $y_m(t-r)$ всех нулевых остаточных функций ненулевыми. Эквивалентность этого преобразования можно обеспечить, если для замены использовать имеющиеся в этом разложении ненулевые остаточные функции. ■

Такая спецификация — это совокупность утверждений, определяющих необходимые и достаточные условия истинности каждого выходного предиката в произвольный момент времени t, причем эти условия не зависят от значений выходных предикатов в этот момент времени.

Теорема 3. Любой детерминированный автомат с конечной памятью может быть специфицирован в виде

$$\forall t \bigwedge_{i=1}^{m} (y_i(t) \to g_i(t)) \& (\neg y_i(t) \to h_i(t)),$$

где $y_1, \ldots, y_m$ — все выходные предикатные символы спецификации; $g_i(t)$, $h_i(t)$ не зависят от атомов $y_1(t), \ldots, y_m(t)$.

Доказательство. Пусть автомат задан формулой $\forall t F(t)$ с выходными предикатными символами $y_1, \ldots, y_m$. Согласно утверждению 3, каждая остаточная функция в разложении функции $\min(F(t))$ по всем переменным, отличным от $y_1(t)$, $\ldots$, $y_m(t)$, либо равна нулю, либо представляет собой конституенту единицы от переменных $y_1(t)$, $\ldots$, $y_m(t)$. Рассмотрим m $\exists$-проекций функции $\min(F(t))$ на все атомы, кроме выходных атомов нулевого ранга, отличных от $y_i(t)$ $(i = 1, 2, \ldots, m)$. Аналогично тому, как это сделано в доказательстве теоремы 1, можно показать, что произведение всех этих проекций совпадает с функцией $\min(F(t))$. Каждая такая проекция может быть представлена в виде $y_i(t)g_i(t) \vee \neg y_i(t)h_i(t)$, что эквивалентно $(y_i(t) \to g_i(t)) \& (\neg y_i(t) \to h_i(t))$, где $g_i(t)$, $h_i(t)$ не зависят от атомов $y_1(t)$, $\ldots$, $y_m(t)$. Так как формулы $\forall t \min(F(t))$ и $\forall t F(t)$ специфицируют один и тот же автомат, то теорема доказана. ■

Несложно показать, что $g_i(t) \& h_i(t) \equiv 0$ $(i = 1, 2, \ldots, m)$.

3. Необходимые и достаточные условия

Возникает вопрос, как построить формулы $g_i(t)$ и $h_i(t)$ в спецификации детерминированного автомата? Для ответа на него проанализируем понятия необходимого и достаточного условий.

Если $\forall t(a(t) \to b(t))$, то будем говорить, что $b(t)$ — необходимое условие истинности $a(t)$, а $a(t)$ — достаточное условие истинности $b(t)$. Таким образом, формулы $g_i(t)$ и $h_i(t)$ в спецификации детерминированного автомата — это необходимые условия соответственно истинности и ложности предиката $y_i(t)$ в момент t. Очевидно, что если $A(t)$ и $B(t)$ — необходимые условия истинности предиката $y(t)$, то $A(t) \& B(t)$ — также необходимое условие. Аналогично, если $A(t)$ и $B(t)$ — достаточные условия, то $A(t) \vee B(t)$ — также достаточное условие. В силу эквивалентности формул $y(t) \to A(t)$ и $\neg A(t) \to \neg y(t)$ отрицание необходимого условия истинности предиката $y(t)$ в момент t является достаточным условием ложности этого предиката. Поэтому возможны различные эквивалентные формы представления спецификации автомата с использованием необходимых или достаточных условий истинности (ложности) выходных предикатов. Однако если в формулах использовать произвольные необходимые или достаточные условия, то полученная таким образом спецификация может не специфицировать требуемый автомат. Поэтому следует уточнить вид этих условий.

Пусть $A_1, A_2, \ldots, A_n$ — все возможные неэквивалентные формулы определённой глубины, которые задают необходимые условия истинности некоторого предиката,

тогда формулу $A_1\&A_2\&\ldots\&A_n$ будем называть соответствующим *полным необходимым условием*. Аналогично, если формулы $A_1, A_2, \ldots, A_n$ задают достаточные условия, то формулу $A_1 \vee A_2 \vee \cdots \vee A_n$ будем называть *полным достаточным условием*. Заметим, что полные необходимые и полные достаточные условия представляют собой необходимые и достаточные условия. Использование в спецификации вида $\forall t \bigwedge\limits_{i=1}^{m} (y_i(t) \to g_i(t))\&(\neg y_i(t) \to h_i(t))$ в качестве $g_i(t)$ и $h_i(t)$ соответствующих необходимых и достаточных условий гарантирует правильность спецификации.

Рассмотрим пример построения спецификации автомата с одной входной двоичной переменной x и одной выходной двоичной переменной y. Требования к функционированию автомата описаны следующим образом. Значение y в момент t равняется единице тогда и только тогда, когда во входной последовательности до момента t включительно отсутствуют нулевые значения переменной x. Как можно видеть из этого описания, поведение автомата детерминировано и всюду определено. Необходимое и достаточное условие истинности $y(t)$ имеет вид $y(t-1)\&x(t)$. Таким образом, спецификация этого автомата может быть записана в виде $\forall t(y(t) \leftrightarrow y(t-1)\&x(t))$.

Еще один пример автомата с такими же входным и выходным алфавитами. Автомат функционирует следующим образом. Если x в момент t равен единице, то y также в этот момент равен единице. Если x изменился в нуль, то со следующего момента после этого изменения y также будет равен нулю (если x в этот момент равен нулю) до того момента, когда x изменится в единицу. Кроме того, в следующий момент времени после того, как x изменится в единицу, он не может равняться нулю. Последнее утверждение определяет частичность автомата. В этом случае автомат удобно специфицировать как всюду определённый, дополнив эту спецификацию условием частичности. Так, спецификацию соответствующего всюду определённого автомата запишем как $\forall t(y(t) \leftrightarrow (x(t-1) \vee x(t)))$. Действительно, всякий раз, когда $y(t)$ равен единице, условие $x(t-1) \vee x(t)$ также равно единице, таким образом, это — необходимое условие. Кроме того, $x(t-1)$ и $x(t)$ — достаточные условия истинности $y(t)$, следовательно, $x(t-1) \vee x(t)$ — необходимое и достаточное условие. Условие частичности имеет вид $\forall t(\neg x(t-2)\&x(t-1) \to x(t))$.

Более сложный пример спецификации реактивных систем можно найти в [6].

Следует также рассмотреть спецификацию автомата в виде необходимых условий изменения (сохранения) значения выходного сигнала. Так, утверждение, что $A(t)$ — необходимое условие изменения значения y с 0 на 1, имеет вид $\forall t(\neg y(t-1)\&$ $\&y(t) \to A(t))$. Несложно показать, что спецификацию детерминированного автомата $\forall t \bigwedge\limits_{i=1}^{m} (y_i(t) \to g_i(t))\&(\neg y_i(t) \to h_i(t))$ можно представить в виде $\forall t \bigwedge\limits_{i=1}^{m} F_i(t)$, где $F_i(t)$ — это конъюнкция четырёх утверждений: $(y_i(t-1)\&y_i(t) \to A_i(t))$, $(y_i(t-1)\&\neg y_i(t) \to$ $\to B_i(t))$, $(\neg y_i(t-1)\&y_i(t) \to C_i(t))$, $(\neg y_i(t-1)\&\neg y_i(t) \to D_i(t))$. Чтобы перейти от этой формы спецификации к предыдущей, следует положить $g_i(t) = y_i(t-1)\&A_i(t) \vee$ $\vee\neg y_i(t-1)\&C_i(t)$, $h_i(t) = y_i(t-1)\&B_i(t) \vee \neg y_i(t-1)\&D_i(t)$.

Рассмотрим, как для первого примера построить такую спецификацию. Запишем спецификацию этого автомата в виде $\forall t(y(t) \to y(t-1)\&x(t))\&(\neg y(t) \to (\neg y(t-1) \vee$ $\vee\neg x(t)))$. Таким образом, $y(t-1)\&x(t) = y(t-1)\&A(t) \vee \neg y(t-1)\&C(t)$, а $(\neg y(t-1) \vee \neg x(t)) = y(t-1)\&B(t) \vee \neg y(t-1)\&D(t)$. Отсюда находим такие значения для $A(t), B(t), C(t)$ и $D(t)$: $A(t) = x(t)$, $B(t) = \neg x(t)$, $C(t) = 0$, $D(t) = 1$.

Если $g_i(t)$ и $h_i(t)$ — необходимые и достаточные условия истинности $y_i(t)$ и $\neg y_i(t)$ соответственно, то полученные таким образом формулы $y_i(t-1)\&A_i(t)$, $\neg y_i(t-1)\&C_i(t)$,

$y_i(t-1)\&B_i(t)$ и $\neg y_i(t-1)\&D_i(t)$ будут необходимыми и достаточными условиями истинности $y_i(t-1)\&y_i(t)$, $\neg y_i(t-1)\&y_i(t)$, и т. д. Действительно, учитывая, что $y_i(t) \leftrightarrow$ $\leftrightarrow g_i(t)$, а $\neg y_i(t) \leftrightarrow h_i(t)$, можно записать $y_i(t) \leftrightarrow (y_i(t-1)\&A_i(t) \vee \neg y_i(t-1)\&C_i(t))$, $\neg y_i(t) \leftrightarrow (y_i(t-1)\&B_i(t) \vee \neg y_i(t-1)\&D_i(t))$. Умножив обе части этих эквивалентностей на $y_i(t-1)$ и на $\neg y_i(t-1)$, получим $y_i(t-1)\&y_i(t) \leftrightarrow y_i(t-1)\&A_i(t)$, $\neg y_i(t-1)\&y_i(t) \leftrightarrow$ $\leftrightarrow \neg y_i(t-1)\&C_i(t)$ и т. п. Поскольку условия $A_i(t)$, $B_i(t)$ и т. д. проще, чем условия $g_i(t)$ и $h_i(t)$, то обычно такая форма спецификации более удобна для её построения.

Часто спецификация пишется при некоторых предположениях, ограничивающих множества допустимых символов входного или выходного алфавитов, например, недопустимость одновременной истинности двух выходных предикатов. В этом случае формулы, характеризующие все такие предположения, должны быть включены в спецификацию.

Заключение

В работе показано, что спецификацию любого детерминированного автомата можно строить в стандартной форме, использующей необходимые и достаточные условия истинности и ложности каждого выходного предиката. Обычно из неформального описания требований к функционированию специфицируемого автомата можно заключить, является ли он детерминированным или вполне определённым. Это позволяет выбрать форму спецификации, удовлетворяющую формальным требованиям. Так, при спецификации детерминированного автомата в виде конъюнкции формул $(y_i(t) \to A_i(t))\&(\neg y_i(t) \to B_i(t))$, ассоциируемых с каждым выходным предикатом y_i, должно выполняться равенство $A_i(t)\&B_i(t) = 0$. Причем в формулах $A_i(t)$ и $B_i(t)$ не должны встречаться выходные атомы нулевого ранга. Равенство $A_i(t) \vee B_i(t) = 1$ свидетельствует о полной определённости специфицируемого автомата. Заметим, что невыполнение этого равенства не всегда говорит о частичности автомата. Как показано во втором примере, спецификацию частичного автомата удобно записывать в виде конъюнкции $F_1\&F_2$, где F_1 — спецификация всюду определённого автомата, а F_2 — спецификация условия частичности. Условия частичности — это требования к функционированию среды, с которой взаимодействует автомат. Если среду рассматривать как детерминированный автомат, то условия частичности можно записать в виде $\forall t \bigwedge\limits_{i=1}^{m} (x_i(t) \to \alpha_i(t))\&(\neg x_i(t) \to \beta_i(t))$, где x_i — входные предикатные символы, а формулы $\alpha_i(t)$ и $\beta_i(t)$ не зависят от атомов нулевого ранга.

Использование стандартной формы представления спецификации, удовлетворяющей определённым, легко проверяемым требованиям, облегчает её написание и уменьшает возможность допущения ошибок. Разделение спецификации автомата на две части, соответствующие собственно автомату и среде, с которой он взаимодействует, также упрощает процесс написания спецификации, а впоследствии и оптимизацию автомата, синтезированного по спецификации.

ЛИТЕРАТУРА

1. *Чеботарев А. Н.* Об одном подходе к функциональной спецификации автоматных систем. I // Кибернетика и системный анализ. 1993. № 3. С. 31–42.
2. *Чеботарев А. Н.* Синтез недетерминированного автомата по его логической спецификации. I // Кибернетика и системный анализ. 1995. № 6. С. 115–127.
3. *Гилл А.* Введение в теорию конечных автоматов. М.: Наука, 1966. 227 с.

4. *Чеботарев А. Н.* Синтез алгоритма по его логической спецификации // Управляющие системы и машины. 2004. № 5. С. 53–60.
5. *Чеботарев А. Н., Куривчак О. И.* Аппроксимация множеств сверхслов формулами языка L // Кибернетика и системный анализ. 2007. № 6. С. 18–26.
6. *Чеботарев А. Н., Алистратов А. В.* Построение логической спецификации реактивного алгоритма // Проблемы программирования. 2002. № 1–2. С. 154–160.

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/10/8
УДК 519.857

НОВЫЕ АЛГОРИТМЫ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСА

В. И. Струченков

Московский государственный институт радиотехники, электроники и автоматики, г. Москва, Россия

E-mail: str1942@mail.ru

Предложены новые алгоритмы решения задачи оптимального распределения ресурса, использующие множества Парето и двусторонние прогностические оценки оптимума, получаемые по методу ветвей и границ.

Ключевые слова: *оптимальное распределение ресурса, множество Парето, метод ветвей и границ.*

1. Постановка задачи

Рассмотрим задачу: найти минимум (или максимум) суммы

$$\sum_{i=1}^{n} g_i(x_i) \tag{1}$$

при ограничениях

$$\sum_{i=1}^{n} f_i(x_i) \leqslant R,\ x_i \in X_i,\ i = 1, 2, \ldots, n, \tag{2}$$

где X_i — конечные множества; $f_i(x_i) \geqslant 0$; $g_i(x_i) \geqslant 0$ и $R > 0$. Целочисленность функций $f_i(x_i)$ и $g_i(x_i)$ необязательна. Предполагается, что множество (2) допустимых решений не пусто. В таком виде могут быть записаны различные задачи, сводящиеся к распределению заданного ресурса R между n потребителями. В задаче на максимум функции $f_i(x_i)$ характеризуют ресурс, а $g_i(x_i)$ — эффективность его использования [1]. При $f_i(x_i) = p_i x_i$ и $g_i(x_i) = c_i x_i$ получается задача оптимальной загрузки транспортного средства предметами, веса которых равны p_i, стоимости — c_i, а количества — x_i ($x_i = 0, 1, 2, \ldots$) [1]; если при этом $x_i \in \{0, 1\}$, то получается известная задача о ранце. В задаче на минимум функции $f_i(x_i)$ — ресурс, а $g_i(x_i)$ — затраты. Например, $f_i(x_i)$ — ресурс времени, а $g_i(x_i)$ — затраты на осуществление некоторого проекта. Или (как при проектировании оптимальной защиты поверхности [2]) $f_i(x_i)$ — допускаемый ущерб от неполной защиты i-го элемента поверхности, а $g_i(x_i)$ — затраты на его защиту.

2. Новые алгоритмы

Для решения задачи оптимального распределения ресурса предложено несколько алгоритмов [1 – 3], в частности, в [2] излагается следующий алгоритм, который является обобщением алгоритма Немхаузера — Ульмана, предназначенного для решения простейшей задачи о ранце [4].

Далее рассматривается задача (1), (2) на максимум. Пусть $Q_m = \{(x_1, \ldots, x_m) : (x_1, \ldots, x_m) \in X_1 \times \ldots \times X_m,\ \sum\limits_{i=1}^{m} f_i(x_i) \leqslant R\}$. Каждому вектору $x = (x_1, \ldots, x_m) \in Q_m$ сопоставляется пара чисел $(F_m(x), G_m(x))$ вида $F_m(x) = \sum\limits_{i=1}^{m} f_i(x_i),\ G_m(x) = \sum\limits_{i=1}^{m} g_i(x_i)$. Из полученной совокупности пар, образующей мультимножество (так как могут существовать такие $x, y \in Q_m$, что $(F_m(x), G_m(x)) = (F_m(y), G_m(y))$, выделяется его носитель N и на нём вводится следующее отношение частичного порядка:

$$(a, b) \succ (c, d) \Leftrightarrow (a \leqslant c\ \&\ b \geqslant d).$$

Подмножество Парето в N составляют пары, максимальные относительно этого порядка, также называемые паретовскими точками. Алгоритм состоит в последовательном построении множеств Парето $S_0, S_1, \ldots, S_n$, где $S_0 = \{(0, 0)\}$, а $S_m = \{(F_m(x), G_m(x)) : x = (x_1, \ldots, x_m) \in Q_m, (F_{m-1}(x_1, \ldots, x_{m-1}), G_{m-1}(x_1, \ldots, x_{m-1})) \in S_{m-1}$ и для любой пары $(F_m(y), G_m(y)) \in S_m$ либо $F_m(y) < F_m(x)$ и $G_m(y) < G_m(x)$, либо $F_m(x) < F_m(y)$ и $G_m(x) < G_m(y)\}$.

Технически, для каждой пары $(F_{m-1}(x), G_{m-1}(x)) \in S_{m-1}$ и таких значений $a \in X_m$, что $(x \circ a) \in Q_m$, где $\circ$ — операция конкатенации, строятся векторы длины m и соответствующие им пары $(F_{m-1}(x) + f_m(a), G_{m-1}(x) + g_m(a))$. Если для очередной полученной пары в множестве S_m нет равных или больших нее, то она также добавляется в S_m, при этом все пары, которые меньше нее, исключаются из S_m. Решением задачи (1), (2) на максимум является вектор $x \in Q_n$, которому соответствует пара $(F_n(x), G_n(x)) \in S_n$ с максимальным значением $G_n(x)$.

Для решения задач большой размерности также актуальна дополнительная отбраковка тех векторов, которым соответствуют бесперспективные паретовские точки. Предлагаемый в данной работе алгоритм (алгоритм отбраковки паретовских точек) делает это, используя идеи метода ветвей и границ.

Далее рассматривается задача (1), (2) на минимум. Обозначим затраты, соответствующие некоторому начальному приближенному решению, через E (в методе ветвей и границ они называются рекордом), а их нижнюю границу через H. Пусть теперь построено частичное решение $x' = (x'_1, \ldots, x'_m) \in Q_m$ и соответствующая ему точка $(F_m(x'), G_m(x'))$ принадлежит S_m, т. е. является паретовской. Для этой точки рассмотрим задачу $(P_m) : \sum\limits_{i=m+1}^{n} f_i(x_i) \leqslant R - F_m(x'_1, \ldots, x'_m),\ \sum\limits_{i=m+1}^{n} g_i(x_i) \longrightarrow \min$. Пусть E_m — затраты ($\sum\limits_{i=m+1}^{n} g_i(x_i)$), соответствующие некоторому приближенному решению z задачи (P_m), а H_m — нижняя граница затрат в решении (P_m). Тогда условие отбраковки этой точки, ввиду ее очевидной бесперспективности, примет вид $G_m(x') + H_m \geqslant E$. Если же $G_m(x') + E_m < E$, то новое значение рекорда равно $G_m(x') + E_m$. Соответственно изменяется и вектор, задающий рекорд. Если $E_m = H_m$, то это значит, что приближенное решение z задачи (P_m) является точным, и её дальнейшее рассмотрение не требуется. Если при этом $G_m(x') + E_m < E$, то вектор $(x' \circ z)$ запоминается и хранится до тех пор, пока не будет получено значение рекорда, меньшее чем $G_m(x') + E_m$. Если «рекорд устоит», то соответствующее ему решение и является оптимальным. Если при некотором m не останется ни одной паретовской точки, то рекорд является решением.

При каждом m можно корректировать нижнюю границу, вычисляя значение $h = \min(G_m(x) + H_m)$, где минимум берётся по всем оставшимся паретовским точкам из S_m, и заменяя H на h при $h > H$. Если не требуется точное решение, то счёт

прекращается при $E - H < \varepsilon H$, где ε определяется требуемой точностью решения задачи. В этом случае рекорд E и соответствующий ему вектор x дают приближённое решение задачи (1), (2). Заметим, что оценки H_m и E_m нужно получать для каждой паретовской точки для $m = 1, 2, \ldots, n-1$. Эти оценки, как правило, получаются решением вспомогательной задачи. Для эффективности алгоритма в целом существенное значение имеют и качество этих оценок (их близость к оптимуму), и быстродействие алгоритма решения вспомогательной задачи. В задачах большой размерности число паретовских точек может быть велико, и время, затрачиваемое на дополнительные вычисления, может сводить к нулю эффект от дополнительной отбраковки паретовских точек.

3. Построение начального приближения и двусторонних оценок

Начальное приближение строится следующим образом: элементы x_i^j для $i = 1, 2, \ldots, n$ пронумеруем так, чтобы выполнялось условие $f_i^1 < f_i^2 < \ldots < f_i^{k_i}$. Здесь x_i^j для $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, k_i$ — элементы конечных множеств X_i; $k_i = |X_i|$; $f_i^j = f_i(x_i^j)$ и $g_i^j = g_i(x_i^j)$ — соответствующие значения ресурса и затрат (в задаче на минимум). Другими словами, для каждого i значения x_i^j упорядочиваем по возрастанию ресурса; соответственно затраты должны строго убывать.

Получим: $g_i^1 > g_i^2 > \ldots > g_i^{k_i}$. Если условие монотонности нарушается, то соответствующая точка $(f_i(x_i^j), g_i(x_i^j))$ исключается из дальнейшего рассмотрения как бесперспективная (непаретовская) ещё на этапе предварительной обработки данных [5, с. 239].

Откладывая f_i^j по оси абсцисс, а g_i^j по оси ординат, получаем последовательность точек, которая определяет для каждого i строго монотонно убывающую кусочно-линейную функцию. Будем считать, что такие функции построены для всех i. Множество их графиков (ломаных линий) обозначим через L_1. Те из ломаных, которые не являются выпуклыми, заменим их выпуклыми оболочками. Полученное множество ломаных линий $w_i(z_i)$, $i = 1, 2, \ldots, n$, обозначим через L_2. Поскольку при замене невыпуклой ломаной её выпуклой оболочкой уменьшается число вершин, введём новые обозначения: z_i вместо f_i и w_i вместо g_i.

В результате получаем оценочную задачу: найти минимум суммы

$$\sum_{i=1}^{n} w_i(z_i) \tag{3}$$

при ограничениях

$$\sum_{i=1}^{n} z_i(x_i) \leqslant R,\ x_i \in X_i,\ i = 1, 2, \ldots, n. \tag{4}$$

Крайние точки ломаных из L_2 совпадают с крайними точками соответствующих ломаных из L_1. Абсолютные величины угловых коэффициентов звеньев ломаных $w_i(z_i)$ будем называть уклонами. В силу выпуклости для каждой ломаной из L_2 последовательность уклонов строго монотонно убывающая. Абсциссы концов звеньев ломаных из L_2 обозначим через b_i^j, их множества — через B_i, а уклоны звеньев — через u_i^j.

Рассматриваем непрерывную задачу: на каждой ломаной из L_2 найти точку с абсциссой z_i^* и ординатой w_i^* так, чтобы сумма абсцисс не превосходила заданный ресурс R, а сумма ординат была минимальной.

Оптимум непрерывной задачи (3), (4), который, очевидно, не более оптимума задачи (1), (2), примем в качестве искомой нижней границы. Решение непрерывной задачи (3), (4) можно найти следующим образом.

1. Фиксируем точку с минимальными значениями всех z_i, то есть на каждой ломаной берём начальную точку. Ей соответствует значение целевой функции

$$\sum_{i=1}^{n} w_i(f_i^1) = \sum_{i=1}^{n} g_i^1.$$

Это максимальное значение целевой функции при допустимых значениях переменных, то есть искомых абсцисс. Предстоит в n-мерном пространстве построить траекторию спуска так, чтобы, не нарушая ограничений, максимально уменьшить целевую функцию, то есть спуститься как можно ниже. Фиксируем остаток ресурса

$$T = R - \sum_{i=1}^{n} f_i^1.$$

2. Из всех звеньев всех ломаных из L_2 выбираем звено с максимальным уклоном. Пусть это будет звено с номером j ломаной с номером s. На первом шаге $j = 1$ в силу выпуклости ломаных. Будем менять только переменную z_s. Её увеличение даёт уменьшение целевой функции с наибольшей скоростью. Вычисляем шаг движения $c = \min(b_s^{j+1} - b_s^j, T)$ и меняем z_s на $z_s + c$. Целевая функция уменьшится на $u_s^j c$, а оставшийся ресурс — на c.

