МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2014 №2(24)

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

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.

Редактор *Н. И. Шидловская*
Верстка *И. А. Панкратовой*

Подписано к печати 26.05.2014.
Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 12,52. Уч.-изд. л. 14. Тираж 300 экз.

Издательство ТГУ. 634029, Томск, ул. Никитина, 4

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ДИСКРЕТНЫХ АВТОМАТОВ

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

DOI 10.17223/20710410/24/1

УДК 519.6

КОМБИНАТОРНЫЕ СВОЙСТВА СИСТЕМ РАЗНОРАЗМЕРНЫХ 0,1-МАТРИЦ

Я. Э. Авезова*, В. М. Фомичев*,**

*Национальный исследовательский ядерный университет «МИФИ», г. Москва, Россия

**Финансовый университет при Правительстве Российской Федерации, г. Москва, Россия

E-mail: avezovayana@gmail.com, fomichev@nm.ru

Исследованы комбинаторные свойства мультипликативной частичной полугруппы, порождённой системой разноразмерных неотрицательных матриц. Понятие примитивности распространено с систем квадратных неотрицательных матриц на системы разноразмерных матриц. Даны оценки экспонента системы разноразмерных неотрицательных матриц.

Ключевые слова: *система разноразмерных матриц, частичная полугруппа, примитивная система матриц, экспонент.*

Введение

Исследование экспонентов квадратных неотрицательных матриц относится к одному из классических направлений исследований в дискретной математике и криптологии. Важность этого направления в криптологии связана с применением матрично-графового подхода к исследованию перемешивающих свойств композиций функций и, в конечном счете, с получением оценок стойкости криптосистем относительно методов последовательного опробования. Второе важное направление приложений — это исследование функций, распространяющих искажения.

В [1] понятие экспонента распространено с матрицы (графа) на систему квадратных неотрицательных матриц (систему графов). Ранее для систем квадратных матриц рассматривался также множественный экспонент [2]. Обзор известных результатов по экспонентам матриц (графов) и систем матриц дан в [3].

В данной работе понятие экспонента распространяется на новый класс алгебраических структур — множество систем неотрицательных разноразмерных (не квадратных) матриц. Таким образом, объект исследования здесь — мультипликативная частичная полугруппа матриц. Прикладное значение расширения объекта исследования определяется использованием в криптографических схемах композиций функций $X^n \to X^m$, где $n \neq m$ и, как правило, $X = \{0, 1\}$. Например, в DES-алгоритме раундовая подстановка построена с использованием отображения расширения, реализующего функцию $X^{32} \to X^{48}$, и системы s-боксов, реализующей функцию $X^{48} \to X^{32}$.

Полученные результаты могут быть использованы как для анализа, так и для построения криптографических систем.

1. Определяющие свойства систем разноразмерных матриц

Пусть $\hat{M} = \{M_1, \ldots, M_p\}$ — система разноразмерных 0,1-матриц, то есть матриц над множеством $\{0, 1\}$, где матрица M_i имеет размеры $m_i \times k_i$ и в общем случае $m_i \neq k_i$, $i = 1, \ldots, p$. Порядком системы разноразмерных матриц (СРМ), обозначаемым $|\hat{M}|$, назовём число p составляющих её матриц.

Рассмотрим СРМ $\hat{M}$ как алфавит, в котором можно строить слова. Длиной слова называется число символов алфавита, составляющих слово.

Пару матриц (M_i, M_j) назовем *разрешённой* (*допустимой*) *биграммой* (то есть разрешённым словом длины 2) в алфавите $\hat{M}$, если $k_i = m_j$, $i, j \in \{1, \ldots, p\}$. Для разрешённой пары (M_i, M_j) определено произведение матриц $M_i M_j$, в противном случае произведение матриц не определено. Слово $M_{i_1} \ldots M_{i_l}$ *разрешённое* (*допустимое*) тогда и только тогда, когда любая биграмма слова является разрешённой. Разрешённому слову $M_{i_1} \ldots M_{i_l}$ длины l в алфавите $\hat{M}$ соответствует матрица $\varphi(M_{i_1} \ldots M_{i_l})$ размера $m_{i_1} \times k_{i_l}$, являющаяся произведением матриц $M_{i_1} \ldots M_{i_l}$.

Обозначим: $\hat{M}^*$ — множество всех слов в алфавите $\hat{M}$; $D(\hat{M}^*)$ — множество всех разрешённых слов в алфавите $\hat{M}$; $\langle \hat{M} \rangle$ — частичная полугруппа разноразмерных матриц (по умножению), порождённая системой $\hat{M} = \{M_1, \ldots, M_p\}$. Частичная полугруппа $\langle \hat{M} \rangle$ состоит из матриц, соответствующих всем разрешённым словам в алфавите $\hat{M}$, то есть $\langle \hat{M} \rangle = \varphi(D(\hat{M}^*))$. При умножении двух матриц в полугруппе $\langle \hat{M} \rangle$ сначала выполняется умножение этих матриц над множеством целых неотрицательных чисел, после чего все положительные элементы заменяются единицами.