3. Из оставшихся звеньев всех ломаных из L_2 выбираем звено с максимальным уклоном и повторяем п. 2, пока оставшийся ресурс T не будет исчерпан.

Отметим, что в точке минимума все переменные z_s, кроме, быть может, той, которая менялась на последнем шаге, примут значения из соответствующих множеств B_s.

Если при выборе звена с максимальным уклоном таких звеньев оказывается несколько, то приоритет отдаётся звену с максимальной длиной, которое оставшийся ресурс позволяет использовать полностью (расход ресурса не превышает его текущий остаток), то есть $b_s^{j+1} - b_s^j \leqslant T$. При отсутствии такого звена среди звеньев с равными максимальными уклонами используется любое из них, например звено с наименьшей длиной. Попытка отказаться от использования максимальных уклонов в этой ситуации и разместить полностью какой-либо другой элемент приводит к неоправданному усложнению алгоритма и увеличению времени счёта

Если окажется, что и последняя из изменяемых переменных z_k в точке минимума примет значение из множества B_k ($b_k^{j+1} - b_k^j = T$), то решение непрерывной задачи совпадает с решением исходной дискретной задачи, и на этом расчёт заканчивается. В противном случае полученное решение непрерывной задачи является начальным приближением, а значение целевой функции в точке минимума является искомой нижней границей H. Если последним рассматривалось звено ломаной, которая изначально была выпуклой, то приближённое решение исходной дискретной задачи получаем, аннулируя последний шаг (этим объясняется стремление сделать его как можно меньше). В противном случае по оптимальному значению z_k^* определяется абсцисса ближайшей слева вершины k-й ломаной из L_1. Ординату этой вершины обозначим через g_k^*. Нижняя граница H не изменяется, а рекорд становится равным $E = H + g_k^* - w_k(z_k^*)$.

Изложенный алгоритм даёт оптимальное решение непрерывной задачи за счёт выбора на каждом шаге максимального из всех оставшихся уклонов, то есть оптимального варианта использования ресурса. Чтобы в этом убедиться, рассмотрим первый шаг алгоритма. Если максимальный уклон u_s^1 не использовать, то соответствующий ему ресурс $b_s^2 - b_s^1$ будет исчерпан каким-то другим способом (изменится одна или несколько других переменных вместо z_s), но при этом снижение затрат будет меньше.

На всех прочих шагах также нет смысла игнорировать максимальный уклон, так как очевидно, что именно он даёт наибольшее снижение затрат.

Другими словами, предположив, что в точке минимума непрерывной задачи $z_s = z_s^1$, приходим к противоречию, так как уменьшение абсциссы любой другой точки при соответствующем увеличении z_s, очевидно, даёт уменьшение целевой функции. Изложенный алгоритм можно использовать и при построении границ затрат на оставшихся шагах для каждой паретовской точки. Вместо R нужно взять оставшийся ресурс, начальная точка имеет координаты f_i^1, $i = r, r+1, \ldots, n$, где r — номер очередного шага. Значение целевой функции в этой точке равно

$$\sum_{i=1}^{n} w_i(f_i^1).$$

Для реализации алгоритма существенное значение имеет способ поиска наибольших уклонов. Простой способ состоит в сортировке всех уклонов всех звеньев ломаных из L_2 как единого массива в порядке невозрастания. Возможны и более экономные способы, но мы не будем их рассматривать.

Аналогично может быть решена и задача (1), (2) на максимум. При этом алгоритм несколько изменяется: используются вогнутые оболочки вместо выпуклых, решение непрерывной задачи становится нижней границей и т. д. Однако, чтобы не разрабатывать новую компьютерную программу, можно преобразовать задачу на максимум в эквивалентную ей задачу на минимум, взяв в качестве целевой функции вместо

$$\sum_{i=1}^{n} g_i(x_i)$$

функцию

$$\sum_{i=1}^{n} (g_{i_{\max}} - g_i(x_i)),$$

где $g_{i_{\max}}$ — максимальная эффективность для i-го потребителя ресурса. В вышеупомянутых частных случаях рассматриваемой задачи вычисления существенно упрощаются, так как при $f_i(x_i) = p_i x_i$ и $g_i(x_i) = c_i x_i$ для каждой ломаной все звенья имеют один уклон, а при $x_i \in \{0, 1\}$ каждая ломаная состоит из одного звена.

Экспериментально установлено, что для задачи (1), (2) общего вида, особенно при вещественных значениях функций $f_i(x_i)$, алгоритм с отбраковкой паретовских точек по методу ветвей и границ даёт существенное снижение времени счёта по сравнению с отбраковкой только непаретовских точек.

Так, уже упоминавшаяся задача о защите поверхности [5] при числе элементов $n = 400$ и числе способов защиты различных элементов k_i, равном 13, 42 или 45, решалась без отбраковки непаретовских точек по алгоритму «киевский веник» за 65 мин, при отбраковке только непаретовских точек за 12 с, а при дополнительной отбраковке паретовских точек по методу ветвей и границ за 2 с при снижении максимального (по шагам) числа точек с 11954 до 2609. Решение по методу регулярной сетки [1] с той же точностью потребовало более 3 ч, что объясняется необходимостью введения малого дискрета и соответственно большим числом дискретов сетки при вещественных и неравноотстоящих значениях ресурса. При этом из-за недостаточности оперативной памяти (512 Мбайт) подключается внешняя память, обмен с которой занимает много времени.

Дополнительно были выполнены расчёты на моделях с числом элементов 600 и способов зашиты до 70. Установлено, что и в этом случае алгоритм дополнительной

отбраковки паретовских точек по методу ветвей и границ даёт существенное сокращение времени счёта по сравнению с отбраковкой только непаретовских точек, но это сокращение в значительной степени зависит от конкретных числовых данных при неизменных n и k_i.

Исследования свойств алгоритмов продолжаются, однако уже сейчас в целом можно констатировать, что новый алгоритм позволяет решать задачи распределения ресурса (1), (2) большой размерности много быстрее, чем традиционный алгоритм регулярной сетки [1] и алгоритм «киевский веник» [3] за счёт отбраковки бесперспективных вариантов (непаретовских и части паретовских точек) и всех их продолжений.

ЛИТЕРАТУРА

1. *Беллман Р., Дрейфус С.* Прикладные задачи динамического программирования. М.: Наука, 1965. 458 с.
2. *Струченков В. И.* Динамическое программирование с использованием множеств Парето // Дискрет. анализ и исслед. опер. 2008. Т. 15. № 6. С. 58–62.
3. *Михалевич В. С.* Последовательные алгоритмы оптимизации и их применение // Кибернетика. 1965. № 1. С. 45–46.
4. *Кузюрин Н. Н., Фомин С. А.* Эффективные алгоритмы и сложность вычислений. М.: МФТИ, 2007. 210 с.
5. *Струченков В. И.* Методы оптимизации в прикладных задачах. М.: Солон-Пресс, 2009. 310 с.

ДИСКРЕТНЫЕ МОДЕЛИ РЕАЛЬНЫХ ПРОЦЕССОВ

DOI 10.17223/20710410/10/9

УДК 519.17

ИССЛЕДОВАНИЕ ЭВОЛЮЦИИ КЛЕТОЧНЫХ АВТОМАТОВ, МОДЕЛИРУЮЩИХ ПРОЦЕСС «РАЗДЕЛЕНИЯ ФАЗ» НА ТРЕУГОЛЬНОЙ СЕТКЕ

И. В. Афанасьев

Новосибирский государственный университет, г. Новосибирск, Россия

E-mail: ivafanas@gmail.com

Для уже существующего клеточного автомата на квадратной сетке, моделирующего процессы «разделения фаз», найдены клеточные автоматы на треугольной сетке, моделирующие такие же процессы. Выделен класс клеточных автоматов на квадратных и треугольных сетках, называемый «разделение фаз». Введены количественные и качественные характеристики клеточных автоматов этого класса. Путём проведения компьютерного моделирования клеточные автоматы «разделение фаз» разбиты на подклассы по поведенческим характеристикам. Показаны свойства некоторых представителей классов.

Ключевые слова: *клеточный автомат, процесс самоорганизации, разделение фаз, треугольная сетка.*

Введение

В середине 80-х годов были предложены клеточные автоматы (КА) для моделирования физико-химических процессов [1, 2], среди которых есть процессы, обладающие свойством самоорганизации (разделение фаз [1, 3], формирование диссипативных структур [4, 5]). *Самоорганизация* — процесс упорядочения (пространственного, временного или пространственно-временного) в системе в результате согласованного взаимодействия множества элементов, её составляющих. Разделение фаз — типичный пример процесса самоорганизации. В рамках этой работы под «разделением фаз» будем понимать процесс, в результате которого два вещества, равномерно распределенные по поверхности, объединяются в отдельные кластеры.

Предложенные ранее КА для процесса «разделение фаз» использовали квадратную сетку — она хорошо подходит для моделирования процессов на плоскости. Зачастую необходимо моделировать физические процессы на поверхности трехмерных геометрических объектов. Поскольку поверхность большинства геометрических объектов может быть аппроксимирована только треугольной сеткой, особый интерес представляет возможность использования треугольных сеток для процессов, моделируемых КА на квадратных сетках.

Гексагональные сетки необходимо использовать для процессов, в которых важна изотропность, например в задачах гидродинамики, газодинамики и диффузии. В рамках процесса «разделения фаз» изотропность не важна, поэтому от использования гексагональной сетки можно отказаться. К тому же гексагональные сетки плохо подходят для аппроксимации поверхностей трехмерных геометрических объектов.

В работе формально определён класс КА «разделение фаз», проведено исследование поведения этих КА на треугольной сетке, в которой каждый треугольник является равносторонним, введены количественные и качественные поведенческие характеристики КА «разделение фаз», исследована зависимость поведения КА от правил перехода. Проведено сравнение эволюции КА «разделение фаз» на треугольной и квадратной сетках.

1. Формальная постановка задачи

Клеточный автомат «разделение фаз» — тройка $\langle \Sigma, M, f \rangle$, где

Σ — алфавит состояний элементарных автоматов, $\Sigma = \{0,1\}$;

M — множество имён элементарных автоматов, $M = \{1, \ldots, N_1\} \times \{1, \ldots, N_2\}$;

f — функция перехода. Определение функции перехода будет дано ниже.

В дальнейшем КА «разделение фаз», встречающиеся в работе, будем называть просто КА. Клеткой называется элемент $m = (i, j)$ множества M (рис. 1). Множество пар $\Omega = \{(x, m)\}$, в котором все клетки различны, а x из Σ — состояние клетки m, называется *глобальным состоянием*.

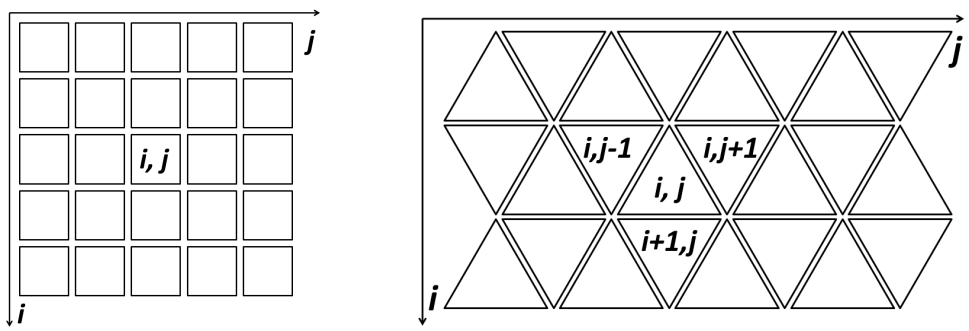

Рис. 1. Клетка КА как элемент множества индексов в случаях квадратной и треугольной сеток

Конечное множество функций $\varphi_k : M \to M$, $k = 0, 1, \ldots, q$, называется *шаблоном соседства* $T(m) = \{\varphi_0(m), \ldots, \varphi_q(m)\}$. Для каждой клетки m шаблон $T(m)$ определяет множество её соседей. При этом принято считать, что $\varphi_0(m) = m$. Функции $\varphi_k(m)$ при периодических граничных условиях можно представить в виде

$$\varphi_k((i, j)) = ((i + a) \bmod N_1,\ (j + b) \bmod N_2),$$

где a и b — некоторые константы.

Множество состояний клеток-соседей клетки m есть $X(T(m)) = \{x_0, \ldots, x_q\}$, где x_k — состояние клетки $\varphi_k(m)$, x_0 — состояние клетки m. *Функцией перехода* называется функция $f(x_0, \ldots, x_q) : \Sigma^{q+1} \to \Sigma$. Применить функцию перехода f к клетке m — значит изменить состояние клетки m на новое значение $f(x_0, \ldots, x_q)$. Применение функции перехода ко всем клеткам КА называется *глобальным оператором*. Смена состояний всех клеток КА в соответствии с функцией перехода называется *итерацией*. Итерация изменяет глобальное состояние КА $\Omega(t)$ на $\Omega(t+1)$, t — номер итерации. Итеративный процесс смены глобальных состояний называется *эволюцией*.

Существуют два основных режима применения глобального оператора: *синхронный* и *асинхронный*. Синхронный режим предполагает, что аргументы функции переходов — состояния клеток-соседей на текущей итерации t. На каждой итерации клетки

вычисляют значение нового состояния, и затем все клетки одновременно заменяют старые состояния на новые. При асинхронном режиме каждая клетка вычисляет функцию перехода от тех значений состояний соседей, которые на данный момент имеют место, и сразу меняет свое состояние. Причем клетка, обновляющая свое состояние, выбирается случайно [7].

Под *устойчивым глобальным состоянием* будем понимать то глобальное состояние КА, за t применений глобального оператора к которому изменяется не более чем ε процентов клеток. Количество клеток, изменивших свое состояние за t итераций, вычисляется по формуле

$$k = k_1 + k_2 + \ldots + k_t,$$

где k_i — количество клеток, изменивших свое состояние за i-ю итерацию. Значения t и ε зависят от конкретной задачи. В численных экспериментах $t = 10$ и $\varepsilon = 10$. Эти значения позволяют не учитывать характерные пограничные колебательные процессы, когда эволюция в целом достигает «достаточно устойчивого» глобального состояния. В рассматриваемом классе КА не отмечено устойчивых колебательных процессов, как при моделировании периодического химического процесса — реакции Белоусова — Жаботинского.

При моделировании процесса «разделения фаз» на квадратных сетках, как правило, выбирают квадратный шаблон соседства размером 3×3, 5×5 или 7×7. При этом в шаблоне соседства оказывается нечётное число клеток-соседей. Чтобы провести сравнительный анализ КА на квадратных и треугольных сетках, необходимо выбрать шаблон соседства для КА на треугольных сетках с таким же числом клеток-соседей. Довольно трудно подобрать симметричный шаблон соседства на треугольной сетке из 9 клеток. Поэтому далее везде шаблон соседства состоит из 25 клеток, $q = 24$ (рис. 2).

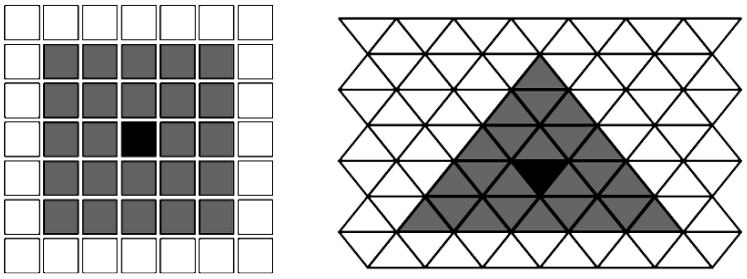

Рис. 2. Чёрным отмечена клетка клеточного массива, а серым — её шаблон соседства

Функция перехода $f(x_0, \ldots, x_{24})$ в рассматриваемом классе КА зависит от суммы состояний всех клеток-соседей. Клеток в шаблоне соседства 25 штук. Состояние каждой клетки равно либо 0, либо 1. Сумма состояний может принимать 26 значений:

$$s = x_0 + x_1 + \ldots + x_{24} \in \{0, 1, \ldots, 25\}. \tag{1}$$

Можно однозначно сопоставить функции перехода $f(x_0, x_1, \ldots, x_{24})$ булев вектор $\mathbf{v} = (\mathbf{v}(0), \ldots, \mathbf{v}(25))$ так, чтобы новое значение клетки было бы равно $\mathbf{v}(s)$:

$$f(x_0, x_1, \ldots, x_{24}) = \mathbf{v}(s). \tag{2}$$

Тогда функцию перехода можно однозначно определить множеством

$$A = \{i \in \{0, 1, \ldots, 25\} : \mathbf{v}(i) = 1\}; \tag{3}$$

$$f(x_0, x_1, \ldots, x_{24}) = \begin{cases} 1, & \text{если } s \in A; \\ 0, & \text{если } s \notin A. \end{cases} \tag{4}$$

Центром множества A назовём число

$$\operatorname{center}(A) = \frac{1}{|A|} \sum_{i \in A} i. \tag{5}$$

Для КА на квадратной и треугольной сетках можно определить понятие *ближайшего соседства* клеток *первого уровня*.

Клетки (i_1, j_1) и (i_2, j_2) КА на квадратной сетке будем называть *ближайшими соседями первого уровня*, если $|i_1 - i_2| + |j_1 - j_2| = 1$.

Клетки (i_1, j_1) и (i_2, j_2) КА на треугольной сетке будем называть *ближайшими соседями первого уровня*, если выполнено хотя бы одно из следующих условий:

1) $i_1 = i_2$ и $|j_1 - j_2| = 1$;
2) число $i_1 + j_1$ нечётное, $i_2 = i_1 + 1$ и $j_1 = j_2$;
3) число $i_1 + j_1$ чётное, $i_2 = i_1 - 1$ и $j_1 = j_2$.

Ближайшие соседи для клеток КА на квадратных и треугольных сетках изображены на рис. 3.

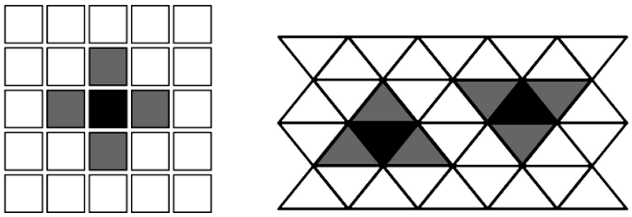

Рис. 3. Чёрным цветом отмечена клетка, серым — множество её ближайших соседей

Клетки m_1 и m_2 будем называть *ближайшими соседями второго уровня*, если они различны и существует клетка m_3, являющаяся ближайшим соседом первого уровня обеим клеткам m_1 и m_2. Множество клеток, содержащее только клетку m и её ближайших соседей первого и второго уровней, называется *ближайшим соседством радиуса 2* (рис. 4).

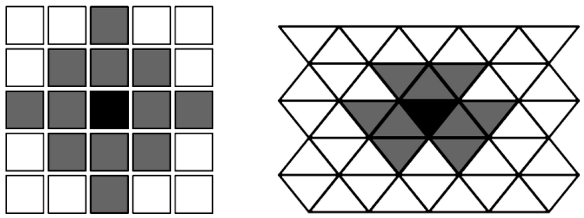

Рис. 4. Черным цветом отмечена клетка, серым — остальные клетки ближайшего соседства радиуса 2

Пример 1. Пример КА на квадратной сетке, моделирующего процесс «разделения фаз». Множество имён $M = \{1, \ldots, 300\} \times \{1, \ldots, 300\}$. Шаблон соседства представлен на рис. 2 слева; функция перехода выбрана следующая:

$$\begin{gathered} T((i,j)) = \{((i+a) \bmod 300, (j+b) \bmod 300) : -2 \leqslant a \leqslant 2, -2 \leqslant b \leqslant 2\}; \\ A = \{11, 13\} \cup \{15, 16, \ldots, 25\}. \end{gathered} \tag{6}$$

Граничные условия — периодические, т. е. крайние правые клетки являются соседями крайних левых и крайние верхние клетки являются соседями крайних нижних. Автомат действует асинхронно. Некоторые глобальные состояния КА показаны на рис. 5.

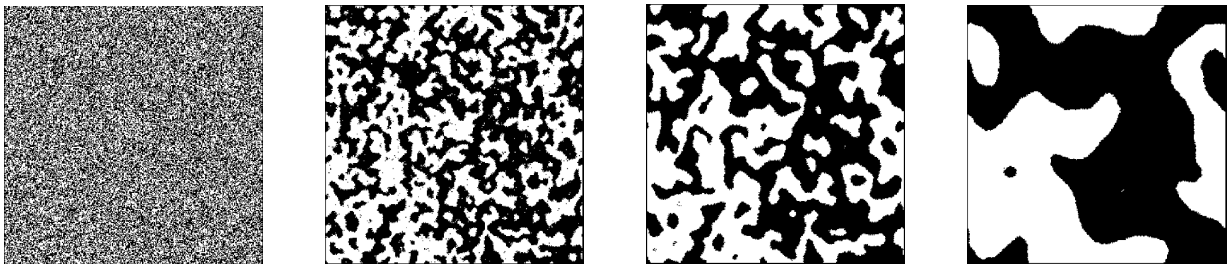

Рис. 5. Эволюция КА «разделение фаз» (300 × 300 клеток). Итерации (слева направо): $t = 0, 5, 20, 300$

Цель настоящего исследования — экспериментально получить зависимость свойств эволюции от функции перехода (параметра A) и сравнить эволюции синхронных и асинхронных КА «разделение фаз» на треугольных и квадратных сетках. Эволюции КА на треугольных и квадратных сетках сравнивались мотивами. *Мотив* — повторяющийся элемент или структура в глобальной картине КА. В данной работе были выбраны следующие критерии для сравнения мотивов.

1) Разделяемость веществ в устойчивом глобальном состоянии не более чем на два кластера. Под *кластером* будем понимать наибольшее по включению множество клеток с одинаковыми состояниями, в котором между любыми двумя клетками есть путь, содержащий только клетки этого множества. *Путём* между клетками m_1 и m_2 назовём конечную последовательность клеток $a_1, a_2, \ldots, a_k$, где $a_1 = m_1$, $a_k = m_2$ и клетка a_i является ближайшим соседом первого уровня клетки a_{i+1} для всех $i \in \{1, \ldots, k-1\}$.

2) Мелкозернистость

$$\text{finegrainess} = \frac{N_{\text{bound}}}{N}, \tag{7}$$

где N — общее число клеток в клеточном массиве; N_{bound} — число граничных клеток. Клетку будем называть *граничной*, если среди её ближайших соседей найдется клетка с другим состоянием.

3) Нечёткость границ

$$\text{fuzzy} = \frac{N_{\text{fuzzy}}}{N_{\text{bound}}}, \tag{8}$$

где N_{fuzzy} — число граничных нечётких клеток в клеточном массиве. Клетку будем называть граничной нечёткой, если в её ближайшем соседстве радиуса 2 все клетки граничные.

В общем случае параметры мелкозернистости и нечёткости границ изменяются в процессе эволюции КА.

2. Метод исследования

Эволюция КА «разделение фаз» исследовалась экспериментально. При выборе шаблона соседства из 25 клеток всего возможно 2^{26} функций перехода состояний клетки. Исследовать 2^{26} вариантов КА — трудоемкая задача. Если принять во внимание

«равноправие» веществ, для которых моделируется разделение фаз, то можно сделать предположение об антисимметричности вектора $\mathbf{v}$:

$$\begin{aligned} &\forall i \in \{0,\ldots,25\} \quad \mathbf{v}(i) = 1 - \mathbf{v}(25-i), \\ &\forall i \in \{0,\ldots,25\} \quad i \in A \Leftrightarrow (25-i) \notin A. \end{aligned} \tag{9}$$

Условие «равноправия» веществ (9) означает, что при наличии ровно i единиц в соседстве клетки её новое состояние будет равным 1 тогда и только тогда, когда при наличии ровно i нулей в соседстве клетки её новое состояние будет равным нулю.

Условие (9) позволяет сократить количество рассматриваемых КА до 2^{13}, так как достаточно расставить произвольным образом нули и единицы в левой половине вектора значений функции перехода и антисимметрично отразить его от середины.

При проведении экспериментов было отмечено, что, зачастую, если $\mathbf{v}(i) = 1$ для некоторого значения $i < 8$, то моделируемый процесс оказывается хаотическим. *Динамический хаос* — процесс в теории динамических систем, при котором поведение нелинейной системы выглядит случайным несмотря на то, что оно определяется детерминистическими законами [6]. Далее такие КА не рассматривались: в проводимых экспериментах $\mathbf{v}(0) = 0, \ldots, \mathbf{v}(7) = 0$.

Таким образом, было экспериментально исследовано $2^5 = 32$ варианта КА следующего вида:

1) $\mathbf{v}(0) = 0, \ldots, \mathbf{v}(7) = 0$ (отсечение большинства хаотических процессов);
2) единицы и нули произвольно расставлены на места $\mathbf{v}(8), \ldots, \mathbf{v}(12)$ (2^5 вариантов);
3) $\mathbf{v}(13), \ldots, \mathbf{v}(25)$ вычисляются из условия «равноправия» веществ (9).

Эксперименты проводились следующим образом:

1) размерность квадратной сетки — 400×400. Размерность треугольной сетки — 600×300;
2) начальное состояние задано случайным распределением 1 и 0 с плотностью 0,5;
3) эволюция останавливается в глобальном устойчивом состоянии;
4) в глобальном устойчивом состоянии измеряются коэффициенты мелкозернистости и нечёткости границ.

Для проведения численных экспериментов было реализовано программное приложение на языке C++ с использованием технологии OpenGL.

3. Результаты исследований

3.1. Разделение КА на классы по свойствам мотивов

В результате экспериментов КА, как на треугольной сетке, так и на квадратной, как синхронные, так и асинхронные, были разделены на классы по свойствам мотивов.

Класс 1. Полное разделение фаз. КА этого класса разделяют вещества на два кластера.

В качестве представителя этого класса рассмотрен синхронный КА на треугольной сетке при $A = \{15,\ldots,25\} \cup \{11, 13\}$ (рис. 6).

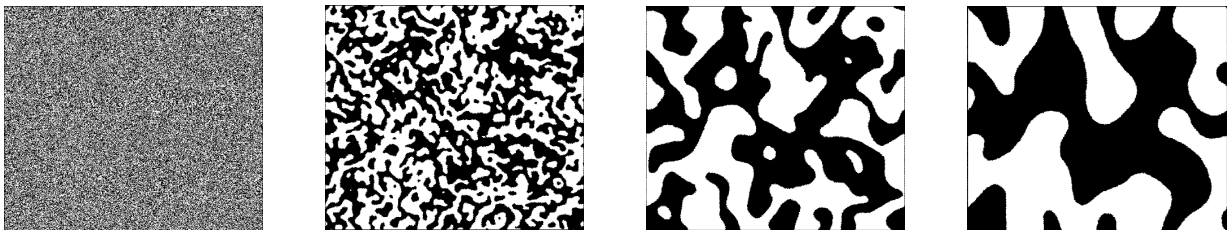

Рис. 6. Эволюция КА — представителя первого класса. Итерации (слева направо): $t = 0, 15, 230, 1000$

Класс 2. Мелкозернистые. К КА этого класса отнесём те КА типа «разделение фаз», коэффициент мелкозернистости в устойчивом глобальном состоянии которых больше некоторой величины, зависящей от цели исследования. В нашем случае это 0,07.

В качестве представителя класса «мелкозернистые» рассмотрен синхронный КА на треугольной сетке при $A = \{14, \ldots, 25\} \cup \{12\}$ (рис. 7).

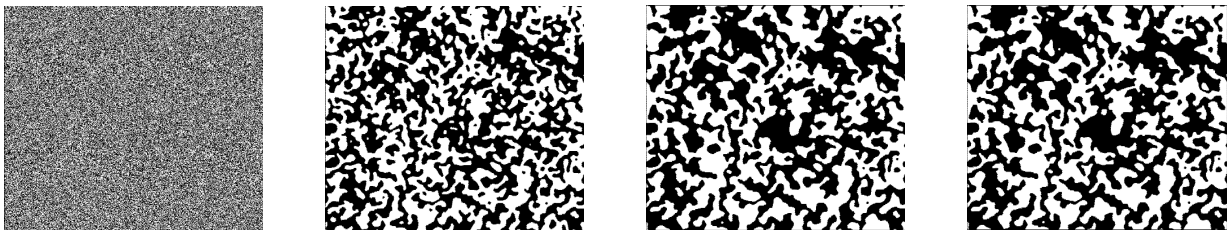

Рис. 7. Эволюция КА — представителя второго класса. Итерации (слева направо): $t = 0, 15, 230, 1000$

Класс 3. Нечёткие границы. К КА этого класса отнесём те КА типа «разделение фаз», коэффициент нечёткости границ в устойчивом глобальном состоянии которых больше некоторой величины, зависящей от цели исследования. В нашем случае это 0,005.

В качестве представителя класса «нечёткие границы» рассмотрен синхронный КА на треугольной сетке при $A = \{18, \ldots, 25\} \cup \{8, 9, 13, 14, 15\}$ (рис. 8).

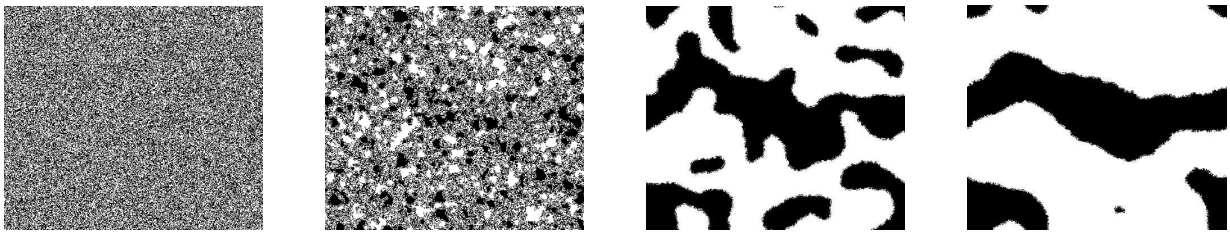

Рис. 8. Эволюция КА — представителя третьего класса. Итерации (слева направо): $t = 0, 15, 230, 1000$

Класс 4. Динамический хаос. КА этого класса не разделяют вещества в итоге процесса на два кластера и не достигают устойчивого глобального состояния.

Синхронный КА на треугольной сетке при $A = \{18, \ldots, 25\} \cup \{8, 9, 10, 11, 13\}$ является представителем класса «динамический хаос» (рис. 9).

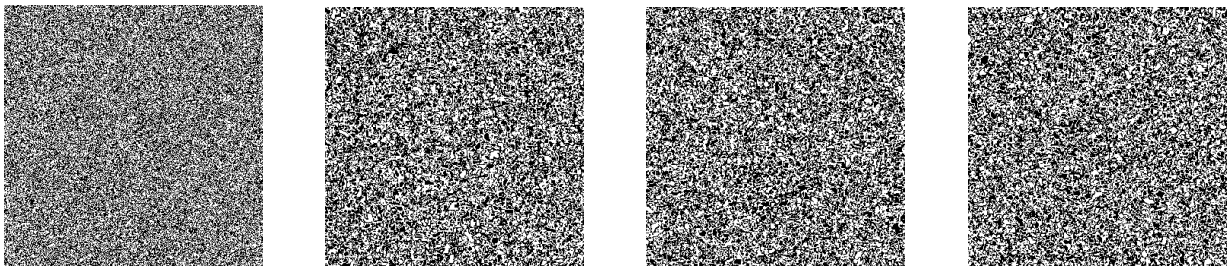

Рис. 9. Эволюция КА — представителя четвёртого класса. Итерации (слева направо): $t = 0, 15, 230, 1000$

Анализ эволюции не проводился для класса «динамический хаос». Получившееся распределение КА «разделение фаз» по классам:

1) КА, центр которых удовлетворяет неравенствам $18{,}077 \leqslant \mathrm{center}(A) \leqslant 18{,}923$; 56 штук. Для некоторых из них (10 КА) коэффициент мелкозернистости в устойчивом глобальном состоянии превосходит 0,07 (критерий принадлежности к классу «мелкозернистые»). Остальные КА относятся к классу «полное разделение фаз». Среди КА на треугольных сетках, попадающих под описанное ограничение на центр множества A, есть КА класса «нечёткие границы». Причем таких КА на треугольных сетках больше среди синхронных (6 КА) и меньше среди асинхронных (2 КА).
2) КА, центр которых лежит в границах $17{,}962 \leqslant \mathrm{center}(A) \leqslant 18{,}077$ (40 штук), принадлежат классу «нечёткие границы». Только один КА (синхронный на квадратной сетке, $A = \{17, \ldots, 25\} \cup \{9, 10, 11, 12\}$) принадлежит классу «мелкозернистые». Четыре КА из этого множества (все асинхронные, два на треугольной сетке и два на квадратной, соответствуют одним и тем же множествам A) принадлежат классу «динамический хаос».
3) Почти все КА, центр которых удовлетворяет условиям $17{,}077 \leqslant \mathrm{center}(A) \leqslant$ $\leqslant 17{,}615$ (28 штук), принадлежат классу «динамический хаос». Исключение составляет один КА на квадратной сетке с синхронным режимом функционирования, с множеством $A = \{18, \ldots, 25\} \cup \{8, 10, 11, 12, 16\}$, принадлежащий классу «нечёткие границы».

Упорядочение КА «разделение фаз» по параметру «центр множества A» позволяет отделить КА, моделирующие хаотические процессы, от остальных КА типа «разделение фаз».

3.2. Динамические свойства КА представителей классов

Интерес также представляет изменение перечисленных выше свойств в динамике процессов. Поэтому для всех классов КА важной характеристикой является коэффициент активности процесса. Это свойство характеризует скорость приближения эволюции к устойчивому состоянию.

Формально *коэффициент активности* процесса на итерации с номером i_0 определим как количество клеток, изменивших своё состояние за последние n итераций ($i_0 > n$):

$$k(i_0) = k_1 + k_2 + \ldots + k_n,$$

где k_i — количество клеток, изменивших свое состояние за $(i_0 - i)$-ю итерацию. Далее считаем $n = 10$. На рис. 10 представлена диаграмма изменения параметра активности для различных представителей классов КА в процессе эволюции.

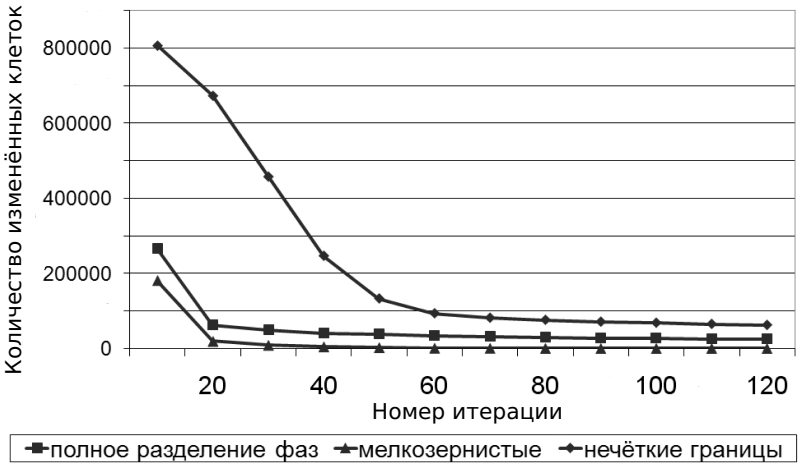

Рис. 10. Изменение коэффициента активности в процессе эволюции

Экспериментально получено, что КА «разделение фаз» класса «мелкозернистые» быстрее, а КА класса «нечёткие границы» медленнее приходят к устойчивому состоянию. Причем активность КА класса «нечёткие границы», как правило, выше, чем активность КА других классов. Это объясняется тем, что вероятность изменения состояния граничной клетки выше, чем не граничной. На начальных итерациях количество граничных клеток в глобальном состоянии КА класса «нечёткие границы» выше, чем у других КА. Это свойство продемонстрировано на рис. 11.

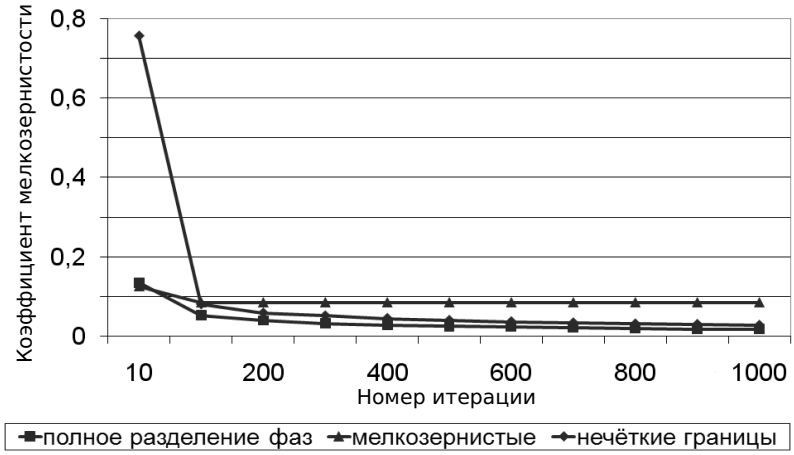

Рис. 11. Изменение мелкозернистости в процессе эволюции

Коэффициент мелкозернистости, пропорциональный количеству граничных клеток, для представителя класса «мелкозернистые» практически не изменяется после 110-й итерации, и КА достигает устойчивого глобального состояния. Далее происходят незначительные колебания состояний граничных клеток. Поэтому параметры активности, мелкозернистости и нечёткости границ для представителя класса «мелкозернистые» почти не изменяются, колеблясь в пределах 0,01 % от абсолютных своих значений.

Коэффициент нечёткости границ очень близок к нулю в процессе эволюции представителей классов «полное разделение фаз» и «мелкозернистые». Для этих КА коэффициент нечёткости границ меньше 10^{-5} уже после 60-й итерации. Коэффициент

нечёткости границ для представителя класса «нечёткие границы» после 100 итераций колеблется около 0,05 (рис. 12).

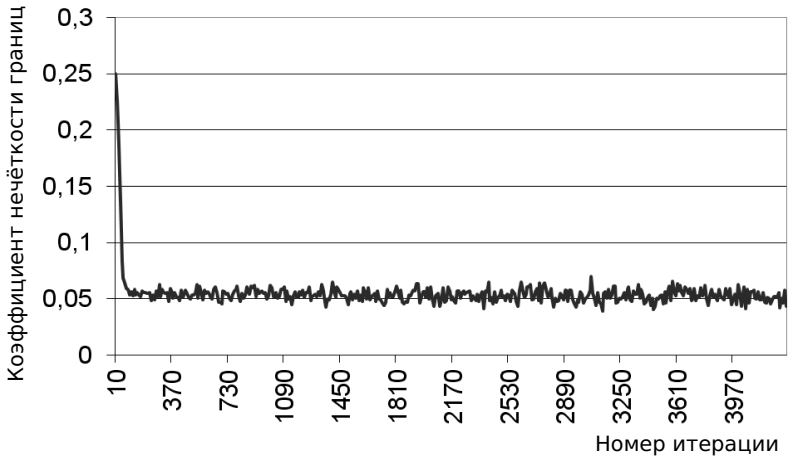

Рис. 12. Изменение параметра нечёткости границ в процессе эволюции КА — представителя класса «нечёткие границы»

Значения мелкозернистости и нечёткости границ измерялись двумя способами:

1) *Значения параметров в устойчивом глобальном состоянии.* Глобальный оператор применялся до тех пор, пока КА не придет к устойчивому глобальному состоянию. Значения параметров вычислялись для этого устойчивого глобального состояния. Они обозначаются индексом «stab». Значения параметров в устойчивом глобальном состоянии никак не отражают динамику этих параметров в процессе эволюции КА.
2) *Значения параметров, нормированные по активности.* Пусть k_t — значение параметра активности на итерации с номером t. Пусть p_t — значение некоторого параметра $p \in \{\text{finegrainess}, \text{fuzzy}\}$ для глобального состояния, соответствующего итерации с номером t с начала эволюции КА. Пусть T — номер итерации, при которой наступает устойчивое глобальное состояние. Нормированное по активности значение параметра p вычисляется по формуле

$$p_{\text{act}} = \frac{p'}{k'}, \quad \text{где} \quad p' = \sum_{i=1}^{T/n} \frac{p_n}{k_n}; \quad k' = \sum_{i=1,}^{T/n} \frac{1}{k_n}.$$

Параметр n выбран равным 10.

В общем случае, чем быстрее протекает процесс, тем быстрее изменяются параметры процесса. Значения параметров, нормированные по активности, позволяют учесть, во-первых, значения параметров безотносительно к скорости протекания процесса, во-вторых, динамику параметров в процессе эволюции КА.

Значения коэффициентов мелкозернистости и нечёткости границ для представителей классов КА показаны в таблице.

Значения коэффициентов мелкозернистости и нечёткости границ для представителей классов КА

Функция перехода	T	Коэффициент мелкозернистости		Коэффициент нечёткости границ	
		stab	act	stab	act
$A = \{15, \ldots, 25\} \cup \{11, 13\}$ «полное разделение фаз»	250	0,035	0,047	0	0
$A = \{14, \ldots, 25\} \cup \{12\}$ «мелкозернистые»	30	0,093	0,097	0	0
$A = \{18, \ldots, 25\} \cup \{8, 9, 13, 14, 15\}$ «нечёткие границы»	2280	0,022	0,029	0,057	0,054

Важность введения нормировки значений параметров по активности можно продемонстрировать на примере коэффициента нечеткости границ для КА-представителя класса «нечёткие границы». Начиная с 100-й итерации, коэффициент нечёткости границ колеблется между 0,039 и 0,07, т. е. коэффициент нечёткости границ в глобальном устойчивом состоянии может отличаться до 1,7 раз. Введение нормировки по активности позволяет учесть колебательный характер изменения коэффициента нечёткости границ в процессе эволюции КА и показать «усреднённое» значение.

Заключение

Для процесса «разделение фаз», моделируемого КА на квадратной сетке, были найдены локальные операторы и шаблон соседства, позволяющие моделировать процесс КА на треугольной сетке с равносторонними треугольниками. Выделены классы КА на квадратной и треугольной сетках, моделирующих самоорганизующийся процесс «разделения фаз».

В результате экспериментов выделено четыре основных класса КА на треугольной сетке, по мотивам эволюции полностью соответствующие четырём классам КА на квадратных сетках.

КА класса «полное разделение фаз» можно использовать при моделировании процесса разделения фаз двух несмешивающихся жидкостей, например масла на поверхности воды.

КА класса «мелкозернистые» можно использовать при моделировании социальных и физических процессов, где имеет место кластеризация элементов в мелкие устойчивые структуры. Например, в задачах пространственного распределения людей разных языковых групп или при построении пористых сред.

ЛИТЕРАТУРА

1. *Toffolli T.* Computation and construction universality of reversible automata // J. Comp. System Science. 1987. V. 15. P. 1–6.
2. *Toffolli T., Margolus N.* Cellular Automata Machines. USA: MIT Press, 1987.
3. *Wolfram S.* A new kind of science. USA: Wolfram Media Inc., Champaign, Ill, 2002.
4. *Пригожин И.* От существующего к возникающему. Время и сложность в физических науках. М.: Наука, 1985.

5. *Deutch A., Dormann S.* Cellular Automaton Modeling of Biological Pattern Formation. Berlin: Birkhauser, 2004.
6. *Saber N.* Discrete Chaos. Chapman and Hall, CRC, 1999.
7. *Бандман О. Л.* Клеточно-автоматные модели пространственной динамики // Системная информатика. 2005. Вып. 10. С. 57–113.

2010 Дискретные модели реальных процессов №4(10)

DOI 10.17223/20710410/10/10

УДК 621.391.1:004.7

МЕТОД ПОСТРОЕНИЯ КЛЕТОЧНО-АВТОМАТНЫХ МОДЕЛЕЙ ПРОЦЕССОВ ФОРМИРОВАНИЯ УСТОЙЧИВЫХ СТРУКТУР[1]

О. Л. Бандман

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Россия

E-mail: bandman@ssd.sscc.ru, bandman@academ.org

Представлен метод построения клеточных автоматов (КА), моделирующих процессы самоорганизации при формирования устойчивых структур. Метод основан на параллельной композиции двух КА. Рассмотрено два случая: 1) когда один компонентный КА функционирует независимо, влияя на эволюцию второго, и 2) когда оба КА взаимодействуют на каждой итерации. Метод иллюстрируется результатами компьютерного моделирования двух самоорганизующихся систем: формирования устойчивых пространственных структур на подогретой поверхности и достижения баланса между хищником и жертвой.

Ключевые слова: *математическое моделирование, клеточный автомат, диссипативные структуры, самоорганизация.*

Введение

Интерес к математическому и компьютерному моделированию диссипативных процессов в неравновесных системах вызван, прежде всего, интенсификацией исследований химических, биологических, социологических и других явлений. Среди процессов с такими свойствами известны обычные, спиралевидные, концентрические, вращающиеся и стоячие волны (автоволны), а также колебательные и устойчивые пространственные структуры. Наиболее известны теоретические исследования диссипативных процессов [1], химических автоволн [2], биологических и экологических явлений [3]. Изучение устойчивых диссипативных структур началось с появления работы А. Тьюринга по морфологии [4]. Полный и хорошо написанный обзор исследований в этой области дан в [5]. Почти все известные работы основаны на математических моделях традиционного типа — системах дифференциальных уравнениях с частными производными. Поскольку исследуемые явления диссипативны и пространственно распределены, они относятся к процессам типа «реакция — диффузия», описываемым уравнениями с нелинейными членами, которые не имеют аналитических решений, а численные методы требует больших вычислительных ресурсов.

Однако, поскольку научный интерес к диссипативным структурам растет, поиск новых моделей интенсифицируется. В частности, появляются модели дискретного и стохастического типа, в которых химические преобразования и пространственная подвижность частиц вещества моделируются непосредственным (прямым) образом. К таким моделям относятся клеточные автоматы [6], клеточно-нейронные сети [7] и кинетические методы Монте-Карло [8]. В данной работе используется математическая модель, основанная на расширенном понятии клеточного автомата [9], отличающаяся

[1]Работа поддержана Программой фундаментальных исследований Президиума РАН №2-6 (2010) и Сибирским отделением РАН, Интеграционный проект 32 (2010).

от классического КА фон Неймана тем, что в ней допускается применение любого алфавита, любых функций переходов, любых режимов смены состояний клеток. Такая широкая трактовка КА позволяет строить математические описания пространственно-временных процессов разного характера, в том числе и обладающих свойствами самоорганизации и формирования устойчивых образов (структур).

Использование КА-моделей для компьютерного моделирования пространственной динамики в химии и физике привлекательно по следующим причинам.

- КА моделирует дискретные изменения состояний в дискретных пространстве и времени, что позволяет непосредственным образом отображать в модели перемещения и трансформации реальных частиц или агентов.
- На функции переходов КА не накладывается никаких ограничений: они могут быть нелинейными, разрывными и вероятностными, что позволяет моделировать такие процессы, как фазовые переходы и химические превращения, а также учитывать условия, при которых допустимо изменение того или иного состояния.
- КА допускают моделирование не синхронизированных процессов (асинхронные КА), что соответствует естественному течению событий во всех реальных явлениях, в которых не введена искусственно синхронизация.
- КА моделируют объекты, которые в [6, 10] называются *сложными системами* (*complex systems*), так как имеют очень простые математические представления, но моделируют сложные пространственно-временны́е процессы. Отсюда вытекает простота программирования КА-моделей как в последовательном, так и в параллельном вариантах.

Исследование КА как дискретных моделей пространственной динамики началось в 80-х годах прошлого столетия работами [11, 12] и в настоящее время интенсивно развивается, охватывая все новые области задач и совершенствуя теорию и методологию [10, 13, 14]. К развитию методологии КА-моделирования и относится предлагаемая работа. В ней после введения даны формальные определения и пример простого КА (п. 1), моделирующего процесс образования устойчивых структур. В п. 2 представлен метод параллельной композиции КА [15], в п. 3 и 4 иллюстрируются два типовых случая его применения для построения КА-моделей этого класса.