На множестве $D(\hat{M}^*)$ определена частичная операция конкатенации слов. Конкатенация слов w_1 и w_2 (записывается как $w_1 w_2$) является разрешённой, если последний символ слова w_1 и первый символ слова w_2 образуют разрешённую пару в алфавите $\hat{M}$. Конкатенация пустого (то есть не содержащего символов) слова с любым словом является разрешённой. Заметим, что $D(\hat{M}^*)$ есть полугруппа относительно операции конкатенации и φ есть гомоморфизм $D(\hat{M}^*) \to \langle \hat{M} \rangle$.

Свойства СРМ удобно описывать с использованием композиционных графов.

Определение 1. *Композиционным графом системы матриц* $\hat{M}$ (обозначается $\Gamma(\hat{M})$) назовём p-вершинный ориентированный граф, в котором вершина i биективно соответствует матрице M_i и пара (i, j) есть дуга графа $\Gamma(\hat{M})$ тогда и только тогда, когда (M_i, M_j) есть разрешённая биграмма в алфавите $\hat{M}$, то есть $k_i = m_j$, $i, j \in \{1, \ldots, p\}$.

В частности, системе из p квадратных матриц порядка n соответствует полный p-вершинный ориентированный граф.

Далее считаем, что полугруппа $\langle \hat{M} \rangle$ не пуста и граф $\Gamma(\hat{M})$ не содержит изолированных вершин.

2. Связность СРМ

Классифицируем СРМ по свойству связности композиционных графов.

Определение 2. Если композиционный граф $\Gamma(\hat{M})$ имеет одну (более одной) компоненту связности, то соответствующую СРМ $\hat{M}$ назовём *связной* (*несвязной*).

Подсистему $\hat{M}'$ несвязной СРМ $\hat{M}$ назовем *компонентой связности системы* $\hat{M}$, если граф $\Gamma(\hat{M}')$ — максимальный связный подграф графа $\Gamma(\hat{M})$ (то есть связный подграф, не являющийся подграфом другого связного подграфа графа $\Gamma(\hat{M})$).

Определение 3. СРМ $\hat{M}$ назовём *сильносвязной*, если её композиционный граф $\Gamma(\hat{M})$ сильносвязный.

Пример 1. Проиллюстрируем связность на примере СРМ порядка 5.

Пусть $\hat{M}^{(1)} = \{M_1^{(1)}, \ldots, M_5^{(1)}\}$, где $M_1^{(1)}, \ldots, M_5^{(1)}$ имеют размеры 8×9, 9×10, 10×8, 11×12 и 12×11 соответственно. Частичная полугруппа $\langle\hat{M}^{(1)}\rangle$ не является связной. На рис. 1,*а* приведён композиционный граф $\Gamma(\hat{M}^{(1)})$.

Пусть $\hat{M}^{(2)} = \{M_1^{(2)}, \ldots, M_5^{(2)}\}$, где $M_1^{(2)}, \ldots, M_5^{(2)}$ имеют размеры 8×9, 9×10, 10×11, 11×12 и 12×9 соответственно. Частичная полугруппа $\langle\hat{M}^{(2)}\rangle$ является связной, но не сильносвязной. На рис. 1,*б* приведён композиционный граф $\Gamma(\hat{M}^{(2)})$.

Пусть $\hat{M}^{(3)} = \{M_1^{(3)}, \ldots, M_5^{(3)}\}$, где $M_1^{(3)}, \ldots, M_5^{(3)}$ имеют размеры 7×8, 8×9, 10×8, 9×10 и 10×7 соответственно. Частичная полугруппа $\langle\hat{M}^{(3)}\rangle$ является сильносвязной. На рис. 1,*в* приведён композиционный граф $\Gamma(\hat{M}^{(3)})$.

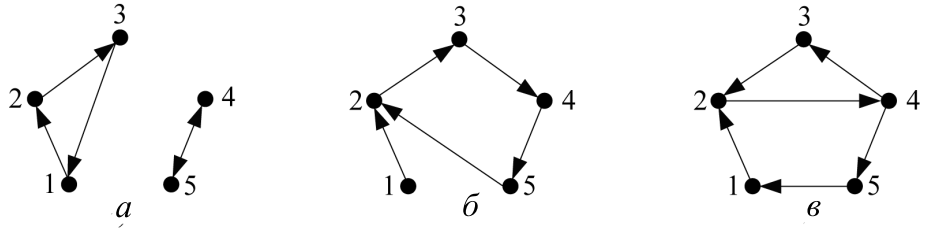

Рис. 1. Композиционные графы несвязной (*а*), связной (*б*) и сильносвязной (*в*) СРМ

3. Регулярность СРМ

Классифицируем СРМ по свойствам слов полугруппы $D(\hat{M}^*)$.

Если конкатенация wv слов w и v разрешена, то слово v называется продолжением слова w. Продолжение слова w называется нетривиальным, если v — непустое слово.

Например, если wvu — разрешённая конкатенация трёх слов, то v и vu — продолжения слова w, u — продолжение слов v и wv.

Определение 4. Пусть $w \in D(\hat{M}^*)$. Слово w называется:

— *нерегулярным*, если длина любого его продолжения ограничена, то есть не превышает некоторой фиксированной числовой границы τ;
— *регулярным*, если для любого натурального числа τ найдётся продолжение слова w, имеющее длину τ;
— *сильно регулярным*, если любое его продолжение допускает дальнейшее нетривиальное продолжение.

Из определения 4 следует:

— множества регулярных и нерегулярных слов не пересекаются;
— сильно регулярные слова являются регулярными.

Определение 5. СРМ $\hat{M}$ называется:

— *нерегулярной*, если полугруппа $D(\hat{M}^*)$ не содержит регулярных слов;
— *регулярной*, если полугруппа $D(\hat{M}^*)$ содержит хотя бы одно регулярное слово;
— *сильно регулярной*, если $D(\hat{M}^*)$ состоит только из сильно регулярных слов.

Теорема 1.

а) СРМ $\hat{M}$ регулярная тогда и только тогда, когда граф $\Gamma(\hat{M})$ циклический. В нерегулярной СРМ длина слов полугруппы $\langle\hat{M}\rangle$ не превышает p.

б) СРМ $\hat{M}$ сильно регулярная тогда и только тогда, когда граф $\Gamma(\hat{M})$ не содержит вершин с нулевой полустепенью исхода.

в) Сильносвязная СРМ является сильно регулярной.

Доказательство.

а) В силу определения регулярной СРМ $\hat{M}$, в полугруппе $D(\hat{M}^*)$ имеется регулярное слово w. Тогда w допускает продолжение сколь угодно большой длины. Так как алфавит $\hat{M}$ конечный, то слово w содержит повторение символов алфавита, то есть граф $\Gamma(\hat{M})$ имеет цикл.

Обратно, если граф $\Gamma(\hat{M})$ имеет цикл, то соответствующее слово, состоящее из k-кратного повторения цикла, принадлежит полугруппе $D(\hat{M}^*)$, $k = 1, 2, \ldots$, то есть полугруппа $D(\hat{M}^*)$ содержит регулярное слово.

Отсюда следует также, что нерегулярное слово не содержит повторяющихся символов, следовательно, длина его ограничена порядком p алфавита $\hat{M}$.

б) По определению сильно регулярной СРМ $\hat{M}$ имеем, что любое слово из $D(\hat{M}^*)$ сильно регулярное, то есть допускает дальнейшее продолжение. Это равносильно тому, что полустепень исхода любой вершины графа $\Gamma(\hat{M})$ отлична от нулевой.

в) Сильносвязный граф $\Gamma(\hat{M})$ не содержит вершин с нулевой полустепенью исхода. Отсюда по теореме 1, б СРМ $\hat{M}$ — сильно регулярная. ■

Пример 2. Рассмотрим композиционные графы сильно регулярных, регулярных и нерегулярных СРМ на примере СРМ порядка 5.

Пусть $\hat{M}^{(4)} = \{M_1^{(4)}, \ldots, M_5^{(4)}\}$, где $M_1^{(4)}, \ldots, M_5^{(4)}$ имеют размеры 2×5, 5×5, 5×5, 6×6 и 3×6 соответственно. Данная СРМ является сильно регулярной, её композиционный граф представлен на рис. 2,*а*.

Пусть $\hat{M}^{(5)} = \{M_1^{(5)}, \ldots, M_5^{(5)}\}$, где $M_1^{(5)}, \ldots, M_5^{(5)}$ имеют размеры 2×5, 5×5, 5×5, 6×7 и 3×6 соответственно. Данная СРМ является регулярной (например, M_1M_2 — регулярное слово), но не сильно регулярной (полустепень исхода вершины 4 равна 0), её композиционный граф представлен на рис. 2,*б*.

Пусть $\hat{M}^{(6)} = \{M_1^{(6)}, \ldots, M_5^{(6)}\}$, где $M_1^{(6)}, \ldots, M_5^{(6)}$ имеют размеры 2×5, 5×6, 6×7, 7×8 и 8×3 соответственно. Данная СРМ является нерегулярной, её композиционный граф представлен на рис. 2,*в*.