1. Простые самоорганизующиеся клеточные автоматы

Атомарным понятием КА является клетка, которая характеризуется именем $x \in X$ и состоянием $a \in A$, где X — *множество имен* клеток, которое обычно интерпретируется как множество координат точек в дискретном пространстве конечных размеров; A — *алфавит* состояний клеток. Состояние клетки x в момент времени t обозначается как $v_x(t)$. На множестве имен определен *шаблон соседства* $T(x) = \{\varphi_j(x) : j = 0, \ldots, q\}$, который определяет имена клеток, взаимодействующих с клеткой x. Если пространство моделирования представлено решеткой с прямоугольной, треугольной или квадратной формой клетки, то функции $\varphi_j(x) \in T(x)$ имеют вид сдвиговых: $\varphi_j(x) = x + l_j$, а шаблон состоит из q плотно примыкающих друг к другу клеток с условным «центром» в x. Шаблон связывает с каждой клеткой $x \in X$ *локальную конфигурацию* $V(x) = \{v_1, \ldots, v_q\}$ таким образом, что каждая клетка $\varphi_j(x) \in T(x)$ имеет состояние $v_j \in V(x)$. Множество состояний всех клеток $x \in X$ в момент t называется *глобальной конфигурацией* $\Omega(t) = \{v_x(t) : x \in X\}$.

Функционирование КА задается локальным оператором над подстановками $\Theta(\theta_1(x), \ldots, \theta_n(x))$. Если локальный оператор задан одной подстановкой, то он назы-

вается *простым*. Подстановка имеет следующий вид:

$$\theta(x) : V(x) \cup V''(x) \to V'(x), \tag{1}$$

где $V(x)$, $V'(x)$ и $V''(x)$ — *базовая*, *новая* и *контекстная* локальные конфигурации соответственно, причем $V(x)$ и $V'(x)$ связаны с одним и тем же шаблоном $T(x)$. Подстановка называется вероятностной, если ее применение выполняется при условии $\texttt{rand} < p$, где $\texttt{rand}$ — случайное число в интервале [0,1], p — заданная вероятность. Применение θ к клетке x состоит в замене состояний $v_j \in V(x)$ на новые $v'_j \in V'(x)$, которые являются значениями функций перехода вида

$$v'_j = f_j(V(x) \cup V''(x)). \tag{2}$$

Применение подстановки к клетке x на итерации t называется успешным, если

$$(V(x) \cup V''(x) \subseteq \Omega(t)) \ \& \ (\texttt{rand} < p), \tag{3}$$

даже если не произошло ни одного изменения состояний, т. е. $f_j(V(x)) = V(x)$. Если (3) не выполняется, то попытка применения подстановки безуспешна.

Функционирование КА подчиняется итерационному алгоритму, в котором итерацией считается применение локального оператора (успешное и безуспешное) ко всем клеткам $x \in X$. При этом происходит переход $\Omega(t) \to \Omega(t+1)$. Последовательность $\Omega(0), \ldots, \Omega(t), \ldots, \Omega(\hat{t})$ называется *эволюцией* КА. Если после какой-то итерации $t = \hat{t}$ дальнейшее успешное применение локального оператора не изменяет глобальную конфигурацию или меняет ее не более чем в заданном числе клеток ε, т. е.

$$|X| - |\Omega(T+1) \cap \Omega(T)| < \varepsilon, \tag{4}$$

то КА обладает свойством *самоорганизации* и относится к классу 2 по классификации Вольфрама [6], а с точки зрения качественной теории нелинейных систем эти КА характеризуются тем, что имеют устойчивый фокус. Свойство самоорганизации проявляется тогда, когда система состоит из множества элементов, взаимодействующих как с положительной, так и с отрицательной обратной связью (физическая интерпретация). Иными словами, взаимодействующие элементы являются по отношению друг к другу либо *активаторами*, либо *ингибиторами* (химическая интерпретация).

Порядок применений подстановок при переходе от $\Omega(t)$ к $\Omega(t+1)$ называется *режимом*. Базовыми являются два режима: синхронный и асинхронный. При синхронном режиме итерация состоит из двух следующих шагов:

1) вычисляются значения функции переходов (2) для всех клеток, в которых (3) выполнено;

2) в базовых конфигурациях всех клеток производится смена состояний согласно (1).

Для синхронного режима важно, чтобы выполнялось условие корректности вычислений [13]

$$\forall t = 1, \ldots, \hat{t} \ (T(x_i) \cap T(x_j) = \varnothing).$$

Очевидно, что оно всегда выполнено при $|V'(x)| = 1$.

При асинхронном режиме итерация состоит из последовательности $|X|$ шагов. На каждом шаге

1) случайно выбирается имя клетки $x_i \in X$;

2) вычисляются значения функций переходов для новой локальной конфигурации $V'(x_i)$ по (2);

3) в базовой конфигурации $V(x_i)$ производится смена состояний на соответствующие из $V'(x_i)$.

Среди простых самоорганизующихся КА наиболее известны два типа: 1) КА, моделирующие процессы, называемые «разделением фаз», и 2) КА с взвешенными шаблонами, моделирующие процессы формирования устойчивых структур [7]. Оба эти типа КА имеют устойчивые состояния как при синхронном, так и при асинхронном режимах. КА первого типа подробно описаны в [13, 14], их эволюция обладает свойством самоорганизации, но получающиеся в результате устойчивые структуры тривиальны — в них $v_x(t) = 1$ или $v_x(t) = 0$ для всех $x \in X$. Второй тип КА принадлежит к классу клеточно-нейронных сетей [7]. Использование в их функциях перехода взвешенных шаблонов позволяет получать устойчивые структуры в виде разного вида пятен и полос.

Пример 1 (рис. 1). Типичный представитель класса КА формирования устойчивых структур имеет $A = \{0, 1\}$, $X = \{(i, j) : i, j = 0, 1, \ldots, M\}$, а локальный оператор опирается на шаблон

$$T(i, j) = \{(i + g, j + h) : g, h = -3, \ldots, 0, \ldots, 3\}. \tag{5}$$

Этому шаблону ставится в соответствие весовая матрица W, элементы которой равны

$$w_{gh} = \begin{cases} a, & \text{если } |g| \leqslant 1,\ |h| \leqslant 1, \\ b & \text{иначе,} \end{cases} \tag{6}$$

где $a > 0$ соответствует *активатору*, а $b < 0$ — *ингибитору*. Локальный оператор $\theta(i, j)$ меняет состояние только одной клетки, используя в функции переходов состояния всех клеток из $T(i, j)$:

$$\theta(i, j) : \{v_{i+g,j+h} : g, h = -3, \ldots, 0, \ldots, 3\} \to v'_{i,j}, \tag{7}$$

где

$$v'_{i,j} = \begin{cases} 1, & \text{если} \quad \sum\limits_{g=-r}^{r} \sum\limits_{h=-r}^{r} w_{gh} v_{i+g,j+h} > 0{,}1, \\ 0 & \text{иначе.} \end{cases} \tag{8}$$

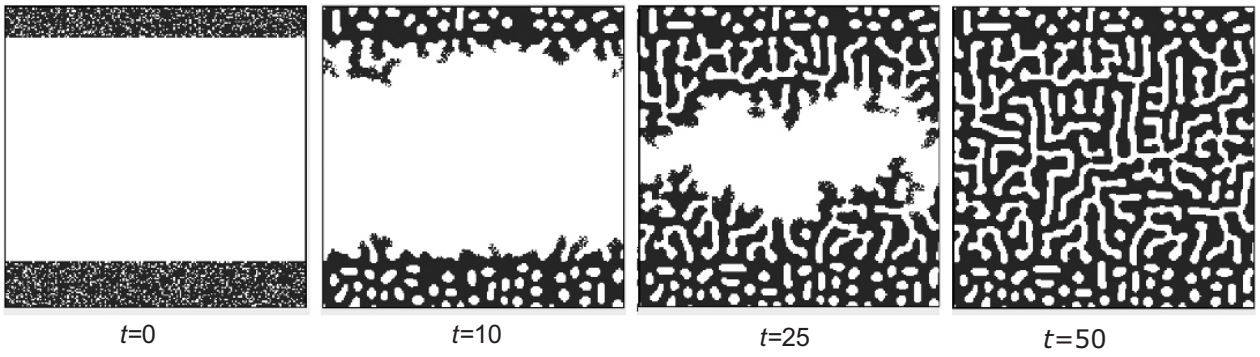

Рис. 1. Эволюция простого КА, моделирующего процесс формирования устойчивых структур типа «полоски»

Эволюции этого КА при работе в синхронном и асинхронном режимах имеют один и тот же поведенческий характер, причем асинхронный режим приводит к устойчивому состоянию за меньшее число итераций.

2. Параллельная композиция КА

Параллельная композиция [15] двух КА $\aleph_1 = \langle A_1, X_1, \Theta_1\rangle$ и $\aleph_2 = \langle A_2, X_2, \Theta_2\rangle$ предполагает, что каждый КА функционирует на своем собственном клеточном поле, имеет свой собственный алфавит состояний, а между множествами их имен X_1 и X_2 существует взаимно-однозначное соответствие.

Композицию могут составлять только КА с одинаковыми режимами работы: либо оба синхронные, либо оба асинхронные. В синхронном случае смена состояний автоматов происходит в обоих КА одновременно, обозначая переход к следующей итерации. В асихронной композиции клетки каждого КА меняют свои состояния независимо от своего окружения, что дает возможность работать компонентным КА с произвольными скоростями. Однако «разноскоростные» композиции пока не исследованы. Принято считать, что на одном отрезке времени в каждом КА происходит одинаковое количество применений локальных операторов.

В [15] введено два типа параллельной композиции КА: однонаправленная и двунаправленная.

3. Однонаправленная параллельная композиция

В однонаправленной композиции один из КА, пусть $\aleph_1$, является основным. Он имитирует исследуемый процесс. Второй КА $\aleph_2$ играет роль контекста — он эволюционирует автономно, влияя на работу первого. В функции переходов локального оператора Θ_2 аргументами являются состояния клеток из Ω_2, а в функции переходов Θ_1 — состояния клеток из $\Omega_2 \cup \Omega_1$. Соответственно шаблоны локальных конфигураций принадлежат разным множествам имен:

$$\forall x \in X_1\ (T_1(x) \subseteq X_2 \cup X_1), \quad \forall x \in X_2\ (T_2(x) \subseteq X_2). \tag{9}$$

Однонаправленная композиция двух КА имеет эволюцию, формирующую устойчивые структуры, если этим свойством обладает основной КА. Контекстный КА играет роль управления, которое может динамически изменять параметры основного как во времени, так и в пространстве. Это дает возможность существенно увеличить разнообразие получаемых структур.

Пример 2. Примером однонаправленной асинхронной композиции может служить КА-модель коагуляции некоторого вещества, наличие которого в точке, соответствующей клетке x, имитируется состоянием $u_x = 1$, а отсутствие — состоянием $u_x = 0$. На процесс воздействует тепловое поле, конфигурация которого представляет собой холодный квадрат в центральной части и сильно нагретую область вокруг него (рис. 2). Температура в каждой точке имитируется осредненным значением состояния в соответствующей клетке. Коагуляцию моделирует асинхронный КА $\aleph_u = \langle A_u, X_u, \Theta_u\rangle$, а изменения теплового поля — асинхронный КА диффузии $\aleph_v = \langle A_u, X_u, \Theta_u\rangle$. При этом $A_u = A_v = \{0, 1\}$, $X_v = X_u = \{(i, j) : i, j = 0, \ldots, 300\}$.

КА коагуляции $\aleph_u$ принадлежит к классу асинхронных КА, моделирующих процесс формирования устойчивых образов, его локальный оператор такой же, как (7), но отличается значениями ингибиторов b, которые равны осредненным значениям состояний в соответствующих клетках $\Omega_v(t)$, т. е.

$$b = \langle v\rangle = (2r+1)^{-1} \sum_{g,h=-r}^{r} v_{i+g,j+h}, \tag{10}$$

где $r = 10$ — радиус осреднения. $\aleph_v$ — асинхронный КА, моделирующий так называемую *наивную диффузию* [16]. Этот КА работает независимо от $\aleph_u$, имитируя распро-

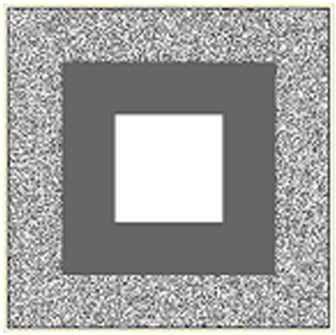

Рис. 2. Начальная глобальная конфигурация контекстного КА $\Omega_v(0)$

странение тепла от горячей области ($v = 1$) к холодному квадрату в центре ($v = 0$) и к теплой области по краям, где плотность распределения единиц равна $\langle v \rangle = 0{,}5$ (рис. 3).

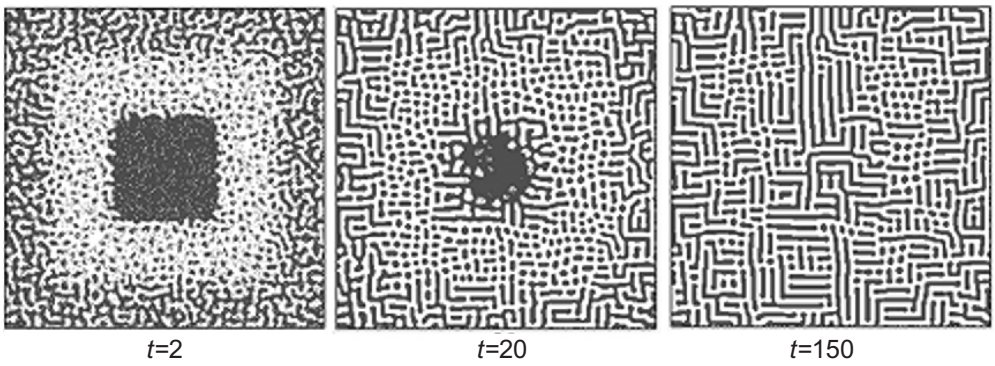

Рис. 3. Эволюция основного КА $\aleph_u$ вида (7), в котором значения ингибиторов управляются распространяющимся теплом, моделируемым контекстным КА $\aleph_v$ с начальным значением $\Omega_v(0)$, показанным на рис. 2

Локальный оператор КА $\aleph_v$ равен $\Theta_v = \{\theta_v\}$, где

$$\theta_v : \{v_{i,j},\ v_{\varphi_1(i,j)},\ v_{\varphi_2(i,j)},\ v_{\varphi_3(i,j)},\ v_{\varphi_4(i,j)}\} \to \{v'_{i,j},\ v'_{\varphi_1(i,j)},\ v'_{\varphi_2(i,j)},\ v'_{\varphi_3(i,j)},\ v'_{\varphi_4(i,j)}\} \quad (11)$$

с функцией перехода

$$\begin{aligned} v'_0 &= v_{\varphi_k(i,j)}, \quad \text{если} \quad 0{,}25k < \texttt{rand} < 0{,}25(k+1); \\ v'_k &= \begin{cases} v_0, & \text{если} \quad 0{,}25k < \texttt{rand} < 0{,}25(k+1), \\ v_{\varphi_k(i,j)} & \text{иначе,} \end{cases} \qquad k = 1, \ldots, 3. \end{aligned} \quad (12)$$

В процессе эволюции контекстного КА плотность состояний в нем становится равномерной (температура выравнивается) и система приходит в устойчивое состояние, в котором основной массив имеет вид горизонтальных и вертикальных полосок (рис. 3).

4. Двунаправленная параллельная композиция

В двунаправленной композиции $\aleph_1$ и $\aleph_2$ работают в тесном взаимодействии, причем каждый играет роль активатора для себя и ингибитора для другого, но ни один из них сам по себе не должен быть самоорганизующимся КА. В функциях перехода их

локальных операторов Θ_1 и Θ_2 контекстными локальными конфигурациями являются состояния клеток из $\Omega_2 \cup \Omega_1$. Соответственно для шаблонов локальных конфигураций выполняется

$$\forall x \in X_1 \ (T_1(x) \subseteq X_1 \cup X_2), \quad \forall x \in X_2 \ (T_2(x) \subseteq X_2 \cup X_1). \tag{13}$$

Пример 3. Типичным примером двунаправленной композиции может служить задача установления баланса в системе типа «хищник — жертва». Задача интерпретируется следующим образом. На площади, представленной дискретным пространством, обитают два типа живых существ: хищник и жертва. Хищник питается жертвой. Если для хищника имеется достаточно пищи, то его плотность увеличивается с вероятностью, пропорциональной количеству сытых хищников (только сытые хищники размножаются). Если пищи не хватает, то плотность хищников уменьшается. Жертва всегда стремится размножиться. Обе популяции перемещаются (диффундируют), причем хищник более подвижен, поэтому характеризуется коэффициентом диффузии, много большим, чем жертва ($d_v/d_u = K$, $K > 10$). Пусть $\aleph_v = \langle A_v, X_v, \Theta_v \rangle$ моделирует поведение хищника, а $\aleph_u = \langle A_u, X_u, \Theta_u \rangle$ — жертвы, причем состояния клеток $(i,j)_v \in X_v$ обозначаются символами $v_{i,j}$, а состояния клеток $(i,j)_u \in X_u$ — символами $u_{i,j}$. Локальные операторы содержат по две подстановки: $\Theta_v = \{\theta_{v1}, \theta_{v2}\}$, $\Theta_u = \{\theta_{u1}, \theta_{u2}\}$. В обоих случаях первые подстановки θ_{v1} и θ_{u1} моделируют диффузию, используя КА из примера 1 с локальным оператором вида (11). Разница в коэффициентах диффузии реализуется путем разного количества применений θ_{v1} и θ_{u1} в течение одной итерации. Вторые подстановки моделируют взаимоотношения хищника и жертвы, которые зависят от состояний локальных конфигураций в обоих клеточных массивах:

$$\begin{array}{ll} \theta_{v2}: & \{v_{i,j}\} \cup (V_v''(i,j) \cup V_u''(i,j)) \to \{v'_{i,j}\}, \\ \theta_{u2}: & \{u_{i,j}\} \cup (V_v''(i,j) \cup V_u''(i,j)) \to \{u'_{i,j}\}, \end{array} \tag{14}$$

где контексты $V_v''(i,j)$ и $V_u''(i,j)$ содержат состояния клеток из

$$T''(i,j) = \{(i+g, j+h) : g, h = -10, \ldots, -1, 1, \ldots, 10\},$$

а новые состояния являются значениями вероятностных функций переходов

$$\begin{array}{l} v'_{i,j} = \left\{ \begin{array}{lll} 0, & \text{если} \ \langle u_{i,j} \rangle > \langle v_{i,j} \rangle & \& \ \texttt{rand} < p_{v\to 0}, \\ 1, & \text{если} \ \langle v_{i,j} \rangle > \langle u_{i,j} \rangle & \& \ \texttt{rand} < p_{v\to 1}, \end{array} \right. \\ u'_{i,j} = \left\{ \begin{array}{lll} 0, & \text{если} \ \langle u_{i,j} \rangle < \langle v_{i,j} \rangle & \& \ \texttt{rand} < p_{u\to 0}, \\ 1, & \text{если} \ \langle u_{i,j} \rangle > \langle v_{i,j} \rangle & \& \ \texttt{rand} < p_{u\to 1}, \end{array} \right. \end{array} \tag{15}$$

где $\langle y \rangle$ — осредненное по (10) значение y.

Значения вероятностей в (15) определяются исходя из следующих соображений. Если $\langle v_{i,j} \rangle > \langle u_{i,j} \rangle$, то плотность хищников уменьшится на $\langle v_{i,j} \rangle - \langle u_{i,j} \rangle$ из-за нехватки пищи. Эта разность равна вероятности того, что $v'_{i,j} = 0$. Если же $\langle v_{i,j} \rangle < \langle u_{i,j} \rangle$, то пищи достаточно всем, и хищник увеличивает свою плотность в соответствии с функцией размножения $F_v(\langle v_{i,j} \rangle) = c \cdot \langle v_{i,j} \rangle (1 - \langle v_{i,j} \rangle)$ [17]. Аналогично, если $\langle v_{i,j} \rangle > \langle u_{i,j} \rangle$, то жертва поедается хищником, ее плотность уменьшается с вероятностью, равной плотности хищника. В противоположном случае остаток жертвы размножается с вероятностью,

равной $\langle u_{i,j}\rangle - \langle v_{i,j}\rangle$. Таким образом,

$$\begin{array}{lll} p_{v\to 0} = & (\langle v_{i,j}\rangle - \langle u_{i,j}\rangle)/\langle v_{i,j}\rangle, & \text{если} \quad \langle v_{i,j}\rangle > \langle u_{i,j}\rangle, \\ p_{v\to 1} = & 0{,}5\langle v_{i,j}\rangle(1 - \langle v_{i,j}\rangle), & \text{если} \quad \langle v_{i,j}\rangle < \langle u_{i,j}\rangle, \\ p_{u\to 0} = & \langle v_{i,j}\rangle, & \text{если} \quad \langle v_{i,j}\rangle > \langle u_{i,j}\rangle, \\ p_{u\to 1} = & \langle u_{i,j}\rangle - \langle v_{i,j}\rangle, & \text{если} \quad \langle v_{i,j}\rangle < \langle u_{i,j}\rangle. \end{array} \tag{16}$$

Композиция $\aleph_v$ и $\aleph_u$ функционирует в соответствии со следующей итерационной процедурой:

1) подстановка θ_{v1} применяется $K \times d_v$ раз, всякий раз к случайно выбранной клетке из X_v;

2) подстановка θ_{u1} применяется $K \times d_u$ раз, всякий раз к случайно выбранной клетке из X_u;

3) случайно выбирается клетка из X_v, к ней применяется θ_{v2};

4) случайно выбирается клетка из X_u, к ней применяется θ_{u2}.

Эти действия повторяются до тех пор, пока не достигается устойчивое состояние. На рис. 4 приведены три глобальных состояния в эволюции $\aleph_v$. В исходном состоянии ($t = 0$) плотность жертвы на всем пространстве равна $\rho_u = 0{,}5$, а хищник плотно сосредоточен в средней полосе $\langle v_{i,j}\rangle = 1$.

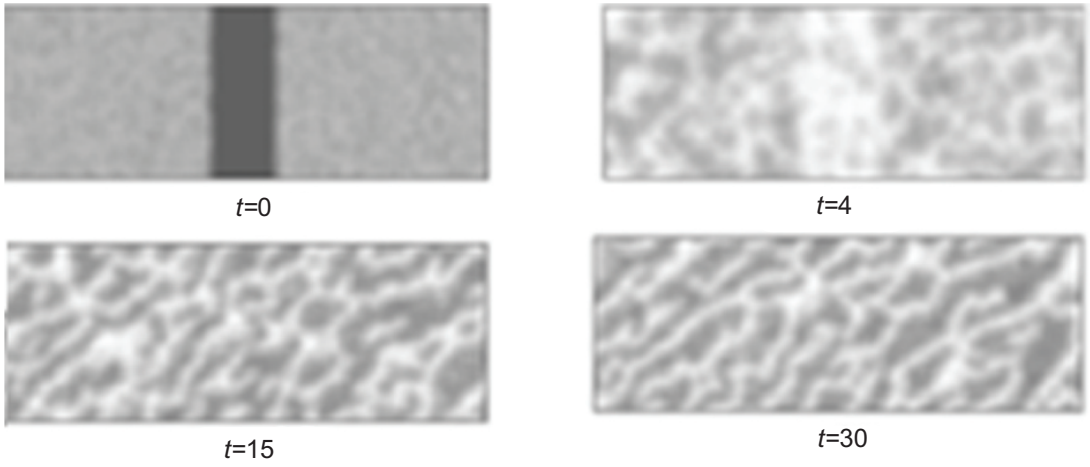

Рис. 4. Три глобальных состояния в эволюции КА $\aleph_v$. При $t > 30$ изменений Ω_v и Ω_u не наблюдается

Моделирование выполнялось на компьютере Intel® Core™ i7. Результаты показали, что представленная выше система «жертва — хищник» быстро самоорганизуется и очень устойчива. Все начальные конфигурации, имеющие в каждом КА даже самые малые плотности и по-разному распределенные, эволюционировали к одному и тому же устойчивому состоянию (рис. 4, $t = 30$).

Заключение

Предложен метод моделирования нелинейных процессов типа «реакция — диффузия» при помощи параллельной композиции клеточных автоматов. Рассмотрено два случая: однонаправленная параллельная композиция, которая позволяет ввести динамическое управление моделируемым процессом, и двунаправленная параллельная композиция, которая моделирует процессы формирования самоорганизующихся структур. Результаты позволяют надеяться, что предложенный подход будет полезен в создании систематической методологии синтеза самоорганизующихся КА.