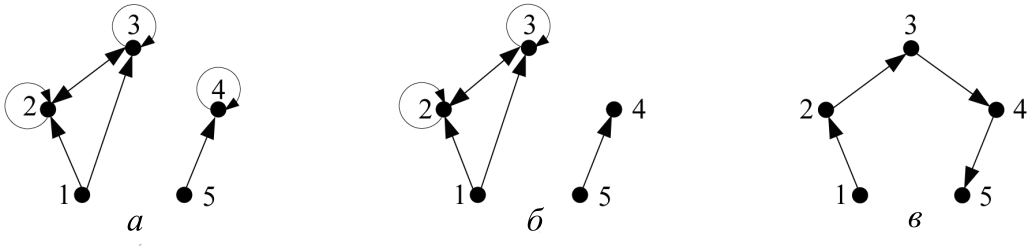

Рис. 2. Композиционные графы сильно регулярной (*а*), регулярной (*б*) и нерегулярной (*в*) СРМ

4. Примитивность СРМ

Данное в [1] определение примитивности системы квадратных неотрицательных матриц порядка n (n-вершинных графов) распространено в [2] на СРМ. Приведём это определение в равносильной формулировке, удобной для дальнейшего изложения.

Система разноразмерных матриц $\hat{M}$ называется примитивной, если в полугруппе $D(\hat{M}^*)$ найдётся слово w, такое, что $\varphi(w) > 0$, то есть все элементы матрицы $\varphi(w)$ положительны. *Экспонентом системы матриц* $\hat{M}$ (обозначается $\exp \hat{M}$) называется наименьшая из длин слов w, таких, что $\varphi(w) > 0$. Если такого слова w не существует,

то полагаем $\varphi(w) = \infty$. Отметим, что такое определение экспонента является корректным и для множества несвязных СРМ.

Пусть X — частично упорядоченное или квазиупорядоченное множество, L — линейно упорядоченное множество. Функция $f : X \to L$ называется изотонной (антиизотонной) [4] (по другой терминологии — монотонной (антимонотонной)), если для любых $x, x' \in X$ из отношения $x \leqslant x'$ следует, что $f(x) \leqslant f(x')$ $(f(x) \geqslant f(x'))$.

Обозначим $M_0(n \times m)$ множество всех неотрицательных матриц размера $n \times m$. Пусть $A, A' \in M_0(n \times m)$, где $A = (a_{ij})$, $A' = (a'_{ij})$. Положим $A \geqslant A'$ тогда и только тогда, когда $a_{ij} \geqslant a'_{ij}$ для всех $i = 1, \ldots, n$, $j = 1, \ldots, m$. Если при этом существуют такие i и j, что $a_{ij} > a'_{ij}$, то запишем $A > A'$. Бинарное отношение $\geqslant$ обладает свойствами рефлексивности, антисимметричности и транзитивности и, следовательно, является отношением частичного порядка.

Обозначим $[M_0]$ множество всех СРМ. Пусть $\hat{M}, \hat{U} \in [M_0]$, где $\hat{M} = \{M_1, \ldots, M_p\}$, $\hat{U} = \{U_1, \ldots, U_s\}$ — связные СРМ. Положим $\hat{U} \leqslant \hat{M}$, если при любом $i = 1, \ldots, s$ найдётся $j \in \{1, \ldots, p\}$, такое, что $U_i \leqslant M_j$. Данное бинарное отношение рефлексивно и транзитивно, но не антисимметрично, следовательно, является отношением квазипорядка.

Рассмотрим экспонент как функцию $[M_0] \to \mathbb{N}$.

Утверждение 1. Экспонент является антиизотонной функцией $[M_0] \to \mathbb{N}$.

Доказательство. Пусть $\hat{U} \leqslant \hat{M}$ и $A_1, \ldots, A_t \in \hat{U}$. Тогда произведение $A_1 \cdot \ldots \times \times A_t \in \langle \hat{U} \rangle$. Так как $\hat{U} \leqslant \hat{M}$, то в СРМ $\hat{M}$ найдутся матрицы $B_1, \ldots, B_t$, такие, что $A_i \leqslant B_i$, $i = 1, \ldots, t$. Тогда $A_1 \cdot \ldots \cdot A_t \leqslant B_1 \cdot \ldots \cdot B_t$. Следовательно, если $A_1 \cdot \ldots \cdot A_t > 0$, то и $B_1 \cdot \ldots \cdot B_t > 0$. Отсюда $\exp \hat{M} \leqslant \exp \hat{U}$. ■

Следствие 1. Экспонент является антиизотонной функцией $M_0(n \times n) \to \mathbb{N}$.

Доказательство. Пусть $\hat{U}$ и $\hat{M}$ — системы порядка 1, то есть $\hat{U} = \{A\}$, $\hat{M} = \{B\}$, где $A, B \in M_0(n \times n)$. Тогда отношение $\hat{U} \leqslant \hat{M}$ равносильно отношению $A \leqslant B$ и $\exp \hat{M} = \exp B$, $\exp \hat{U} = \exp A$. Тогда $\exp B \leqslant \exp A$ по утверждению 1. ■

Следствие 2. Если $\hat{U} \subseteq \hat{M}$, то $\exp \hat{M} \leqslant \exp \hat{U}$.

Доказательство. Если $\hat{U} \subseteq \hat{M}$, то $\hat{U} \leqslant \hat{M}$. Тогда $\exp \hat{M} \leqslant \exp \hat{U}$ по утверждению 1. ■

Замечание 1. Из утверждения 1 и его следствий получаем:

— если СРМ $\hat{M}$ не примитивна, то любая её подсистема $\hat{U}$ тоже не примитивна;

— если подсистема $\hat{U}$ системы $\hat{M}$ примитивная, то СРМ $\hat{M}$ тоже примитивная; в частности, СРМ, содержащая квадратную примитивную матрицу, примитивна.

Утверждение 2. Если $\tilde{M}_1, \ldots, \tilde{M}_r$ суть все примитивные компоненты связности несвязной СРМ $\hat{M}$, то $\exp \hat{M} = \min\{\exp \tilde{M}_1, \ldots, \exp \tilde{M}_r\}$.

Доказательство. Так как СРМ $\hat{M}$ несвязная, то алфавит $\hat{M}$ разбивается на непустые блоки $\tilde{M}_1, \ldots, \tilde{M}_r$, соответствующие компонентам связности графа $\Gamma(\hat{M})$. Это разбиение индуцирует разбиение полугруппы $D(\hat{M}^*)$ на подполугруппы $D(\tilde{M}_1^*), \ldots, D(\tilde{M}_r^*)$. Следовательно, если наименьшая длина слова w_i из $D(\tilde{M}_i^*)$, такого, что $\varphi(w_i) > 0$, равна $\exp \tilde{M}_i$, $i = 1, \ldots, r$, то кратчайшее слово w из $D(\hat{M}^*)$, такое, что $\varphi(w) > 0$, совпадает с одним из слов $w_1, \ldots, w_r$, и длина слова w равна $\min\{\exp \tilde{M}_1, \ldots, \exp \tilde{M}_r\}$. ■

Разрешённое слово $w = M_{i_1} \ldots M_{i_l}$ длины l в алфавите $\hat{M}$ назовём *правильным*, если $m_{i_1} = k_{i_l}$. Если слово w правильное, то $\varphi(w)$ — квадратная матрица порядка m_{i_1}

и определён $\exp\varphi(w)$. Обозначим через $R(\hat{M}^*)$ множество всех правильных слов в алфавите $\hat{M}$.

Утверждение 3.

а) Разрешённое слово $w = M_{i_1} \dots M_{i_l}$ является правильным тогда и только тогда, когда $(i_1, \dots, i_l)$ есть цикл в графе $\Gamma(\hat{M})$.

б) Пусть $R(\hat{M}^*) = \{w_1, w_2, \dots\}$, тогда $\exp\hat{M} \leqslant \min\{\exp w_1, \exp w_2, \dots\}$.

Доказательство.

а) Путь $(i_1, \dots, i_l)$ в графе $\Gamma(\hat{M})$ является циклом тогда и только тогда, когда в $\Gamma(\hat{M})$ имеется дуга (i_l, i_1), то есть $m_{i_1} = k_{i_l}$. Это равносильно тому, что слово w правильное.

б) Любая степень матрицы $\varphi(w_i)$ принадлежит полугруппе $\langle\hat{M}\rangle$, $i = 1, 2, \dots$ Если матрица $\varphi(w_i)$ примитивная, то существует $\exp\varphi(w_i)$, равный наименьшему натуральному $t(i)$, такому, что $(\varphi(w_i))^{t(i)} > 0$. Следовательно, $\exp\hat{M} \leqslant t(i) = \exp w_i$, $i = 1, 2, \dots$ Отсюда получаем утверждение. ■

5. Эквивалентность СРМ

СРМ $\hat{M}$ и $\hat{U}$ называются *эквивалентными* (обозначим $\hat{M} \approx \hat{U}$), если $\exp\hat{M} = \exp\hat{U}$.

Утверждение 4. $\hat{M} \approx \hat{U}$, если $\hat{U} \leqslant \hat{M}$ и $\hat{M} \leqslant \hat{U}$.

Доказательство. Если $\hat{U} \leqslant \hat{M}$, то $\exp\hat{M} \leqslant \exp\hat{U}$ по утверждению 1. Если $\hat{M} \leqslant \hat{U}$, то по утверждению 1 $\exp\hat{U} \leqslant \exp\hat{M}$. Следовательно, $\exp\hat{U} = \exp\hat{M}$. ■

Пусть матрица $A \in \hat{M}$. Матрица A называется *максимальной матрицей системы* $\hat{M}$, если из отношения $A \leqslant A'$ следует $A = A'$.