ЛИТЕРАТУРА

1. *Nicolis G., Prigogine I.* Self-Organization in Nonequilibrium Systems. N.Y.: Wiley-Interscience, 1977. 236 p.
2. *Жаботинский А. М.* Периодические процессы окисления малоновой кислоты в растворе (исследование кинетики реакции Белоусова) // Биофизика. 1964. Т. 9. С. 306–310.
3. *Deutsch A., Dormann S.* Cellular Automaton Modeling of Biological Pattern Formarion. Berlin: Birkhäuser, 2004. 330 p.
4. *Turing A. M.* The chemical basis of Morphogenesis // Phil. Trans. R. Soc. London. 1952. V. B 237. No. 641. P. 37–82.
5. *Ванаг В. К.* Диссипативные структуры в реакционно-диффузионных системах. М.; Ижевск: Ин-т компьютерных исследований, 2008. 300 с.
6. *Wolfram S.* A new kind of science. USA: Wolfram Media Inc., 2002. 1197 p.
7. *Chua L.* CNN: A paradigm of complexity. Singapore: World Scientific, 2002. 320 p.
8. *Elokhin V. I., Latkin E. I., Matveev A. V., Gorodetskii V. V.* Application of Statistical Lattice Models to the Analysis of Oscillatory and Autowave Processes on the Reaction of Carbon Monoxide Oxidation over Platinum and Palladium Surfaces // Kinet. Catalys. 2003. V. 4. No. 5. P. 672–700.
9. *Achasova S., Bandman O., Markova V., Piskunov S.* Parallel Substitution Algorithm. Theory and Application. Singapore: World Scientific, 1994. 180 p.
10. Simulating Complex Systems by Cellular Automata / eds. A. G. Hoekstra, J. Kroc, P. M. A. Sloot. Berlin: Springer, 2010. 350 p.
11. *Toffolli T.* Cellular Automata as an Alternative to (rather than Approximation of) Differential Equations in Modeling Physics // Physica D. 1984. V. 10. P. 117–127.
12. *Wolfram S.* Statistical mechanics of Cellular automata // Rev. Mod. Phys. 1993. V. 55. P. 607–640.
13. *Бандман О. Л.* Клеточно-автоматные модели пространственной динамики // Системная информатика: Сб. научн. тр. Новосибирск: Изд-во СО РАН, 2006. Вып. 10. С. 59–111.
14. *Бандман О. Л.* Дискретное моделирование физико-химических процессов // Прикладная дискретная математика. 2009. № 3(5). С. 33–49.
15. *Бандман О. Л.* Методы композиции клеточных автоматов для моделирования пространственной динамики // Вестник Томского госуниверситета. Приложение. 2002. № 9(1). С. 188–192.
16. *Toffolli T., Margolus N.* Cellular Automata Machine. USA: MIT Press, 1987. 280 p.
17. *Свирежев Ю. М.* Нелинейные волны, диссипативные структуры и катастрофы в экологии. М.: Наука, 1987. 320 с.

DOI 10.17223/20710410/10/11

УДК 004.007.519-7

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ ПОРШНЯ В ГАЗОВОЙ СРЕДЕ КЛЕТОЧНЫМ АВТОМАТОМ[1]

Ю. Г. Медведев

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Россия

E-mail: medvedev@ssd.sscc.ru

Обсуждаются способы задания граничных условий в многочастичной клеточно-автоматной модели потока FHP-MP. Особое внимание уделено нестационарным граничным условиям. Изложен метод перемещения стенок в исследуемой модели. Он позволяет впервые в клеточно-автоматном моделировании потоков жидкости и газа использовать препятствия, движущиеся в процессе моделирования. Выводы подкреплены компьютерными экспериментами на примере задачи моделирования движения поршня в цилиндре, позволяющими заключить, что моделируемый процесс качественно соответствует законам физики.

Ключевые слова: *клеточный автомат, поток газа, нестационарные граничные условия, движение поршня.*

Введение

Класс клеточных автоматов, используемый для моделирования процессов газовой и гидродинамики, называется Lattice Gases [1], или, как пишут в русскоязычной литературе, «решёточные газы». В рамках настоящей работы этот класс назван газодинамическими клеточными автоматами. Впервые подобный автомат, именуемый по инициалам создателей HPP, был предложен Харди и др. в 70-х годах прошлого века [2]. Его квадратные клетки имеют по четыре соседа, а поведение детерминировано. В силу небольшого числа правил столкновения он обладает очень узкими границами применимости и для моделирования потоков практически не используется. Спустя десять лет Фриш и др. предложили другой клеточный автомат — FHP, ставший основой для многих последующих моделей клеточно-автоматной газодинамики [3]. Этот автомат использует достаточно богатый набор вероятностных правил столкновения. Каждая его клетка является правильным шестиугольником и имеет шесть соседей. Еще через пятнадцать лет был предложен трехмерный газодинамический клеточный автомат [4, 5]. Он был назван RD, так как его элементарный объем имеет форму ромбического додекаэдра. В этом многограннике двенадцать граней, следовательно, каждая клетка имеет двенадцать соседей. Поведение этого автомата вероятностное.

Граничные условия в газодинамических клеточных автоматах задаются путём назначения некоторым (граничным) клеткам специальных правил столкновения. Например, все частицы, попадающие в один из граничных типов клеток — стенки, отскакивают назад. Из таких клеток-стенок выстроены препятствия, которые поток вынужден обтекать. Также их обычно используют для ограждения моделируемого объема по периметру. При осреднении значений оказывается, что скорость потока на границах таких препятствий равна нулю, что совпадает с физическими представлениями о потоке.

[1]Работа выполнена при поддержке интеграционного проекта СО РАН №32, 2010 г. и проекта 14-6 программы Президиума РАН, 2010 г.

Все три клеточных автомата — HPP, FHP и RD — оперируют над булевыми векторами состояний клеток. Каждый разряд вектора состояния обозначает наличие или отсутствие в клетке частицы единичной массы с единичной скоростью, направленной к одному из соседей, сопоставленному этому разряду. Очевидно, в одной клетке не могут одновременно оказаться несколько частиц с одинаковыми векторами скорости. Это обстоятельство делает невозможным перемещать стенки в процессе вычислительного эксперимента, не нарушая при этом простых физических принципов, положенных в основу правил взаимодействия частиц с препятствиями и друг с другом.

Примеры коллизий, получающихся при попытке перемещения стенки, приведены на рис. 1. На рис. 1, *а* показано исходное состояние фрагмента клеточного автомата модели FHP. Четыре клетки с фоном из кирпичиков и координатами $(i, 1)$ — стенки. На рис. 1, *б* изображено состояние после перемещения стенки вправо вдоль жирной стрелки в позицию с координатами $(i, 2)$. Чтобы корректно моделировать движущееся препятствие, например поршень, клетки-стенки должны толкать перед собой все частицы газа, встречающиеся на их пути. Это легко получилось у клетки (1, 1) при перемещении в (1, 2) — в этих двух клетках присутствует всего лишь одна частица газа, которой не с кем вступать в коллизию. Другое дело в клетках (2, 1) и (2, 2) — в них пять частиц. Но так как среди них нет ни одной пары с сонаправленными коллинеарными векторами скорости, перемещение стенки также проходит без коллизий. В клетках (3, 1) и (3, 2) по одной частице. Но так как векторы скорости этих частиц направлены в одну и ту же сторону, происходит коллизия — после перемещения стенки обе эти частицы оказываются в клетке (3, 2), и каждая из них требует хранить себя в единственном соответствующем этому направлению разряде булева вектора состояния, что, разумеется, невозможно. При движении стенки из (4, 1) в (4, 2) также происходит коллизия, но это уже не важно.

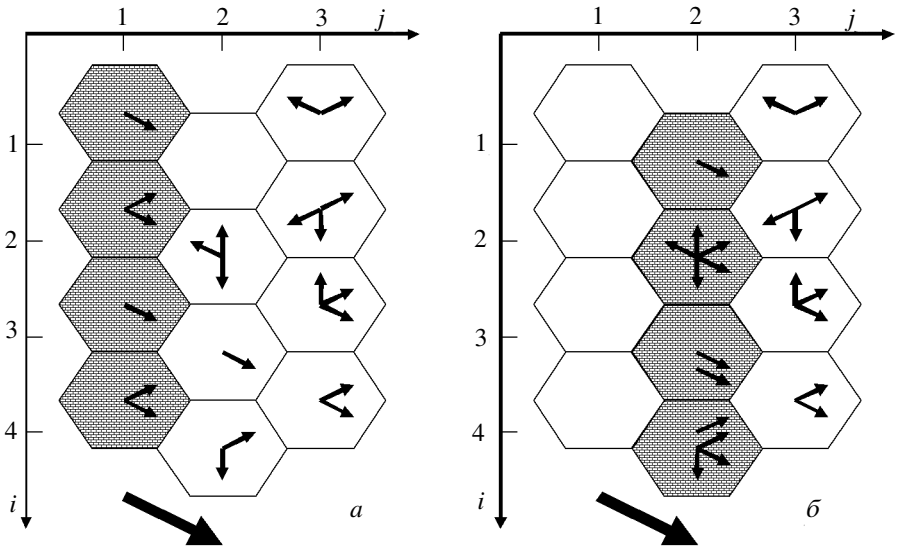

Рис. 1. Коллизии при перемещении стенок в модели FHP с булевым алфавитом

Всё выглядит так, что традиционные газодинамические клеточные автоматы с булевыми векторами состояний не могут моделировать потоки с нестационарными краевыми условиями, т. е. с движущимися стенками. Эта способность принесена в жертву той невероятной скорости, с которой они работают, особенно на современных суперкомпьютерах.

Совсем недавно была разработана новая модель потока FHP-MP — газодинамический клеточный автомат с клетками-шестиугольниками, как у FHP, но использующий целые числа, по одному на каждое из шести направлений, для хранения количества частиц в клетке, имеющих соответствующие векторы скорости [6]. И хотя этот автомат требует значительно большего времени счета, чем FHP, в нем можно избежать коллизий, возникающих в традиционных клеточных автоматах при попытках перемещать стенки. Следует заметить, что хотя возможность перемещения стенок в модели FHP-MP продекларирована, исследований этого до сих пор не проводилось.

В настоящей работе кратко представлен газодинамический клеточный автомат FHP-MP, особое внимание уделено описанию граничных условий, изложен метод задания нестационарных краевых условий, приведены результаты компьютерного моделирования на примере движения поршня в цилиндре, произведено сравнение качественных характеристик полученных результатов с известными физическими законами.

1. Основные определения

Многочастичный клеточный автомат FHP-MP — это тройка объектов $\langle \mathrm{K}, N, \Theta \rangle$, где $\mathrm{K} = \{c_1, c_2, \ldots, c_i, \ldots\}$ — множество клеток, заданное их индексами i и j в некотором дискретном пространстве (см. рис. 1). Для каждой клетки $c_{ij} \in \mathrm{K}$ определено упорядоченное множество соседства $N(c_{ij}) = \{n_k(c_{ij}) \in \mathrm{K} : k = 0, 1, \ldots, 6\}$, определяемое следующим образом:

$$N(c_{ij}) = \begin{cases} \{c_{i,j}, c_{i-1,j}, c_{i-1,j+1}, c_{i,j+1}, c_{i+1,j}, c_{i+1,j-1}, c_{i,j-1}\} & \text{при нечётном } j, \\ \{c_{i,j}, c_{i-1,j}, c_{i,j+1}, c_{i+1,j+1}, c_{i+1,j}, c_{i+1,j-1}, c_{i-1,j-1}\} & \text{при чётном } j. \end{cases} \tag{1}$$

Каждая клетка $c \in \mathrm{K}$ характеризуется состоянием $\mathbf{s}(c)$, которое зависит от дискретного времени t и представляет собой вектор, имеющий целочисленные компоненты $s_k(c)$, $k = 0, 1, \ldots, 6$, указывающие на количество модельных частиц газа в клетке c с единичной массой и вектором скорости $\mathbf{e}_k(c)$, направленным в сторону соседа $n_k(c)$. Вектор скорости $\mathbf{e}_0(c)$ имеет нулевую длину, поэтому частицы $s_0(c)$ принято называть частицами покоя. Остальные векторы имеют единичную длину, так что частицы в каждом направлении несут импульс $\mathbf{p}_k(c) = s_k(c)\,\mathbf{e}_k(c)$. Множество состояний $\mathbf{s}(c)$ всех клеток $c \in \mathrm{K}$ в один и тот же момент времени t называется глобальным состоянием $\Omega(t) = \{\mathbf{s}(c_{ij}) : \forall i, j\}$ клеточного автомата.

Для каждой клетки $c_{ij} \in \mathrm{K}$ определены координаты её центра $x(c_{ij})$ и $y(c_{ij})$ на декартовой плоскости следующим образом:

$$\begin{aligned} x(c_{ij}) &= \begin{cases} i & \text{при нечётном } j, \\ i + \dfrac{1}{2} & \text{при чётном } j; \end{cases} \\ y(c_{ij}) &= \frac{\sqrt{3}}{2} j. \end{aligned} \tag{2}$$

Расстояние $d(c_{i_1 j_1}, c_{i_2 j_2})$ между клетками $c_{i_1 j_1} \in \mathrm{K}$ и $c_{i_2 j_2} \in \mathrm{K}$ обусловлено соотношением

$$d^2(c_{i_1 j_1}, c_{i_2 j_2}) = \left(x(c_{i_1 j_1}) - x(c_{i_2 j_2})\right)^2 + \left(y(c_{i_1 j_1}) - y(c_{i_2 j_2})\right)^2. \tag{3}$$

Осредненные значения скорости $\mathbf{u}$ являются модельными значениями скорости потока, а осредненные значения плотности ρ являются модельными значениями давления. Они вычисляются для некоторой окрестности $Av(c) = \{c_{ij} \in \mathrm{K} : d(c_{ij}, c) \leqslant r\}$, где

r — радиус осреднения. Осредненная скорость вычисляется как отношение суммарного импульса всех частиц в клетках $c_{ij} \in Av\,(c)$ к суммарной массе этих частиц:

$$\mathbf{u}\,(c) = \frac{\sum\limits_{c_{ij} \in Av(c)} \sum\limits_{k=0}^{6} \mathbf{p}_k\,(c_{ij})}{\sum\limits_{c_{ij} \in Av(c)} \sum\limits_{k=0}^{6} s_k\,(c_{ij})}. \tag{4}$$

Осредненная плотность частиц подсчитывается в той же окрестности следующим образом:

$$\rho\,(c) = \frac{1}{|Av\,(c)|} \sum_{c_{ij} \in Av(c)} \sum_{k=0}^{6} s_k\,(c_{ij}), \tag{5}$$

где $|Av(c)|$ — мощность окрестности осреднения $Av(c)$.

2. Стационарные граничные условия в модели FHP-MP

Стационарные граничные условия задаются разбиением клеток $c \in \mathrm{K}$ на типы. Клетками среды $c \in \mathrm{K}_c$ называются клетки, в которых выполняются законы сохранения массы и импульса. Клетки стенок $c \in \mathrm{K}_w$ — это клетки, в которых выполняется закон сохранения массы, но может нарушаться закон сохранения импульса. И, наконец, источники $c \in \mathrm{K}_s$ — клетки, в которых могут нарушаться как закон сохранения массы, так и закон сохранения импульса.

Клеточный автомат FHP-MP функционирует в синхронном режиме. На каждой итерации происходит смена состояний $s\,(t)$ всех клеток $c \in \mathrm{K}$ на состояния $s\,(t+1) = \theta\,(s\,(t))$, где $\theta\,(s\,(t)) \in \Theta$ — соответствующая функция переходов клетки c. Клеточный автомат при этом переходит из глобального состояния $\Omega(t)$ в новое глобальное состояние $\Omega(t+1)$. Для того чтобы модель адекватно отображала физический процесс, функция θ в клетках среды $c \in \mathrm{K}_c$ должна удовлетворять законам сохранения массы

$$\sum_{c \in \mathrm{K}_c} \sum_{k=0}^{6} \theta\,(s_k\,(c)) = \sum_{c \in \mathrm{K}_c} \sum_{k=0}^{6} s_k\,(c) \tag{6}$$

и импульса

$$\sum_{c \in \mathrm{K}_c} \sum_{i=1}^{6} \theta\,(\mathbf{p}_k\,(c)) = \sum_{c \in \mathrm{K}_c} \sum_{k=1}^{6} \mathbf{p}_k\,(c). \tag{7}$$

Каждая итерация выполняется за две стадии: сдвиг и столкновение. Функция переходов θ состоит, таким образом, из суперпозиции функций θ_1 (сдвиг) и θ_2 (столкновение):

$$\theta\,(s) = \theta_2\,(\theta_1\,(s))\,. \tag{8}$$

На стадии сдвига в каждой клетке $c \in \mathrm{K}$ каждая частица, учтённая в компоненте $s_k\,(c)$, $k = 1, \ldots, 6$, вектора состояния $\mathbf{s}\,(c)$ перемещается в соседнюю клетку $n_k(c)$, расположенную на единичном расстоянии в направлении вектора её скорости $\mathbf{e}_k\,(c)$. Частицы покоя, соответствующие компоненте $s_0\,(c)$, остаются в клетке c. Таким образом, k-е компоненты $s_k\,(c)$ вектора состояния $\mathbf{s}\,(c)$ клетки c после сдвига принимают значения

$$\theta_1\,(s_k\,(c)) = \begin{cases} s_k\left(N_{((k+2) \bmod 6)+1}\,(c)\right) & \text{для } k = 1, 2, \ldots, 6, \\ s_k\,(c) & \text{для } k = 0. \end{cases} \tag{9}$$

На стадии столкновения θ_2 в каждой клетке $c \in \mathrm{K}$ происходит изменение направления движения частиц согласно некоторым правилам столкновения, зависящим только от состояния и типа клетки c и не зависящим от состояний соседних клеток из $N(c)$.

В клетках среды $c \in \mathrm{K}_c$ результатом функции θ_2 равновероятно выбирается одно из возможных значений, при которых сохраняются масса и импульс:

$$\begin{aligned} \sum_{k=0}^{6} \theta_2\left(s_k\left(c\right)\right) &= \sum_{k=0}^{6} s_k\left(c\right),\ c \in \mathrm{K}_c, \\ \sum_{k=1}^{6} \theta_2\left(\mathbf{p}_k\left(c\right)\right) &= \sum_{k=1}^{6} \mathbf{p}_k\left(c\right),\ c \in \mathrm{K}_c. \end{aligned} \tag{10}$$

В клетках $c \in \mathrm{K}_w$, являющихся стенками, частицы «отражаются» в обратном направлении, нарушая при этом закон сохранения импульса:

$$\theta_2\left(s_k\left(c\right)\right) = \begin{cases} s_{((k+2) \bmod 6)+1}\left(c_w\right) & \text{для } k = 1, 2, \ldots, 6, \\ s_k\left(c_w\right) & \text{для } k = 0, \end{cases} \quad c \in \mathrm{K}_w. \tag{11}$$

Такое поведение частиц в клетках-стенках моделирует условие нулевой скорости потока на границах препятствий.

Каждая клетка-источник $c \in \mathrm{K}_s$ по алгоритму, определяемому граничными условиями, генерирует или поглощает (генерирует отрицательное количество) частицы с определёнными направлениями вектора скорости. Из клеток-источников можно создавать различные объекты, задавая различные способы введения газа в моделируемый объём. Например, установив такие клетки в пространстве в одну линию (как правило, у границы клеточного массива), можно получить источник равномерного потока частиц заданной плотности. Отдельно установленный источник будет моделировать форсунку. В компьютерных экспериментах, описанных ниже, используется алгоритм функционирования источников, поддерживающих заданное (например, атмосферное) давление:

$$\theta_2\left(s_k\left(c\right)\right) = \rho_k,\ c \in \mathrm{K}_s. \tag{12}$$

Граница клеточного автомата, выстроенная из таких клеток, моделирует открытый в атмосферу край объекта.

3. Нестационарные граничные условия в модели FHP-MP

Подробно осветить все многообразие нестационарных граничных условий в рамках одной статьи не представляется возможным. Вместе с тем все способы изменения границ похожи друг на друга, поэтому ниже изложен принцип изменяющихся во времени граничных условий на примере движения стенки по определенному закону. В простейшем случае это поступательное движение поршня вдоль одной из координатных осей без ускорения, в том числе и без учёта влияния давления газа на поршень.

В случае нестационарных граничных условий функция переходов θ состоит из композиции трёх функций θ_1 (сдвиг), θ_2 (столкновение) и θ_3 (перемещение границ):

$$\theta\left(s\right) = \theta_3\left(\theta_2\left(\theta_1\left(s\right)\right)^M\right), \tag{13}$$

где $M = \dfrac{|\mathbf{u}_\mathbf{p}|}{u_s}$ — отношение абсолютной величины скорости движения поршня к скорости звука в газе в модельных единицах. Фактически, M является модельной величиной, обратной числу Маха. Итерацией здесь будем называть однократное выполнение

$\theta_2(\theta_1(s))$. Стадии сдвига и столкновения проходят так же, как и со стационарными граничными условиями.

На стадии перемещения границ подвижная стенка (поршень) перемещается на одну клетку на каждой M-й итерации. При этом через поршень не должно просочиться ни одной частицы. На рис. 2, *а* изображен фрагмент клеточного автомата до перемещения поршня, располагающегося в столбце $j = 1$ с некоторым набором частиц в клетках. В каждом направлении может быть ориентировано несколько $(0, 1, 2, \ldots)$ векторов скорости частиц. На рисунке все такие векторы, сколько бы их ни было, изображены одной стрелкой. На рис. 2, *б* изображен тот же фрагмент после перемещения поршня на одну клетку вправо. Клетки столбца $j = 1$ стали клетками среды, клетки столбца $j = 2$ — стенками. Частицы столбца $j = 2$ переместились в столбец $j = 3$, добавившись к тем частицам, которые уже там были, а на их место переместились частицы столбца $j = 1$. Коллизий не возникло. Таким образом, функция перемещения границ имеет вид

$$\begin{aligned}
&\theta_3(s_k(c_{i,3})) = s_k(c_{i,3}) + s_k(c_{i,2}),\\
&\theta_3(s_k(c_{i,2})) = s_k(c_{i,1}),\\
&\theta_3(s_k(c_{i,1})) = 0,\\
&c_{i,2} \text{ переходят из } \mathrm{K}_c \text{ в } \mathrm{K}_w,\\
&c_{i,2} \text{ переходят из } \mathrm{K}_w \text{ в } \mathrm{K}_c.
\end{aligned} \tag{14}$$

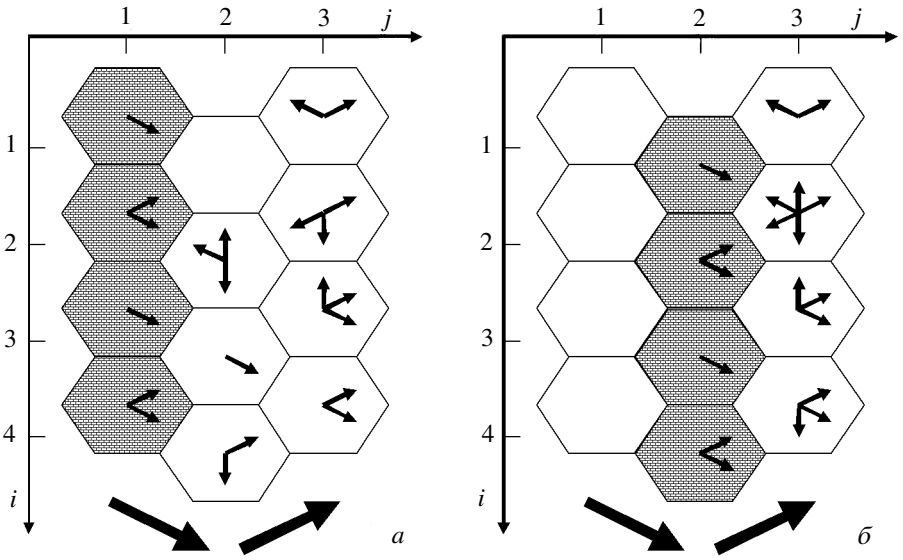

Рис. 2. Перемещение стенки в модели FHP-MP с целочисленным алфавитом

Перемещение частиц из столбца $j = 2$ в столбец $j = 3$ влечет за собой то, что в стенке не окажется ни одной частицы с вектором скорости, направленным влево, вверх или вниз, что, в свою очередь, гарантирует то, что ни одна из частиц не проникнет сквозь поршень. Образовавшийся за поршнем в столбце $j = 1$ вакуум займут частицы, переместившиеся туда слева (при их наличии) за следующие M итераций, пока поршень покоится.