СРМ $\hat{M}$ называется *сокращённой*, если она состоит только из максимальных матриц. Заметим, что любая СРМ может быть приведена к сокращённой СРМ удалением всех немаксимальных матриц.

Утверждение 5. Любая СРМ $\hat{M}$ эквивалентна своей сокращенной подсистеме.

Доказательство. Рассмотрим СРМ $\hat{U} = \hat{M} \setminus \hat{V}$, где $\hat{V}$ — подмножество всех немаксимальных матриц системы $\hat{M}$. По построению $\hat{U} \subseteq \hat{M}$, значит, $\exp\hat{M} \leqslant \exp\hat{U}$ по следствию 2 утверждения 1.

С другой стороны, $\hat{M} \leqslant \hat{U}$ в силу того, что СРМ $\hat{U}$ состоит из всех максимальных матриц системы $\hat{M}$. Следовательно, по утверждению 1 $\exp\hat{U} \leqslant \exp\hat{M}$, то есть выполнено обратное неравенство. Отсюда $\hat{M} \approx \hat{U}$. ■

6. Универсальная нижняя оценка экспонента сильносвязных СРМ

СРМ $\hat{M} = \{M_1, \dots, M_p\}$ поставим в соответствие (обозначим это соответствие ψ) систему квадратных матриц $\hat{M}' = \{M_1', \dots, M_p'\}$ порядка n, где $n = \max\{m_1, \dots, m_p\}$. Матрица M_i' получена из матрицы M_i размера $m_i \times k_i$ дополнением снизу $n - m_i$ нулевыми строками размера k_i и справа $n - k_i$ нулевыми столбцами размера n, $i = 1, \dots, p$.

Теорема 2. Если СРМ $\hat{M}$ сильносвязная, то:

а) $\max\{k_1, \dots, k_p\} = n$;

б) ψ — гомоморфизм со свойством $\psi(\langle\hat{M}\rangle) = \langle\hat{M}'\rangle$;

в) если матрица $A = M_1' + \dots + M_p'$ примитивная, то $\exp\hat{M} \geqslant \exp A$.

Доказательство.

а) По определению числа n в СРМ $\hat{M}$ имеется матрица размера $n \times k$, где $k \in \{k_1, \ldots, k_p\}$. Пусть для определённости это матрица M_1. В сильносвязном графе $\Gamma(\hat{M})$ имеется дуга $(j, 1)$, где $j \in \{2, \ldots, p\}$. Тогда матрица M_j имеет размеры $m_j \times n$, следовательно, $\max\{k_1, \ldots, k_p\} \geqslant n$.

Если $\max\{k_1, \ldots, k_p\} = n' > n$, то в СРМ $\hat{M}$ имеется матрица размера $m \times n'$, где $m \in \{m_1, \ldots, m_p\}$. Пусть для определённости это матрица M_p. В сильносвязном графе $\Gamma(\hat{M})$ имеется дуга (p, j), где $j \in \{1, \ldots, p-1\}$. Тогда матрица M_j имеет размеры $n' \times k_j$, отсюда $\max\{m_1, \ldots, m_p\} = n' > n$, то есть имеем противоречие с определением числа n. Следовательно, $\max\{k_1, \ldots, k_p\} = n$.

б) Из правил построения матриц $M'_1, \ldots, M'_p$ и умножения матриц следует, что для разрешённой биграммы $(M_i M_j)$ выполнено $\psi(M_i M_j) = \psi(M_i)\psi(M_j)$, то есть ψ — гомоморфизм полугрупп $\langle \hat{M} \rangle \to \langle \hat{M}' \rangle$.

в) При любом натуральном t матрица A^t есть сумма матриц, соответствующих всем словам длины t в алфавите $\hat{M}'$, а именно

$$A^t = \sum_{w \in (\hat{M})^t} \psi(\varphi(w)).$$

Удалив из суммы все матрицы, соответствующие неразрешённым словам в алфавите $\hat{M}$, выполним суммирование по словам длины t из полугруппы $\langle \hat{M} \rangle$. Тогда

$$A^t \geqslant \sum_{w \in (\hat{M})^t \cap \langle \hat{M} \rangle} \psi(\varphi(w)).$$

Если хотя бы одному разрешённому слову длины t соответствует положительная матрица, то $A^t > 0$. Вместе с тем матрица A^t может быть положительной, когда ни одна из слагаемых матриц не положительна. Следовательно, $\exp \hat{M} \geqslant \exp A$. ■

ЛИТЕРАТУРА

1. *Фомичев В. М.* Методы дискретной математики в криптологии. М.: Диалог-МИФИ, 2010. 424 с.
2. *Сачков В. Н., Тараканов В. Е.* Комбинаторика неотрицательных матриц. М.: ТВП, 2000. 448 с.
3. *Когос К. Г., Фомичев В. М.* Положительные свойства неотрицательных матриц // Прикладная дискретная математика. 2012. № 4(18). С. 116–121.
4. *Биркгоф Г.* Теория решёток. М.: Наука, 1984. 567 с.