Принцип, изложенный выше, можно распространить и на более сложное перемещение стенок. Скорость перемещения стенки регулируется коэффициентом M. Если направление движения не параллельно ни одной из координатных осей, то перемещение осуществляется чередующимися этапами вдоль проекций на оси. Если движение

не является поступательным — добавляется вращательный компонент, при этом происходит пересчёт координат клеток, являющихся в каждый момент стенками, с округлением до ближайшего целого. Если необходимо промоделировать обратную связь — влияние давления газа на скорость движения поршня, то перед каждым его перемещением вычисляется разность давлений на его противоположные стенки, которая и влияет на изменение скорости поршня в зависимости от инерции. И в каждом случае нужно заботиться о том, чтобы частицы газа не проходили сквозь движущиеся стенки.

4. Компьютерное моделирование

При проведении вычислительных экспериментов ставилась задача посмотреть качественную картину процесса, получив только модельные значения количественных характеристик, не вычисляя их в физических единицах. За единицу длины принято расстояние между соседними клетками. Единицей массы служит масса одной частицы. Скорость потока в точке вычисляется по формуле (4), а давление газа в точке — по формуле (5), так как оно пропорционально плотности модельных частиц. Код программ написан на языке Си. Хранение состояния автомата осуществляется в двумерном массиве. Из-за синхронного режима функционирования приходится использовать два таких массива. Функция сдвига θ_1 использует в качестве аргумента значения из одного из них, а результат помещает в другой. Для записи результата массивы используются поочередно: один на чётных, а другой на нечётных итерациях. Функции столкновения θ_2 и перемещения границ записывают свои результаты в тот же массив, в котором находятся их аргументы, и применяются к каждому из двух массивов поочередно, после того как функция θ_1 запишет в него свой результат.

Программным ограничением является максимальное количество частиц в одной клетке, имеющих одинаковый вектор скорости $\mathbf{e}_k\,(c)$, равное 255, вследствие того, что каждый из семи компонентов $s_k\,(c)$ вектора состояния хранится в одном байте памяти. Еще один байт занимает информация о типе клетки. Таким образом, состояние каждой клетки занимает 16 байт памяти с учетом дублирования во втором массиве. К примеру, клеточный автомат, эксперименты с которым описаны ниже, имеющий размер 100×200 клеток, требует около 0,3 мегабайта памяти.

Моделируемая камера равномерно заполнена частицами газа, по одной частице в каждом направлении (включая частицы покоя). Типы клеток автомата следующие. Верхняя и нижняя границы и препятствия внутри камеры — неподвижные стенки. Левая граница — поршень, который в процессе эксперимента будет двигаться с некоторой скоростью вправо до середины камеры, затем будет покоиться некоторое время, а затем двигаться с той же скоростью влево к своему исходному положению. Правая граница в эксперименте 1 — неподвижные стенки, а в экспериментах 2 и 3 — источники, моделирующие открытый выход в атмосферу.

Реализация функции сдвига θ_1 тривиальна. Она использует второй массив для записи своих значений согласно (9), поэтому при последовательном обходе клеток автомата исходные значения, хранящиеся в первом массиве, не портятся. После того, как эта функция отработала, указатели на массивы обмениваются своими значениями.

Функция θ_2 последовательно перебирает все клетки автомата по следующим правилам. Если клетка имеет тип «стенка», то векторы скорости всех частиц в ней меняются на антипараллельные (11). Клетки-источники поддерживают плотность частиц газа на заданном уровне (12). Клетки среды производят столкновение по следующему алгоритму. Вначале вычисляются концентрация частиц в клетке и проекции их импульсов на декартовы оси. Затем полным перебором подсчитывается количество

состояний, сохраняющих массу и импульс, обусловленные выражениями (10). После этого равновероятно выбирается одно из этих состояний; его и полагают результатом функции θ_2. Из-за полного перебора столкновение является самой затратной вычислительной функцией.

Функция θ_3 производит замену типов клеток по нижеописанному алгоритму, если номер итерации кратен M, или ничего не делает — в противном случае. Во-первых, частицы газа из клеток, куда будет перемещен поршень, перемещаются на одну клетку в направлении движения поршня и с учетом векторов скорости добавляются к тем частицам, которые там уже были. Во-вторых, частицы газа, находящиеся в клетках поршня, перемещаются на одну клетку в направлении движения поршня на освободившееся место. В-третьих, клетки поршня меняют тип с клеток-стенок на клетки среды. И, в-четвертых, клетки, куда переместился поршень, меняют тип с клеток среды на клетки-стенки (14).

После этого происходит следующая итерация и т. д. Значение M для эксперимента 1 выбрано равным 20, а в экспериментах 2 и 3 — равным 10. В каждом эксперименте вначале поршень двигался вправо 100 раз (это соответствует 2000 итерациям в эксперименте 1 и 1000 итерациям в экспериментах 2 и 3). Затем поршень оставался в покое, т. е. следующие 30 перемещений не выполнялись. И, наконец, поршень совершал 100 перемещений влево, возвращаясь в начальное положение.

На рис. 3 приведены поле скорости потока газа, вызванного движением поршня внутри камеры, и поле давления газа в начальном положении, после 70-го перемещения вправо, после времени покоя в правом положении, после 20-го и после 70-го перемещения влево. Длина стрелки пропорциональна скорости потока в точке. Давление проиллюстрировано цветом фона — чем темнее фон, тем выше давление. Белый цвет соответствует вакууму. На рисунке виден поток через диффузор, вызванный изменением давления в связи с перемещением поршня. Сравнивая последнюю строку экспериментов 1 и 2, можно заметить, что скорость потока пропорциональна разности давлений между половинками камеры (качественное совпадение с законом Торричелли). При сравнении предпоследней строки экспериментов 2 и 3 видно, что скорость потока обратно пропорциональна диаметру диффузора (качественное совпадение с законом Пуазейля). Третья строка во всех экспериментах показывает, что давление в правой половинке камеры после времени релаксации распределено приблизительно равномерно (качественное совпадение с законом Паскаля).

Заключение

В работе представлен метод перемещения стенок в модели FHP-MP. Он позволяет впервые в клеточно-автоматном моделировании потоков жидкости и газа использовать препятствия, движущиеся в процессе моделирования. Проведенные эксперименты позволяют заключить, что полученные результаты качественно соответствуют законам Торричелли, Паскаля и Пуазейля.

Дальнейшие усилия в этом исследовании будут направлены на получение количественных характеристик и выведению зависимостей между модельными величинами FHP-MP и соответствующими им физическими величинами.

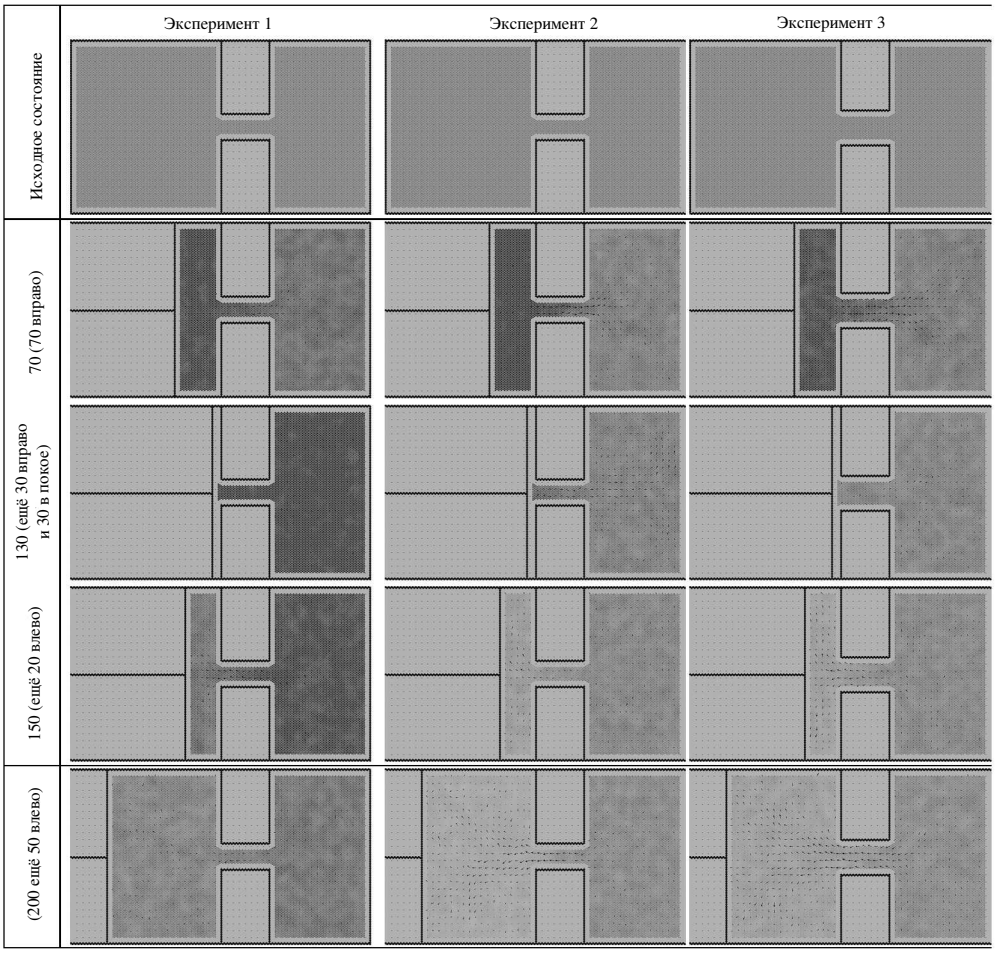

Рис. 3. Результаты компьютерного моделирования. Поля скорости и давления

ЛИТЕРАТУРА

1. *Rothman D. H., Zaleski S.* Lattice-gas cellular automata: simple models of complex hydrodynamics. Cambridge University Press, 1997.
2. *Hardy J., Pomeau Y., and de Pazzis O.* 2D Lattice-Gas mode // J. Math. Phys. 1973. No. 14. P. 1746.
3. *Frisch U., Hasslacher B., and Pomeau Y.* Lattice-Gas automata for Navier-Stokes equations // Phys. Rev. Lett. 1986. No. 56. P. 1505.
4. *Медведев Ю. Г.* Трехмерная клеточно-автоматная модель потока вязкой жидкости // Автометрия. 2003. Т. 39. № 3. С. 43–50.
5. *Бандман О. Л.* Клеточно-автоматные модели пространственной динамики // Системная информатика. 2006. № 10. С. 59–113.
6. *Медведев Ю. Г.* Многочастичная клеточно-автоматная модель потока жидкости FHP-MP // Вестник Томского госуниверситета, сер. «Управление, вычислительная техника и информатика». 2009. № 1(6). С. 33–40.

АНАЛИТИЧЕСКИЕ ОБЗОРЫ

DOI 10.17223/20710410/10/12

УДК 519.7

SIBECRYPT'10. ОБЗОР ДОКЛАДОВ

Г. П. Агибалов

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: agibalov@isc.tsu.ru

Приводится аналитический обзор докладов, представленных на IX Сибирской научной школе-семинаре «Компьютерная безопасность и криптография» — Sibecrypt'10, состоявшейся 7 – 10 сентября 2010 г. в Тюмени.

Ключевые слова: *прикладная дискретная математика, криптография, компьютерная безопасность, стеганография.*

Введение

Sibecrypt — это Сибирская научная школа-семинар с международным участием «Компьютерная безопасность и криптография». Её ежегодно, начиная с 2002 г., организует и в первой трети сентября проводит кафедра защиты информации и криптографии Национального исследовательского Томского государственного университета (г. Томск) в сотрудничестве с кафедрой программирования и компьютерной безопасности Института криптографии, связи и информатики (г. Москва). Среди её участников — учёные всех возрастов и званий — от студента до академика из вузов и научных учреждений страны, ближнего и дальнего зарубежья (России, Укаины, Беларуси, Канады, Франции и др.). Место проведения школы-семинара привязано к территории России от Урала до Байкала. Из года в год она кочует по городам и весям этого гостеприимного края, неся в него свет большой науки и знакомя своих участников с его достопримечательностями. Её уже принимали Томск (ТГУ, 2002, 2003, 2005), Иркутск (ИДСТУ, 2004), Шушенское (ШуБор, 2006), Горно-Алтайск (ГАГУ, 2007), Красноярск (СибГАУ, 2008), Омск (ОмГТУ, 2009). Школа-семинар 9-го созыва (Sibecrypt'10) состоялась 7 – 10 сентября 2010 г. в Тюмени. Тезисы докладов, включённых в её программу, опубликованы в [1]. Обзор их содержания является целью данной статьи.

Кроме докладов, на школе-семинаре Sibecrypt для её участников, а также для сотрудников и студентов принимающей организации (вуза, НИИ) ведущими специалистами в данной области (из числа участников школы-семинара) читаются лекции по современным проблемам компьютерной безопасности и криптографии. На школе-семинаре Sibecrypt'10 были прочитаны следующие лекции:

1. *Пичкур А. Б.*, *Черёмушкин А. В.* (г. Москва). Теоретико-числовые методы в криптографии.

2. *Девянин П. Н.* (г. Москва). Классическая и расширенная модели *Take-Grant*.

3. *Абросимов М. Б.* (г. Саратов). Графовые модели отказоустойчивости вычислительных систем.

4. *Агибалов Г. П.* (г. Томск). О свойстве обратимости с конечной задержкой конечных автоматов.

К сожалению, формат статьи не позволяет представить содержание этих достаточно насыщенных лекций. Можно только сказать, что материал лекций П. Н. Девянина можно найти в последнем его учебнике, материал лекции Г. П. Агибалова готовится к публикаци в журнале ПДМ, а лекции других авторов вошли в их учебные пособия, сданные в издательства.

1. Проблематика исследований, проводимых в России и за рубежом по тематике школы-семинара

Тематика школы-семинара имеет математическую направленность, и её научную основу образует прикладная дискретная математика. В соответствии с этим проблематику исследований, проводимых в России и за рубежом по тематике школы-семинара, составляют проблемы дискретной математики, возникающие в компьютерной безопасности и криптографии. Они распределяются по следующим основным направлениям:

1) теоретические основы прикладной дискретной математики — алгебраические структуры, дискретные функции, комбинаторный анализ, теория чисел, математическая логика, теория информации, системы уравнений над конечными полями и кольцами;

2) математические основы информатики и программирования — формальные языки и грамматики, алгоритмические системы, языки программирования, структуры и алгоритмы обработки данных, теория вычислительной сложности;

3) вычислительные методы в дискретной математике — теоретико-числовые методы в криптографии, вычислительные методы в теории чисел и общей алгебре, комбинаторные алгоритмы, параллельные вычисления, методы дискретной оптимизации, дискретно-событийное и клеточно-автоматное моделирование;

4) математические методы криптографии — синтез криптосистем, методы криптоанализа, генераторы псевдослучайных последовательностей, оценка стойкости криптосистем, криптографические протоколы, математические методы квантовой криптографии;

5) математические методы стеганографии — синтез стегосистем, методы стегоанализа, оценка стойкости стегосистем;

6) математические основы компьютерной безопасности — математические модели безопасности компьютерных систем (КС), математические методы анализа безопасности КС, математические методы синтеза защищенных КС;

7) прикладная теория кодирования — коды для сжатия данных и защиты информации, коды для обнаружения и исправления ошибок, построение оптимальных кодов, анализ свойств кодов;

8) прикладная теория автоматов — автоматные модели сетевых протоколов, криптосистем и управляющих систем, автоматы без потери информации, эксперименты с автоматами, декомпозиция автоматов, автоматные уравнения, клеточные автоматы;

9) логическое проектирование дискретных автоматов — математические модели и методы анализа, синтеза, оптимизации и оценки сложности дискретных автоматов, аппаратная реализация криптоалгоритмов;

10) математические основы надежности вычислительных и управляющих систем (ВиУС) — математические модели функциональной устойчивости ВиУС (к отказам, неисправностям, сбоям, состязаниям, исследованию), математические методы анализа функциональной устойчивости ВиУС, математические методы синтеза функциональ-

но устойчивых ВиУС, математические методы верификации логических схем и программ, математические методы синтеза самопроверяемых и контролепригодных схем;

11) математические основы интеллектуальных систем — базы данных, базы знаний, логический вывод, экспертные системы;

12) прикладная теория графов — графовые модели в информатике и программировании, в компьютерной безопасности, вычислительных и управляющих системах, в интеллектуальных системах.

В разные годы на школе-семинаре представляются разные направления из этого перечня. В 2010 г. на ней были представлены так или иначе все перечисленные направления.

2. Теоретические основы прикладной дискретной математики

В этом направлении, в связи с решением задач криптографии и защиты информации, внимание теоретиков по-прежнему занимают исследования по дискретным функциям, группам и конечным полям.

С использованием линейных комбинаций координатных функций степенных преобразований конечных полей построены классы нелинейных приближений произвольных булевых функций, сформулированы условия на вид преобразования, при которых эти приближения более точные, чем линейные, описаны множества показателей степени преобразования, удовлетворяющих этим условиям, эффективность таких приближений продемонстрирована применительно к бент-функциям, построенным с помощью координатных функций степенных преобразований поля с 2^n элементами (А. В. Иванов, В. Н. Романов). Для произвольной бент-функции и дуальной к ней функции установлено взаимнооднозначное соответствие между множествами подпространств, на которых соответственно эти функции аффинны (Н. А. Коломеец). Предложен новый метод построения бент-функций $g(a_1, a_2, x)$ от $n + 2$ переменных из бент-функций от n переменных $f_0(x), f_1(x), f_2(x), f_3(x)$ как $g(a_1, a_2, x) = f_i(x)$ для $i = a_1 + 2a_2$ (Н. Н. Токарева).

Исследованы оптимальные (максимальные и минимальные) кривые рода 3 над конечным полем с дискриминантом -19 (Е. С. Алексеенко, С. И. Алешников, А. И. Зайцев); приведены их уравнения и показано, что среди них нет одновременно максимальной и минимальной и нет гиперэллиптической.

Отмечены некоторые отличительные особенности дискретного преобразования Фурье в поле комплексных чисел и в конечном поле (А. М. Гришин).

Изучены возможности порождения подстановок множеством J_N полурегулярных инволюций степени $N = 2n$, а также величины $l(G)$ и $d(G)$, представляющие собой минимальное число соответственно первых и необязательно первых слоёв J_N^k, исчерпывающих группу $G \in \{A_N, S_N\}$ (М. Э. Тужилин); показано, в частности, что $J_4^2 = J_4^4 = W_4$ (четверная группа Клейна), $J_N^4 = A_N$ при $N \neq 4$, $<J_N> = A_N$ и $d(A_N) = 1$, $l(A_N) = 4$ при чётном $n \neq 2$, $<J_N> = S_N$ и $d(S_N) = 2$ при нечётном n и, кроме того, $l(S_2) = 2$ и $l(S_N) = 5$ при $n > 1$; описаны цикловые структуры подстановок, не входящих в J_N^3; рассчитаны мощности слоёв J_N^k для $N \leqslant 20$ и указано количество классов сопряжённых элементов в слое J_N^k для $k = 2, 3, 4, 5$.

Криптографические свойства гаммы, вырабатываемой генератором с внешним управлением, в значительной степени зависят от свойств управляющей последовательности (УП), в частности от свойства её h-периодичности относительно разных функций h, определённых на множестве всех слов в ней. Для конкретной функции h это свойство означает существование натуральных v и s, таких, что на множестве слов

длины s, образующих разбиение части УП, начинающейся с её v-го члена, функция h постоянна. Наименьшие такие v и s называются длинами соответственно h-предпериода и h-периода данной УП. В докладе В. М. Фомичёва показано, что: 1) в случае аддитивной h длина h-периода чисто периодической УП делит её период; 2) длина h-периода линейной рекуррентной последовательности над конечным полем P, имеющей максимальный период, совпадает с длиной последнего, если h является частотой m_a символа $a \in P \setminus \{0\}$, набором m частот символов (функцией маркировки) или (в случае $P = \mathrm{GF}(k)$, $k = 2, 3$) суммой wt всех символов в слове и делит её с делителем, делящим $(k-1)/2$, если $P = \mathrm{GF}(k)$, $k > 3$ и $h = \mathrm{wt}$; 3) длина h-периода последовательности де Брёйна порядка n при всех $h \in \{m_0, m_1, m, \mathrm{wt}\}$ равна 2^r, где $r < n$; 4) в классе генераторов гаммы, включающем генераторы «$\delta-\tau$-шагов» и генераторы с перемежающимся шагом, если длина m-периода УП с длиной m-предпериода, равной 0, равна τ, то при некотором $i \in \{0, 1, ..., t-1\}$ символы $\gamma_{i+r\tau}$ гаммы при каждом $r = 0, 1, ...$ линейно выражаются через символы состояния генератора в i-м такте.

3. Математические методы криптографии

Это направление на школе-семинаре представлено методами анализа и синтеза симметричных шифров — блочных, поточных, автоматных.

Важной характеристикой стойкости всякого поточного шифра к той или иной атаке на него является его теоретическая стойкость, называемая также расстоянием единственности, определяемая как наименьшая длина начального отрезка ключевого потока (гаммы), достаточной для достижения успеха этой атаки. В криптоанализе поточных шифров широко распространена атака на поточный шифр с угрозой однозначного предсказания его гаммы, сводящаяся к однозначному определению текущего состояния генератора гаммы. Теоретическая стойкость шифра к такой атаке есть длина L кратчайшего отрезка гаммы, по которому возможно однозначное восстановление того состояния генератора гаммы, в котором он окажется после выработки этого отрезка. Если через N_t обозначить число состояний генератора, в которые он может перейти за $t \geqslant 0$ тактов работы и не может перейти за меньшее число тактов из всевозможных начальных состояний без предшественников (тех состояний, в которые нет переходов из других состояний), и положить $K_t = K - \Sigma N_i$, где K — число всех состояний генератора гаммы и сумма Σ берётся по всем i от 0 до t, то $L \leqslant \min(t_1, P + T)$, P и T — длины соответственно предпериода и периода гаммы, t_1 — наименьшее t, при котором $K_t = 1$. В докладе С. А. Киселёва поставлена задача вычисления чисел N_t для поточного шифра A5/1 и показано, что для этого шифра $N_0 = 3 \cdot 2^{61}$, $N_1 = 13 \cdot 2^{58}$, $N_2 = 334 \cdot 2^{53}$, $N_3 = 2792 \cdot 2^{49}$. Кроме того, доказаны теорема о том, что обратимость функции переходов схемы генератора гаммы из регистров сдвига с обратной связью и обратимой функцией переходов равносильна возможности однозначного определения текущего значения функции управления сдвигом состояний регистров в ней по её состоянию в следующий такт работы, и с помощью этой теоремы — утверждение о невозможности такого определения в схеме A5/1.

Описана в общем виде дифференциальная атака на произвольный итеративный r-раундовый блочный шифр с желаемой вероятностью успеха (А. И. Пестунов): при заданной дифференциальной характеристике первых $r-1$ раундов шифра с разностью Δ_{inp} на входе первого раунда и с разностью Δ_{out} на выходе $(r-1)$-го раунда создаются $G \cdot T$ пар блоков открытого текста и столько же пар соответствующих блоков шифртекста, разбитых на G групп по T пар в каждой; последовательно перебираются ключи последнего, r-го, раунда; на каждом таком ключе k с помощью функции раунда

расшифровываются пары блоков шифртекста в g-й группе для $g = 1, 2, ..., G$, и если в каждой группе хотя бы одна пара блоков шифртекста расшифровывается в блоки с разностью Δ_{out}, то k принимается за истинный ключ последнего раунда шифра. Параметры G и T этой атаки зависят от вероятности используемой дифференциальной характеристики, от параметров шифра и желаемой вероятности её успеха. Приведена таблица, в которой для разных параметров шифра и вероятности успеха не менее 0,99 указаны расчётные значения G и T и сложности данной атаки — количества шифрований, требуемых блоков и объёма памяти.

В последнее время всё больше работ посвящается атакам на шифры, основанным на методе связанных ключей и использующим подходящие слабости алгоритма развёртывания ключа. Об атаке такого рода на шифры с алгоритмами развёртывания ключа из класса алгоритмов, свойственных таким шифрам, как 25-раундовый ГОСТ 28147-89, LOKI89, LOKI91, MMB, TREYFER, KeeLog, сообщено в одном из докладов М. А. Пудовкиной. Трудоёмкость атаки равна трудоёмкости опробования одного раундового ключа.