2014 Теоретические основы прикладной дискретной математики №2(24)

DOI 10.17223/20710410/24/2

УДК 519.716

ОБ ОДНОМ КОНТИНУАЛЬНОМ СЕМЕЙСТВЕ β-ЗАМКНУТЫХ КЛАССОВ ФУНКЦИЙ МНОГОЗНАЧНОЙ ЛОГИКИ[1]

Д. К. Подолько

Московский государственный университет им. М. В. Ломоносова, г. Москва, Россия

E-mail: podolko_dk@mail.ru

Приводится пример континуального семейства β-замкнутых классов функций многозначной логики, содержащих только функции, принимающие не более трёх значений, где оператор β-замыкания определён на основе кодирования функций многозначной логики в двоичной системе счисления. Доказываются некоторые свойства данного семейства классов.

Ключевые слова: *функции многозначной логики, суперпозиция, замкнутые классы, β-замыкание.*

Введение

Широко известно, что семейство классов функций k-значной логики, замкнутых относительно операции суперпозиции, является континуальным при $k \geqslant 3$ [1, 2]. В связи с этим возникают значительные сложности при описании данных классов, и поэтому для их изучения часто рассматривают различные усиления операции суперпозиции, которые позволяют получить семейства замкнутых классов с более обозримой структурой [3 – 5].

В работе [6] использован аналогичный подход. В ней определен оператор β-замыкания на основе кодирования функций многозначной логики в двоичной системе счисления и построено отображение семейства β-замкнутых классов функций k-значной логики в семейство замкнутых классов булевых функций. При этом установлено, что в каждый класс булевых функций отображается конечное (и непустое) множество β-замкнутых классов функций k-значной логики, содержащих только функции, принимающие не более двух значений. Таким образом, доказана счётность семейства различных β-замкнутых классов, содержащих только функции, принимающие не более двух значений, в то время как аналогичное семейство классов, замкнутых относительно операции суперпозиции, является континуальным [1, 2].

В настоящей работе приводится континуальное семейство β-замкнутых классов функций k-значной логики, содержащих только функции, принимающие не более трёх значений, а также доказываются некоторые свойства данного семейства классов.

1. Основные определения и обозначения

Сформулируем основные определения (подробнее см. [6]). Пусть $k \geqslant 2$. Через E_k обозначим множество $\{0, 1, \ldots, k-1\}$, а через E_k^n, где $n \geqslant 1$, — множество всех наборов длины n, компоненты которых принадлежат E_k. Через P_k обозначим множество всех функций k-значной логики.

[1]Работа выполнена при финансовой поддержке РФФИ (проекты №11-01-00508 и 14-01-00598) и программы фундаментальных исследований ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения» (проект «Задачи оптимального синтеза управляющих систем»).

Будем рассматривать только случаи, когда значность k логики задаётся соотношением $k = 2^m$, где $m \geqslant 2$. Тогда каждое число α из E_k можно записать в двоичной системе счисления. Это означает, что ему взаимно однозначно сопоставляется двоичный вектор $(\alpha_1, \ldots, \alpha_m)$ из E_2^m, который будем обозначать через $\langle \alpha_1, \ldots, \alpha_m \rangle$ или $\widehat{\alpha}$. Переменной x, принимающей значения из E_k, можно поставить в соответствие вектор-переменную $\langle x_1, \ldots, x_m \rangle$, где $x_1, \ldots, x_m$ принимают значения из множества E_2, таким образом, что каждому значению α переменной x ставится в соответствие значение $\langle \alpha_1, \ldots, \alpha_m \rangle$ вектор-переменной $\langle x_1, \ldots, x_m \rangle$. Данную вектор-переменную будем обозначать также через $\widehat{x}$, а каждую из переменных $x_1, \ldots, x_m$ называть *компонентой вектор-переменной* $\widehat{x}$ или просто *компонентой переменной* x.

Произвольной n-местной функции $F(x^1, \ldots, x^n)$ из P_k можно сопоставить вектор-функцию $\langle f_1, \ldots, f_m \rangle$, где функции $f_1, \ldots, f_m$ являются булевыми и каждая из них зависит от всех булевых переменных $x_1^j, \ldots, x_m^j$, $j = 1, \ldots, n$ (здесь $\widehat{x}^j = \langle x_1^j, \ldots, x_m^j \rangle$). Данную вектор-функцию будем также обозначать через $\widehat{F}(\langle x_1^1, \ldots, x_m^1 \rangle, \ldots, \langle x_1^n, \ldots, x_m^n \rangle)$ или $\widehat{F}$.

Описанные представления будем называть *двоичным представлением числа* α, *переменной* x и *функции* F соответственно.