Ряд атак на ГОСТ 28147-89 с использованием двух или четырёх связанных ключей продемонстрирован в докладе М. А. Пудовкиной и Г. И. Хоруженко. Так, с использованием пары ключей $k, k' \in \{0,1\}^{256}$, связанных соотношением $k \oplus k' = \varepsilon = (e0\dots0e0\dots0e0\dots0e0\dots0e0\dots0)$, где $e = 10\dots0 \in \{0,1\}^{32}$, на основе метода дифференциального криптоанализа и метода связанных ключей находятся раундовые ключи $k_{26}, \dots, k_{32}$, а раундовый ключ k_{25} определяется с помощью методов бумеранга и связанных ключей. В зависимости от свойств блоков замены трудоёмкость атаки лежит в пределах от $2^{26,6}$ до 2^{40}, количество пар блоков открытого текста — в границах между 2^{15} и 2^{29}; вероятность успеха атаки равна 0,98. Описан класс блоков замены, при которых эта атака неприменима. С использованием четвёрки ключей, связанных соотношениями $k \oplus k' = k'' \oplus k''' = \varepsilon$, $k \oplus k'' = k' \oplus k''' = (e0\dots0)$, предложена атака на основе методов связанных ключей, дифференциального криптоанализа и бумеранга с трудоёмкостью нахождения ключа шифрования $2^{44,8}$ шифрований, с числом блоков открытого текста $2^{26,2}$ и с вероятностью успеха 0,99.

Введено семейство блочных симметричных шифров, в которых алгоритмы шифрования и развёртывания ключа имеют структуру алгоритма шифрования криптосистемы Whirlpool (ещё один доклад М. А. Пудовкиной). Построена атака методом дифференциального криптоанализа на 6 раундов произвольного шифра в этом семействе. В ней первые три раунда моделируются трёхраундовой дифференциальной характеристикой шифра, а прохождение последних трёх раундов обеспечивается возможностью вычисления части блока на выходе 3-го раунда по части раундового ключа 5-го раунда, обязанной свойству алгоритма развёртывания ключа. Трудоёмкость нахождения ключа шифрования оценивается сверху числом $2^{236,3}$, вероятность успеха атаки равна 0,9999999993, число используемых блоков открытого текста — $2^{107,3}$, что меньше квадратного корня из числа всех ключей.

Задача восстановления закрытого ключа по открытому в криптосистеме Мак-Элиса на основе кодов Рида–Маллера и в криптосистеме Мак-Элиса — Сидельникова заключается, как известно, в решении матричного уравнения, связывающего операцией умножения неизвестную порождающую матрицу кода с некоторыми другими матрицами, не все из которых известны. В докладе И. В. Чижова доказано, что если данная задача решается за полиномиальное время для криптосистемы Мак-Элиса, то за полиномиальное время решается как сама эта задача для криптосистемы Мак-Элиса — Сидельникова, так и каждая такая её подзадача для криптосистемы Мак-Элиса, в ко-

торой вместо порождающей матрицы кода фигурирует её подматрица, полученная из неё вычёркиванием некоторой строки, и наоборот, если каждая из этих подзадач для криптосистемы Мак-Элиса решается за полиномиальное время, то за полиномиальное время решается и сама задача для криптосистемы Мак-Элиса — Сидельникова.

В докладе И. В. Широкова предложена новая модель симметричного шифра на основе некоммутативной алгебры полиномов, представляющей собой кольцо многочленов над некоторым полем, взятое вместе с дополнительной операцией — композицией многочленов: $(f \circ g)(x) = f(g(x))$. Открытыми параметрами шифра являются некоторые различные многочлены $f_1(x), \ldots, f_n(x)$ степени $\geqslant 2$ и неприводимый многочлен $h(x)$; секретным ключом является некоторая подстановка $\sigma \in S_n$. Открытый текст представляется многочленом $m(x)$, шифртекстом будет многочлен $c(x) = (f_{\sigma(n)} \circ \ldots \circ f_{\sigma(1)} \circ m)(x) \bmod h(x)$, вычисляемый по рекуррентной формуле: $c_0(x) = m(x)$; $c_i(x) = (f_{\sigma(i)} \circ c_{i-1})(x) \bmod h(x)$, $i = 1, \ldots, n$; $c(x) = c_n(x)$. Расшифрование заключается в последовательном решении последней системы уравнений относительно $c_{n-1}, \ldots, c_1, c_0$. Приведены аргументы за то, что наиболее быстрый способ определения ключа атакой с известным открытым текстом состоит в переборе всего ключевого пространства.

Влияние выбора ключа шифрования и блоков замены шифра ГОСТ 28147-89 на вид системы булевых функций, представляющих процедуру одного раунда шифрования, исследовано в докладе В. Ю. Золотухина и Т. А. Чалкина. Экспериментально установлено ожидаемое, а именно: показатели нелинейности и лавинного эффекта системы для каждой таблицы замен с изменением ключа могут и улучшаться, и ухудшаться, и оставаться неизменными.

На основе клеточных автоматов построен высокоскоростной генератор псевдослучайной последовательности (В. М. Сухинин). В его составе — два двумерных булевых клеточных автомата размера 37×11 каждый и регистр сдвига длиной 63 с линейной обратной связью. Функция клетки в каждом автомате своя. Окрестность клетки в автомате состоит из 8 соседних клеток. Выходом каждого автомата служит 256-битное состояние подрешётки размера 32×8. Выход регистра в каждый такт работы прибавляется по модулю 2 к состоянию одной из клеток каждого автомата. Выход генератора является побитной суммой по модулю 2 выходов обоих автоматов. Схема генератора на базе ПЛИС (программируемой логической интегральной схемы) Altera Cyclone II работает с частотой 100 МГц и вырабатывает 23,8 Гбит/с. Путём тестирования генератора на наборе тестов NIST подобраны функции ячеек автоматов такими, что последовательность, вырабатываемая генератором, успешно проходит все тесты из набора. Ведётся работа над программной реализацией генератора на базе графического адаптера ПЭВМ.

В докладе А. В. Милошенко предложена программно-аппаратная реализация симметричной шифрсистемы на основе сильносвязного конечного автомата с функцией выхода, биективной в каждом состоянии, получившего название автомата Закревского в честь А. Д. Закревского, предложившего его в 1959 г. на эту роль. Программная часть реализации включает в себя генератор шифрующих автоматов, генератор ключей — подмножеств переходов автомата, программу кодирования состояний автомата и транслятор с его табличного задания в язык описания аппаратуры VHDL. Аппаратная часть шифрсистемы строится на базе ПЛИС, программирование которой осуществляется с помощью САПР Xilinx ISE. Экспериментальное исследование показало, что по скорости работы и по эффективности использования ресурсов ПЛИС предложенная

реализация автоматной шифрсистемы сравнима с реализациями на ПЛИС других известных блочных шифров (TripleDES, IDEA и т. п.).

Последние два доклада в равной мере относятся и к направлению 9 (логическое проектирование дискретных автоматов), к тому его разделу, где речь идёт об аппаратной реализации криптоалгоритмов.

4. Математические методы стеганографии

Метод выбора элементов стегоконтейнера, модифицируемых в процессе встраивания информации, оказывает критическое влияние как на стойкость стегосистемы, так и на её пропускную способность. Строго говоря, задача состоит в таком выборе элементов контейнера для встраивания информации, который позволил бы максимизировать либо стойкость стегосистемы при заданном размере скрываемого сообщения, либо пропускную способность стегосистемы при заданной стойкости.

В докладе О. В. Моденовой проанализированы существующие методы встраивания информации в файлы формата MPEG-2 — модификация коэффициентов дискретного косинусного преобразования, удаление нескольких из них и встраивание на уровне битовых элементов. Дан сравнительный анализ этих методов как по критериям стойкости и пропускной способности, так и по сложности техники встраивания информации.

В докладе Е. В. Разинкова и Р. Х. Латыпова предложен общий метод распределения скрываемого сообщения в произвольном контейнере, позволяющий повысить пропускную способность стегосистемы при заданной стойкости или повысить стойкость стегосистемы при заданной пропускной способности. Контейнер разбивается на m групп элементов с k_i элементами в i-й группе и с областью C_i допустимых значений элементов в ней, $C_i = \{c_1^i, ..., c_{k_i}^i\}$. Предполагается, что модификация одного элемента в i-й группе позволяет встроить q_i бит, $q_i = \lfloor \log_2 |C_i| \rfloor$. Количество модифицируемых элементов в i-й группе обозначается x_i, $\Sigma x_i q_i = n$; функции плотности распределения элементов i-й группы неизменённого контейнера и стегоконтейнера со встроенной информацией обозначаются $f_i(c)$ и $f_i'(c, x_i)$ соответственно, $f_i'(c, x_i) = f_i(c)(k_i - x_i)/k_i + x_i/k_i|C_i|$. Если, кроме того, $P(S)$ и $P'(S)$ суть произведения соответственно $f_i(c_j^i)$ и $f_i'(c_j^i, x_i)$, взятые по всем возможным i и j и представляющие собой вероятности соответственно того, что в качестве контейнера будет выбран объект S, и того, что в результате встраивания информации будет получено стего S, то стойкость стегосистемы оценивается относительной энтропией (расстоянием Кулльбака — Ляйблера) как $D(P||P') = \sum\limits_S P(S)\log_2(P(S)/P'(S))$, а именно: чем меньше $D(P||P')$, тем выше стойкость стегосистемы.

Для защиты от копирования и несанкционированного использования медиаконтента широко применяется одна из разновидностей цифровых водяных знаков — идентификационные номера (ИН): в контейнер с медиаконтентом, предназначенным конкретному пользователю, внедряется персональный ИН, по которому можно определить имя этого пользователя и привлечь его к ответственности в случае, если копии контейнера он будет распространять среди других (нелегальных) пользователей. Противодействовать этой защите можно атакой сговором: несколько легальных пользователей путём сравнения своих контейнеров обнаруживают в них ИН, создают контейнер с тем же медиаконтентом, но с другим, ложным, ИН, отличным от ИН в их контейнерах, и распространяют его среди нелегальных пользователей, возможно, подставляя тем самым (идентифицируя) какого-то другого легального пользователя. Противостоять этой атаке можно, используя ИН из так называемого допустимого множества булевых векто-

ров некоторой длины n; в нём разные подмножества векторов покрываются разными минимальными интервалами булева пространства размерности n, и его мощность k не превосходит n. В докладе Т. М. Соловьёва и Р. И. Черняка продолжены исследования свойств допустимых множеств ИН, найден метод построения всех таких множеств, изучены возможности идентификации участников сговора по их ложному ИН и возможности создания ими ложного ИН, не идентифицирующего никого. Основным аппаратом в этих исследованиях стало представление допустимого множества мощности k булевой матрицей размера $k \times n$ со строками в качестве элементов множества. Перестановка строк и (или) столбцов, удаление повторяющихся столбцов и инверсия столбцов не меняют свойства допустимости представляемого матрицей множества, и матрицы, получаемые одна из другой с помощью этих операций, называются эквивалентными. Так же называются и представляемые ими множества. В каждом классе их эквивалентности есть матрица с единичной подматрицей E_k на первом месте. Допустимое множество называется сильно допустимым, если удаление из его матрицы любого одного столбца влечёт потерю свойства допустимости множества. Матрица такого множества эквивалентна E_k. Для матрицы любого допустимого множества мощности k существует эквивалентная матрица вида $[E_k||A]$, являющаяся конкатенацией E_k и матрицы $A = A_{k\times(n-k)}$ недопустимого множества. Тем самым показано, что все допустимые множества с точностью до эквивалентности строятся конкатенацией E_k с любыми матрицами $A_{k\times(n-k)}$. Показано, что три или более участников сговора всегда могут создать ложный ИН, никого не идентифицирующий. В нём каждая компонента является мажоритарной функцией от компонент соответствующего столбца матрицы допустимого множества ИН. Защита от этой атаки пока не найдена.

В математической проблематике стеганографических исследований важное место занимает задача выявления факта наличия вкраплений в случайных последовательностях. Известно, например, что гарантированно обнаружить факт наличия независимых вкраплений в последовательность, полученную по полиномиальной схеме с известными вероятностями исходов, возможно только в том случае, когда объём вкраплений растёт по порядку быстрее корня от длины исходной последовательности. Аналогичное утверждение справедливо и для последовательности, образующей простую цепь Маркова с известной матрицей переходных вероятностей. В докладе А. М. Шойтова этот результат обобщён на случай простой цепи Маркова с неизвестной матрицей переходных вероятностей.

5. Математические основы компьютерной безопасности

В проблематике этого направления важнейшее место по-прежнему занимают разработка и исследование математических моделей безопасности КС. Поиску tg-путей и островов в графе доступов модели *Take-Grant* посвящён доклад Д. М. Бречки. Пути ищутся алгоритмом Дейкстры в графе, полученном из графа доступов исключением рёбер без прав $Take$ и $Grant$, а острова — с помощью алгоритма Флойда в графе доступов без рёбер, не содержащих прав $Take$ и $Grant$, и без рёбер, вершины которых не являются субъектами. Сложности этих алгоритмов на графах с n вершинами оцениваются как $O(n^2)$ и $O(n^3)$ соответственно, а сложность процедуры исключения из графа «лишних» рёбер — как $O(n^2)$.

На основе ФАС, ФПАС и ФС ДП-моделей разработан проект математической модели электронных почтовых систем — ЭПС ДП-модели (К. А. Грищенко). В ней, кроме всего прочего, отражены клиент-серверная архитектура системы с субъектами-операционными системами, защищённые сущности, не являющиеся субъектами, и дове-

ренные субъекты-задачи, обладающие правами доступа и реализующие доступ к защищённым сущностям и кодирование в них данных. На компьютере-сервере субъекту-операционной системе подчинён в иерархии доверенный субъект-сервер, которому подчинены в иерархии доверенные субъекты-задачи. Субъекту-задаче соответствуют процессы (или их потоки) в операционной системе (ОС), реализующие механизмы доступа клиентов к серверу, маршрутизации почты, репликации баз данных и др. Субъекты, не реализующие доступ к защищённым сущностям, не обладают правами доступа и не могут получать доступ к этим сущностям, но они могут обладать правами доступа или получать доступ к сущностям-образам защищённых сущностей. Недоверенный субъект-задача может создать доверенного субъекта в случае, когда недоверенный субъект реализовал к себе информационные потоки по памяти от всех сущностей, параметрически ассоциированных с некоторым потенциальным доверенным субъектом. ЭПС ДП-модель предназначается для анализа возможности получения недоверенными субъектами ЭПС доступа к защищённым сущностям и реализации в ЭПС от данных сущностей запрещённых информационных потоков.

Установлены необходимые и достаточные условия передачи прав доступа и реализации информационных потоков в базовой ролевой ДП-модели компьютерной системы в случае, когда на траекториях функционирования системы субъект-сессии не получают доступа владения друг к другу с использованием информационных потоков по памяти к функционально ассоциированным с субъект-сессиями сущностям (П. Н. Девянин).

В КС ОС GNU/Linux и СУБД MySQL обнаружены примеры информационных потоков по времени, которые не подпадают под описания в существующих ДП-моделях (М. А. Качанов). Так, в ОС GNU/Linux возможна передача информации от одного процесса другому через количество нитей, которыми оперирует первый процесс, фиксируемое операционной системой в файле, доступном любому процессу. Аналогично, один пользователь БД может передать информацию другому пользователю через количество своих запросов к БД, фиксируемое ядром СУБД в счётчике запросов, доступном любому пользователю. В связи с обнаружением в КС информационных потоков по времени нового типа возникает необходимость отражения их в моделях безопасности компьютерных систем. Это можно сделать, например, развив далее подходящим образом ДП-модели, а именно, введя в них ассоциированные сущности нового вида, указывающие на возможность реализации к ним информационных потоков по времени в зависимости от выполняемых субъектом действий, и соответственно новые правила преобразования состояний, а также сформулировав и обосновав новые необходимые и достаточные условия возможности реализации в КС информационных потоков по времени между сущностями системы.

Структура двухсеместрового курса по дисциплине «Основы построения защищённых вычислительных сетей» предложена в докладе Д. Н. Колегова. Теоретическая часть курса базируется на руководстве Cisco Safe и архитектурах сетевой безопасности Cisco, а практическая (лабораторные работы) — на среде эмуляции Cisco Packet Tracer.

В научной проблематике безопасности компьютерных сетей весьма актуальной представляется задача отслеживания аномальной активности на участке сети (в канале связи, сегменте сети или локальной машине) посредством статистического анализа трафика на этом участке. В докладе О. В. Ниссенбаум и А. С. Присяжнюка предложен адаптивный метод решения этой задачи. В его основу положены следующие соображения. Трафик компьютерной сети достаточно хорошо приближается дважды

стохастическим потоком событий, в частности альтернирующим потоком. Последний характеризуется тройкой параметров (λ, a_1, a_2), которые можно оценивать в реальном времени и представлять точкой в трёхмерном пространстве. Следя за этой точкой, можно сделать вывод о типичности или нетипичности трафика на данном участке сети в любой промежуток времени. В этой части работы используются методы фильтрации и кластерного анализа.

В трёхуровневой компьютерной системе, состоящей из клиента, внешнего сервера и внутреннего сервера и построенной по модели доверенной подсистемы, клиент для взаимодействия с внутренним сервером должен пройти аутентификацию от своего имени перед внешним сервером, после чего внешний сервер должен пройти аутентификацию перед внутренним от имени группы пользователей, к которой принадлежит данный клиент. Возникает задача разработки такой схемы двухуровневой аутентификации, при которой внешний сервер для взаимодействия с внутренним смог бы использовать учётную запись только той группы, к которой принадлежит клиент, и только тогда, когда клиент взаимодействует с ним. Например, если клиент относится к группе «гости», то внешний сервер может пройти аутентификацию перед внутренним только от имени учётной записи «гость» и только при помощи клиента из группы «гости». Парольная схема аутентификации не решает эту задачу: после сеанса связи клиента со скомпрометированным внешним сервером злоумышленник получает возможность обратиться с внешнего сервера к внутреннему от имени учётной записи группы этого клиента. В докладе П. А. Паутова предложено решение этой задачи с помощью схемы аутентификации на основе произвольных коммутативного алгоритма шифрования E и хэш-функции H. В ней каждой учётной записи внутреннего сервера ставится в соответствие некоторое число S, каждому клиенту внешнего сервера — ключ K. На внешнем сервере хранится $E_K(S)$, где S соответствует учётной записи внутреннего сервера для группы клиента — владельца K. При аутентификации клиента выполняется следующий протокол: клиент посылает внешнему серверу своё имя; внешний сервер внутреннему — имя учётной записи группы клиента; внутренний сервер внешнему — случайный ключ K' алгоритма E; внешний сервер клиенту — величину $E'_K(E_K(S))$; клиент внешнему серверу — значение $h = H(D_K(E'_K(E_K(S))))$; внешний внутреннему — значение h; внутренний сервер сравнивает h с $H(E'_K(S))$, и в случае сравнения клиент проходит аутентификацию перед внутренним сервером от имени учётной записи своей группы. В асимметричном варианте алгоритма E каждому клиенту ставится в соответствие пара ключей: открытый K_e, закрытый K_d, на внешнем сервере для каждого клиента хранятся $E_{K_e}(S)$ и K_e.

Внедрение кода в процесс в операционной системе расширяет возможности для исследования, в том числе для обнаружения уязвимостей в ОС, и мотивирует разработку адекватных методов защиты ОС от угроз, сопутствующих этому действию. Методы внедрения кода в процесс в ОС Windows хорошо изучены. В докладе И. В. Смита два таких метода перенесены на ОС GNU/Linux. Условиями для их применения являются наличие прав доступа пользователя ОС, в процесс которого внедряется код, и возможность исполнять системный вызов `ptrace` этим пользователем.

В мире компьютерной безопасности популярны международные соревнования Capture the Flag (CTF), в которых участники — студенты и профессионалы в этой области — соревнуются в умении успешно защищать свои компьютерные сети и атаковать сети соперников. Участвующие в них студенты, аспиранты и молодые специалисты получают богатый опыт практической работы по защите компьютерных систем и огромные моральные стимулы к занятию научными исследованиями в области ком-

пьютерной безопасности и криптографии. Существующие правила этих соревнований время от времени совершенствуются с целью достижения большего эффекта от участия в них. Последнее из усовершенствований их предложено командой SiBears Томского государственного университета. Реализация новых правил требует разработки и нового сервера для управления соревнованиями по ним. В докладе Н. О. Ткаченко и Д. В. Чернова сообщается о разработанном и реализованном ими сервере соревнований CTF по новым правилам. Архитектура сервера построена по шаблону Модель — Представление — Контроллер (Model — View — Controller). Модель предоставляет данные (обычно для Представления) и реагирует на запросы (обычно от Контроллера); Представление отвечает за отображение информации (выступает как пользовательский интерфейс); Контроллер интерпретирует данные пользователя и информирует Модель и Представление о необходимости соответствующей реакции. Модель реализована в виде реляционной базы данных, работа с которой ведётся методом Object Rational Mapping. Для представления пользователю информации о состоянии сервера и для приёма его запросов используется клиент-серверная архитектура: жюри и команды с помощью веб-браузера отправляют запросы веб-серверу, который передаёт их обработчикам, возвращающим результат в формате XTML.

В докладе М. И. Цоя приведены результаты качественного анализа автоматизированного средства Scyther, предназначенного для моделирования криптографических протоколов с целью обнаружения в них уязвимостей со стороны нарушителя. Протокол в нём представляется множеством состояний и правилами перехода между состояниями. Состояния, достижимые из начального, проверяются на удовлетворение условиям безопасности. В отсутствие среди них состояния, в котором эти условия нарушаются, протокол считается безопасным. С помощью этого средства промоделирован протокол взаимной аутентификации сторон SCID3, предположительно уязвимый атакой «человек посередине». Однако результаты моделирования говорят скорее за то, что это не так.

6. Математические основы надежности вычислительных и управляющих систем

Надёжность программных и аппаратных средств вычислительных и управляющих систем рассматривается как один из показателей их безопасности. Традиционно это направление на школе-семинаре пользуется заслуженным вниманием и интересом. На этот раз оно представлено работами по синтезу отказоустойчивых и самодиагностируемых систем.

В качестве одной из моделей отказоустойчивой системы часто фигурирует рёберное или вершинное расширение определённой кратности k графа системы G, представляющее собой граф со свойством: удаление из него любых k рёбер или вершин соответственно, вызванное отказами в системе, приводит к графу H, в который вкладывается граф G. В случае $H \simeq G$ расширение называется точным. В докладе М. Б. Абросимова описаны все минимальные рёберные k-расширения всех направленных звёзд при любом $k \geqslant 1$, а доклад М. Б. Абросимова и Д. Д. Комарова посвящён построению и описанию минимальных вершинных 1-расширений сверхстройных деревьев. Каждое такое дерево является объединением некоторого числа t цепей с общей концевой вершиной. Показано, в частности, что число дополнительных рёбер минимального вершинного 1-расширения сверхстройного дерева не меньше $t + 1$ и что при $t > 3$ сверхстройное дерево, являющееся объединением цепей длины не больше 2, среди которых есть цепь длины 1 и цепь длины 2, имеет в точности 2 неизоморфных вершинных 1-расширения.

На данный момент известны турниры, являющиеся точными вершинными расширениями диграфов. В докладе А. А. Долгова построено семейство турниров, которые являются точными вершинными 1- и 2-расширениями турниров же.

В связи с проблемой синтеза отказоустойчивых систем возникает задача порождения графов с соответствующими свойствами. В докладе В. А. Мелентьева предложен аналитический подход к её решению применительно к регулярным графам заданного порядка и заданной степени. Подход основан на представлении графа его проекциями, описываемыми в аналитической форме и содержащими всю информацию о структуре и количественных характеристиках графа. Процесс синтеза состоит в построении базовой проекции остовного дерева с последующим доопределением неизвестных рёбер в соответствии с требуемыми структурой и значениями характеристик синтезируемого графа.

В докладе Ю. К. Димитриева и А. Ф. Задорожного рассмотрена задача самодиагностирования модулярных вычислительных систем в присутствии кратных неисправностей с использованием ненадёжных тестов. Зависимость эффективности самодиагностирования от свойств последних изучена методом имитационно-статистического моделирования. Проведено сравнение эффективности самодиагностирования с использованием ненадёжных тестов, соответствующих известной PMC-модели, и ненадёжных тестов, предложенных авторами, и найдены условия, при которых авторские тесты обеспечивают более высокую эффективность.

7. Математические основы информатики и программирования

Развитие математических основ информатики и программирования рассматривается как одно из необходимых условий успешного решения научных и практических проблем компьютерной безопасности и криптографии.

В сегодняшней программной инженерии выделены пять сложностных классов алгоритмов, определяемых в терминах O-большое и o-малое, — субполиномиальные, полиномиальные, субэкспоненциальные, экспоненциальные и гиперэкспоненциальные алгоритмы. Распознавание класса конкретного алгоритма непосредственно по определению часто сопряжено с трудностями вычислительного характера. В докладе В. В. Быковой в качестве меры вычислительной сложности алгоритма взята эластичность его функции $t(n)$ временной сложности, представляющая собой коэффициент пропорциональности между темпами роста величин $t(n)$ и n, и перечисленные классы функций охарактеризованы в терминах этой меры. Свойства эластичности позволяют без особого труда находить эластичность для широкого спектра алгоритмов и распознавать их сложностной класс. В их числе и многие теоретико-числовые алгоритмы, применяемые в криптографии. Предложена также схема сравнения алгоритмов по асимптотическому поведению их эластичности.

Проблема тестирования программного обеспечения (ПО) становится всё более актуальной, в том числе и в связи с компьютерной безопасностью, что неизменно подтверждается растущим интересом к ней со стороны участников школы-семинара Sibecrypt.

Важными свойствами любой программы являются её надёжность и безопасность. Первая означает способность программы работать без причинения вреда окружению, вторая — её устойчивость к внешнему воздействию, которое может нарушить работу программы. В докладе В. В. Горелова сообщено о системе обнаружения (в процессе отладки) действий или бездействий программы при её работе с ресурсами, нежелательных с точки зрения надёжности и безопасности и классифицируемых как ошиб-

ки, или дефекты программы. В их числе: утечки ресурсов, использование ресурсов после их освобождения, повторные освобождения ресурсов, использование неинициализированных ресурсов без их предварительного захвата, использование ресурсов за их границами (относится к динамической памяти и адресному пространству), нарушение вызываемого механизма захвата и освобождения ресурса (функция захвата возвращает идентификатор ранее захваченного и ещё не освобождённого ресурса) и др. Система включает в себя язык описания ресурсов и функций для работы с ними, программу-преобразователь, которая по описанию ресурсов создаёт код для перехвата функций, работающих с интересующими ресурсами, и программу-анализатор, которая считывает соответствующие события, возникающие в процессе работы отлаживаемой программы, и выводит обнаруженные опасные участки программы для анализа человеком. Система реализована для прикладных программ на языке Си под ОС POSIX и WINDOWS.

В тестировании функциональности ПО на первый план выходит задача разработки такого средства порождения тестов, которое обеспечивало бы создание некорректных данных для тестируемого (целевого) ПО, максимально возможную независимость генератора тестов от целевого ПО, минимальное время построения тестов для нового целевого ПО, достаточно высокий объём покрытия кода целевого ПО. В докладе А. Н. Макарова исследована возможность создания генератора тестов с использованием скриптового языка описания структур исходных данных и процедур их формирования. Выяснена недостаточная выразительность этого языка для описания тестов со сложноструктурированными данными. Предложено расширение его средствами описания дополнительной декларативной информации о внутренних связях, ограничениях и зависимостях в данных, позволяющими автоматически генерировать исходные данные с нарушениями этих связей, ограничений и т. п.

В аналитической теории кс-языков известно, что существуют аффинные кс-языки, коммутативные образы которых являются диагоналями коммутативных образов линейных языков с одним дополнительным символом. В докладе К. В. Сафонова и Д. А. Калугина-Балашова сформулированы новые условия, при которых система уравнений Хомского — Щютценберже определяет кс-язык с тем же свойством.

В настоящее время в анализе криптографических систем значительную роль играют методы решения систем уравнений над конечным полем, в частности над полем GF(2) — булевых уравнений. Для их применения криптоаналитик должен иметь перед собой систему уравнений анализируемого криптоалгоритма, представленную в определённой форме — дизъюнктивной, полиномиальной и т. п. Автоматизированная система представления алгоритмов дискретной математики в виде систем булевых уравнений — Transalg доложена в сообщении И. В. Отпущенникова и А. А. Семёнова. Она включает в себя С-подобный язык описания алгоритмов, транслятор с него в язык булевых уравнений и средства приведения последних к нормальным формам разного вида.

Один из методов защиты КС обработки информации заключается в интеграции её с модулем политики безопасности (ПБ). Возможность такой интеграции без изменения кода КС обеспечивается средствами аспектно-ориентированного программирования (АОП). Для их представления используются языки АОП, один из которых — AspectTalk — разработан специально для целей интеграции КС и ПБ. Для доказательства полноты его выразительных средств в докладе Д. А. Стефанцова и А. Е. Крюковой формально доказана семантическая эквивалентность ядра языка AspectTalk и языка объектно-ориентированного программирования Smalltalk. Сделано

это путём доказательства коммутативности диаграммы гомоморфизмов между множествами синтаксических областей и доменов этих языков. Тем самым доказана возможность автоматической трансляции программ с языка AspectTalk в язык Smalltalk и обратно.

8. Вычислительные методы в дискретной математике

Это направление исследований по-прежнему стимулируется потребностями криптоанализа и синтеза стойких криптоалгоритмов.

В докладе Д. В. Беспалова, В. Г. Булавинцева и А. А. Семёнова исследованы возможности графических ускорителей (GPU) в криптоанализе шифров DES и A5/1 атакой грубой силы и показано вполне ожидаемое — их бесперспективность в этой роли.

В соответствии с теорией К. Шеннона сочетание линейных и нелинейных преобразований в операциях шифрования способствует стойкости шифров ко многим атакам. Для эффективной их реализации в блочных шифрах часто длину n блока представляют как $n = m{\cdot}n' = k{\cdot}m'{\cdot}n'$, а в качестве линейного преобразования блока $x = x_1x_2\ldots x_m$ с $|x_i| = n'$, $i = 1,\ldots,m$, берут композицию $A(P(x_1)P(x_2)\ldots P(x_m))$, где P — перестановка в подблоках длины n', $A = \operatorname{diag}(A_1, A_2, \ldots, A_k)$, $A_j \in \mathrm{GL}(m'\cdot n', 2)$, $j = 1,\ldots,k$. В докладе А. А. Дмуха показано, что наилучшие с точки зрения скорости шифрования и количества тактов процессора, необходимых для обработки одного байта информации, значения параметров $m' \in \{2, 4, 8, 16\}$ и $k' \in \{2, 3, \ldots, 8\}$, полученные экспериментально при $n' = 8$ и сопоставимом по криптографическим характеристикам числе итераций, следующие: $m' = k = 2$, $m' = k = 4$, $m' = k = 8$ при длине блока $n = 32, 128, 512$ соответственно. С этими значениями скорость шифрования в 2,5 – 1,3 раза выше, чем с другими значениями.

Решение задачи криптоанализа нередко состоит в определении значения ключа методом опробования возможных значений до тех пор, пока не будет получено истинное значение. Значения ключа часто неравновероятны, и тогда можно осуществить их направленный перебор, начав с наиболее вероятных. Трудоёмкость такого перебора определяется как математическое ожидание длины перебора $m = \sum\limits_{j=1}^{H} q_j \cdot j$, где H — количество различных значений ключа, q_j — вероятность его j-го значения и $q_1 \geqslant q_2 \geqslant \ldots \geqslant q_H$. При слишком большом H вычисление по этой формуле практически неосуществимо. В докладе И. В. Панкратова и О. А. Теплоуховой предложен следующий метод перебора значений ключа с параметром $n \geqslant 1$: ключевое пространство разбивается на два равномощных подмножества мощности $h = H/2$, сначала опробуются первые n элементов первого подмножества, затем по очереди — оставшиеся элементы первого подмножества и $(h - n)$ элементов второго и наконец — остаток второго подмножества. Построена рекуррентная формула для математического ожидания $m_0(n)$ длины перебора этим методом, которая служит верхней оценкой для искомого m. Варьируя в ней параметр n, можно достичь наименьшего значения $m_0(n)$. Практические исследования показывают, что это значение отличается от истинного не более чем на 17 %.

В докладе Р. Т. Файзуллина задачи существования в графе гамильтонова цикла и существования изоморфизма двух графов сводятся к поиску глобального минимума некоторых функционалов и предложены алгоритмы минимизации последних.

9. Прикладная теория автоматов, графов и кодов

Свойства минимальных детерминированных конечных автоматов, распознающих префиксный код заданной мощности n, представлены в докладе И. Р. Акишева и М. Э. Дворкина. Показано, что задача синтеза такого автомата равносильна задаче построения кратчайшей аддитивной цепочки чисел, заканчивающейся числом n. Последняя задача хорошо известна в дискретной математике. К её решению сводится задача об оптимальном алгоритме возведения числа в заданную степень, представляющая интерес для современной криптографии с открытым ключом.

Упорядоченное множество, элементами которого служат классы отношения σ взаимной достижимости состояний автомата A, а отношением порядка — отношение обратной достижимости, называется каркасом этого автомата. Доклад В. Н. Салия посвящён изучению свойств каркаса, связанных с такими алгебраическими понятиями для автомата, как «подавтомат», «гомоморфизм», «конгруэнция». Доказано, в частности, что: 1) каждое конечное упорядоченное множество изоморфно каркасу некоторого автомата с двумя входными символами; 2) конечное упорядоченное множество тогда и только тогда изоморфно каркасу автономного автомата, когда у каждого его элемента есть не более одного нижнего соседа; 3) решётки подавтоматов двух автоматов изоморфны, если и только если изоморфны каркасы этих автоматов; 4) для автоматов с изоморфными каркасами вложение одного автомата в другой является изоморфизмом автоматов; 5) каркас фактор-автомата автомата A по некоторой конгруэнции θ тогда и только тогда изоморфен каркасу самого автомата, когда $\theta \subseteq \sigma$.

В связи с применением конечных автоматов в криптографии возникает задача построения автоматов с поведением, изменяемым по параметру, задаваемому извне и играющему роль ключа шифра. Проблемы нет, если в этой роли выступает начальное состояние автомата, и совсем другое дело, когда ключевой информацией является, например, подмножество переходов в автомате. В докладе В. Н. Тренькаева предложено решение этой задачи в виде композиции двух автоматов с общими множествами входных символов и состояний и блока управления, вырабатывающего управляющий символ в зависимости от ключа, общего состояния и общего входного символа автоматов. Управляющий символ со значениями, поставленными во взаимнооднозначное соответствие автоматам, в каждый такт работы вызывает считывание состояния в общую память и выходного символа с того из автоматов, который соответствует значению управляющего символа в этот момент. Допускается, что функции переходов и выходов автоматов также могут зависеть от ключа.

И. А. Бадеха и П. В. Ролдугин доказали справедливость следующих соотношений между степенью Δ и плотностью ρ в любом связном регулярном графе G, в котором каждое ребро лежит не менее чем в двух максимальных кликах и нет двух смежных вершин с одинаковыми шарами с центрами в них радиуса 1: 1) $\rho \leqslant \Delta - 1$; 2) если $\rho = \Delta - 1$, то $\Delta = 4$ и G изоморфен графу B, состоящему из цикла длины 4 и двух несмежных вершин, смежных каждой вершине в цикле; 3) если $\rho = \Delta - 2$, то либо в G есть две вершины степени Δ, либо G получается из B добавлением доминирующей вершины.

В докладе А. В. Пролубникова предложен алгебраический полный инвариант ациклических графов, который получается не путём канонизации графа, как это обычно делается, а представляет собой множество (для графа с n вершинами) из $1+n(n+1)/2$ числовых значений, каждое из которых есть произведение собственных значений из спектра графа и спектров его подграфов.

Задача восстановления графа обходом его «в глубину» рассмотрена в докладе Е. А. Татаринова. Указаны два класса графов, для которых верхняя оценка временной сложности восстановления является линейной функцией, — класс деревьев и класс колец. Указаны три операции над графами, которые не ухудшают верхнюю оценку временной сложности восстановления графа, — добавление висячей вершины, добавление вершины в ребро и соединение двух графов через вершину сочленения.

В связи с потребностями стандарта цифровой сотовой связи CDMA возникает задача построения линейных кодов с векторами значений бент-функций в качестве кодовых слов. Известные такие коды, построенные на основе бент-функций из класса Мак-Фарланда, обладают следующими параметрами: длина кода $m = 2^n$, размерность кода $k = 2^{n/2} + n/2$ и кодовое расстояние $d = 2^{n/2}$. Например, при $n = 6, 8$ эти параметры для них имеют значения (26, 11, 8), (28, 20, 16) соответственно. В докладе А. В. Павлова предложен алгоритм построения максимальных линейных кодов на основе любых бент-функций, доставляющий коды с $m = 2^n$ и $d = 2^{n/2}$, среди которых есть коды с $n = 6$ и $k = 15$, с $n = 8$ и $k = 30$. В основе алгоритма лежит утверждение о том, что две бент-функции от n переменных тогда и только тогда находятся на минимальном расстоянии $2^{n/2}$ друг от друга, когда они отличаются на аффинном подпространстве размерности $n/2$ и обе на нём аффинны.

10. Математические основы интеллектуальных систем

В криптографии и компьютерной безопасности есть много задач, в решении которых возможно, а иногда просто необходимо применение интеллектуальных систем. Это и принятие решения в условиях неполной, искажённой или неформализуемой информации, и распознавание текстов и сетевого трафика, идентификация источника информации и обнаружение в последовательностях скрытых вложений или свойств, понимание естественного языка и многое другое. К сожалению, пока нельзя сказать, что математическая теория интеллектуальных систем имеет ощутимое представительство в этом направлении. На школе-семинаре Sibecrypt'10 она представлена двумя докладами, имеющими дело с идентификацией состояний динамических объектов (С. И. Колесникова и А. А. Белоус) и логическим выводом решений в классификации объектов (А. Е. Янковская и А. И. Гедике). В первом докладе предложены новые алгоритмы формирования характеристических признаков как системы обобщённых эталонов, на основе которых возможны распознавание состояний динамического объекта (ДО), сглаживание временного ряда и идентификация тренда фрагмента временного ряда, соотнесённого с состоянием ДО. Во втором докладе представлен алгоритм распознавания принадлежности объекта образу с использованием смешанных (условных и безусловных) диагностических тестов (СДТ), в котором решение принимается одновременно с построением СДТ, сокращая тем самым вычислительные затраты.

ЛИТЕРАТУРА

1. Тезисы докладов IX Сибирской научной школы-семинара с международным участием «Компьютерная безопасность и криптография» — Sibecrypt'10 (Тюмень, ТюмГУ, 7–10 сентября 2010 г.) // Прикладная дискретная математика. Приложение. 2010. № 3. 120 с.

СВЕДЕНИЯ ОБ АВТОРАХ

АГИБАЛОВ Геннадий Петрович — профессор, доктор технических наук, заведующий кафедрой защиты информации и криптографии Национального исследовательского Томского государственного университета, г. Томск. E-mail: **agibalov@isc.tsu.ru**

АФАНАСЬЕВ Иван Владимирович — магистрант Новосибирского государственного университета, г. Новосибирск. E-mail: **ivafanas@gmail.com**

БАНДМАН Ольга Леонидовна — профессор, доктор технических наук, главный научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск.
E-mail: **bandman@ssd.sscc.ru, bandman@academ.org**

ГАНОПОЛЬСКИЙ Родион Михайлович — кандидат физико-математических наук, заместитель директора Центра информационных технологий Тюменского государственного университета, г. Тюмень. E-mail: **rodion@utmn.ru**

ГОРЕЛОВ Владимир Владимирович — аспирант кафедры защиты информации и криптографии Национального исследовательского Томского государственного университета, г. Томск. E-mail: **skylark@mail.tsu.ru**

МЕДВЕДЕВ Юрий Геннадьевич — кандидат технических наук, научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **medvedev@ssd.sscc.ru**

НУРУТДИНОВА Алсу Рафаиловна — соискатель, Казанский государственный технический университет им. А. Н. Туполева, г. Казань. E-mail: **alsu124@mail.ru**

ПАУТОВ Павел Александрович — аспирант кафедры защиты информации и криптографии Национального исследовательского Томского государственного университета, г. Томск. E-mail: **__Pavel__@mail.ru**

ПОТТОСИН Юрий Васильевич — доцент, кандидат физико-математических наук, ведущий научный сотрудник Объединенного института проблем информатики НАН Беларуси, г. Минск. E-mail: **pott@newman.bas-net.by**

СТРУЧЕНКОВ Валерий Иванович — доктор технических наук, профессор Московского государственного института радиотехники, электроники и автоматики, г. Москва. E-mail: **str1942@mail.ru**

ТАРКОВ Михаил Сергеевич — кандидат технических наук, старший научный сотрудник Института физики полупроводников им. А. В. Ржанова СО РАН, г. Новосибирск. E-mail: **tarkov@isp.nsc.ru**

ЧЕБОТАРЕВ Анатолий Николаевич — ведущий научный сотрудник Института кибернетики НАН Украины, г. Киев. E-mail: **ancheb@gmail.com**

ШАЛАГИН Сергей Викторович — доцент, кандидат технических наук, доцент Казанского государственного технического университета им. А. Н. Туполева, г. Казань. E-mail: **sshalagin@mail.ru**

ШЕСТАКОВ Евгений Анатольевич — кандидат технических наук, старший научный сотрудник Объединенного института проблем информатики НАН Беларуси, г. Минск. E-mail: **she@newman.bas-net.by**

АННОТАЦИИ СТАТЕЙ НА АНГЛИЙСКОМ ЯЗЫКЕ

Ganopolsky R. M. **THE NUMBER OF DISORDERED COVERS OF A FINITE SET BY SUBSETS HAVING FIXED CARDINALITIES.** This article describes a new type of combinatorial numbers which calculate amount of the covers of a finite set by subsets having fixed cardinalities — parameters of numbers. A series of relations and identities are proved for them. Some sums of these numbers are computed. Special cases of new combinatorial numbers with parameters satisfying certain relations are investigated. Several other applications of these numbers in discrete mathematics are shown.
Keywords: *cover, finite set, combinatoric numbers.*

Pautov P. A. **AUTHENTICATION IN MULTI-TIER SYSTEMS USING PROXY SIGNATURES.** Two authentication protocols for multi-tier system based on proxy signatures are provided. Implementation of these protocols using certificates is considered. In the first protocol, proxy signature is only used for the authentication of a client, but in the second one, it is also used for the authentication of requests inside the system. This makes the second protocol more secure.
Keywords: *multi-tier systems, authentication, proxy signature.*

Gorelov V. V. **A SECURE COMPUTER ARCHITECTURE MODEL AND ITS EMULATION.** A model for secure computer architecture is proposed. The security is provided with protecting values of types and references throughout the machine instructions, operating system, and applications. The protection is achived by means of checking up ranges and operation permissibility for types and references. The implementation of this architecture is made through the development of virtual machine running over the existing operating system.
Keywords: *software, computer architecture, security, safety, virtualization, isolation.*

Tarkov M. S. **ON MAPPING GRAPHS OF PARALLEL PROGRAMS ONTO GRAPHS OF DISTRIBUTED COMPUTER SYSTEMS BY RECURRENT NEURAL NETWORKS.** A problem of mapping graphs of parallel programs onto graphs of distributed computer systems by recurrent neural networks is formulated. The network parameters providing the absence of incorrect solutions are experimentally determined. By introduction of a penalty coefficient into the Lyapunov function for the program graph edges non-coincided with the edges of the computer system, the optimal solutions are computed for mapping the "line" program graph onto a two-dimensional torus. To increase the optimal solution probability a method of the mapping decomposition is proposed. The method essence is a reduction of the solution matrix to a block-diagonal shape. For exclusion of incorrect solutions in mapping the line onto three-dimensional torus, a recurrent Wang network is used because it is converged more rapidly than the Hopfield network.
Keywords: *mapping, graphs of parallel programs, distributed computer systems, Hopfield network, recurrent Wang network.*

Nurutdinova A. R., Shalagin S. V. **MULTI-PARAMETRIC CLASSIFICATION OF AUTOMATON MARKOV MODELS BASED ON THE SEQUENCES THEY GENERATE.** This article is devoted to multi-parametric classification of automaton

Markov models (AMMs) on the base of output sequences with the use of discriminant analysis. The AMMs under consideration are specified by means of stochastic matrices belonging to subclasses defined a priori. A set of claasification features is introduced to distinguish AMMs specified by matrices from different subclasses. The features are related to the frequency characteristics of sequences generated by AMMs. A method is suggested for determining the minimal length of the sequence need to calculate the features with a required accuracy

Keywords: *Markov chain, ergodic stochastic matrix, identification, automaton Markov model, discriminant analysis, linear discriminant functions.*

Pottosin Yu. V., *Shestakov E. A.* **SERIES PARALLEL DECOMPOSITION OF A SYSTEM OF INCOMPLETELY SPECIFIED BOOLEAN FUNCTIONS.** The decomposition problem for a system of incompletely specified Boolean functions is considered. The concept of function dependence on some arguments is introduced for appreciating the complexity of decomposition components. A method for series parallel decomposition of a system of incompletely specified Boolean functions is suggested. The peculiarity of the method is that the arguments of components are not given and are found in the process of decomposition.

Keywords: *system of incompletely specified Boolean functions, decomposition.*

Chebotarev A. N. **REGULAR FORM OF DETERMINISTIC FSMS SPECIFICATIONS IN THE LANGUAGE L.** Some forms for representation of deterministic FSMs specification in the language L are investigated. The use of such forms in constructing specifications decreases the possibility of errors.

Keywords: *specification language, finite automaton, cyclic automaton.*

Struchenkov V. I. **NEW ALGORITHMS FOR OPTIMAL RESOURSE ALLOCATION.** New algorithms for the optimal allocation of resources are suggested using Pareto sets and branch-and-bound method.

Keywords: *optimal allocation of resources, Pareto set, branch-and-bound method.*

Afanasyev I. V. **RESEARCH OF EVOLUTION OF CELLULAR AUTOMATA MODELING "PHASE SEPARATION" PROCESS ON TRIANGULAR MESH.** Cellular automata (CA) on square and triangular meshes modeling phase separation processes and called phase separation CA are considered. It is shown how, for a given phase separation CA on square mesh, to constract a phase separation CA on triangular mesh. Some classes of phase separation CA are introduced according to the features of their behavior.

Keywords: *cellular automaton, self-organization, phases separation, triangular mesh.*

Bandman O. L. **A METHOD FOR CONSTRUCTION OF CELLULAR AUTOMATA SIMULATING PATTERN FORMATION PROCESSES.** A method for constructing Cellular Automata (CA) which simulate self-organizing process of stable patterns formation is proposed. The method is based on parallel composition of two CA. Two basic cases are considered: 1) when one CA operates independently having a controlling influence on the other CA behavior; 2) when both CA interact at each iteration step. The method is illustrated by the computer simulation results for two selforgzanizing systems: pattern formation on heated surface and achieving balance between prey and predator.

Keywords: *mathematical modelling, cellular automata, dissipative structures, selforganization.*

Medvedev Yu. G. **SIMULATING A PISTON MOTION BY A GAS-LATTICE MODEL.** In the paper, ways for defining the boundary conditions of the multiparticle lattice-gas FHP-MP model are discussed. Particular attention is paid to the unsteady boundary conditions. A method for moving walls in the model is suggested. It allows using obstacles which are walking while the simulation is run. Computer experiments with a piston movement in the cylinder were carried out. They showed that the simulated process corresponds qualitatively to the physical laws.
Keywords: *cellular automaton, gas flow, unsteady boundary conditions, piston motion.*

Agibalov G. P. **SIBECRYPT'10 REVIEW.** This is a survey of papers presented at 9th Siberian Workshop SIBECRYPT devoted to mathematical problems in computer security and cryptography and held in Toumen, Russia, in September 7–10, 2010.
Keywords: *applied discrete mathematics, computer security, cryptography, steganography.*

Журнал «Прикладная дискретная математика» включен в перечень ВАК рецензируемых российских журналов, в которых должны быть опубликованы основные результаты диссертаций, представляемых на соискание учёной степени кандидата и доктора наук, а также в перечень журналов, рекомендованных УМО в области информационной безопасности РФ в качестве учебной литературы по специальности «Компьютерная безопасность».

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте vestnik.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также и правила подготовки рукописей статей в журнал.

Тематика публикаций журнала:

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надежности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Дискретные модели реальных процессов*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и ее приложениям*