Пусть $F \in P_k$ и $\widehat{F} = \langle f_1, \ldots, f_m \rangle$. Каждую из булевых функций $f_1, \ldots, f_m$ будем называть *компонентой вектор-функции* $\widehat{F}$, а также *компонентой функции* F. Множество всех компонент функции F обозначим через $b(F)$.

Пусть $\mathcal{A} \subseteq P_k$. Класс булевых функций, совпадающий с замыканием множества $\bigcup\limits_{F \in \mathcal{A}} b(F)$ относительно операций суперпозиции и введения несущественной переменной, будем называть *булевым замыканием множества* $\mathcal{A}$ и обозначать через $B(\mathcal{A})$.

Будем говорить, что функция H из P_k получена из функций множества $\mathcal{A}$ при помощи *операции двоичной суперпозиции*, если найдутся функция F из множества $\mathcal{A}$ и функции $g_1, \ldots, g_{mn}$ из множества $B(\mathcal{A})$ (где n — число переменных функции F), такие, что выполняется следующее равенство:

$$\widehat{H} = \widehat{F}\big(\langle g_1, \ldots, g_m \rangle, \ldots, \langle g_{m(n-1)+1}, \ldots, g_{mn} \rangle\big).$$

Множество всех функций, которые могут быть получены из функции системы $\mathcal{A}$ при помощи операций двоичной суперпозиции и введения несущественной переменной, будем называть *β-замыканием множества* $\mathcal{A}$ и обозначать через $[\mathcal{A}]_\beta$. В работе [6] установлено, что введённый оператор β-замыкания удовлетворяет всем необходимым свойствам оператора замыкания, алгебраическое описание которых можно найти, например, в [7].

Пусть $F \in P_k$. Через $D(F)$ будем обозначать множество значений функции F, а через $P_{k|r}$, где $1 \leqslant r \leqslant k$, — множество всех функций k-значной логики, принимающих не более r значений.

2. Континуальное семейство β-замкнутых классов

Приведём пример континуального семейства β-замкнутых классов функций из $P_{k|3}$. Для каждого $n \geqslant 2$ определим функцию F_n из P_k, зависящую от переменных x^1, …, x^n, y^1, …, y^n, через её двоичное представление $\langle f_{n,1}, \ldots, f_{n,m} \rangle$:

$$f_{n,1} = x_1^1 \vee x_1^2 \vee \cdots \vee x_1^n \vee y_1^1 \& y_1^2 \& \ldots \& y_1^n;$$
$$f_{n,i} = y_1^1 \vee y_1^2 \vee \cdots \vee y_1^n \vee d_n(x_1^1, x_1^2, \ldots, x_1^n) \vee \overline{x_1^1} \& \overline{x_1^2} \& \ldots \& \overline{x_1^n}, \quad 2 \leqslant i \leqslant m,$$

где $x \vee y$, $x\&y$, $\overline{x}$ — соответственно дизъюнкция, конъюнкция и отрицание в двузначной логике, а $d_n(x_1^1, \ldots, x_1^n) = \bigvee\limits_{1 \leqslant i < j \leqslant n} x_1^i \& x_1^j$.

Так как каждая из компонент функции F_n зависит только от первых компонент переменных $x^1, \ldots, x^n, y^1, \ldots, y^n$, в дальнейшем будем обозначать булевы переменные x_1^i, y_1^i через x_i и y_i соответственно, $i = 1, \ldots, n$. Тогда

$$f_{n,1} = x_1 \vee \cdots \vee x_n \vee y_1 \& \ldots \& y_n;$$
$$f_{n,i} = y_1 \vee \cdots \vee y_n \vee d_n(x_1, \ldots, x_n) \vee \overline{x_1} \& \ldots \& \overline{x_n}, \quad 2 \leqslant i \leqslant m.$$

Для функций F_n, $n \geqslant 2$, выполняются следующие свойства.

1) Компоненты функции F_n существенно зависят только от переменных $x_1, \ldots, x_n$ и $y_1, \ldots, y_n$.
2) $B(\{F_n\}) = O^\infty$, где через O^∞ обозначается замкнутый класс всех булевых функций f, для которых найдётся переменная x, от которой зависит функция f, и такая, что $f \geqslant x$ (подробнее см., например, [8]).
   Данное свойство следует из того, что все компоненты функции F_n принадлежат классу O^∞, а функция $f_{n,2}$ не содержится в классах MO^∞ и O_0^∞ — обоих предполных классах в классе O^∞ [8] — и тем самым порождает все функции из этого класса.
3) $D(F_n) = \{\alpha_{01}, \alpha_{10}, \alpha_{11}\}$, где двоичные представления чисел α_{01}, α_{10}, α_{11} равны $\langle 0, 1, 1, \ldots, 1\rangle$, $\langle 1, 0, 0, \ldots, 0\rangle$ и $\langle 1, 1, 1, \ldots, 1\rangle$ соответственно.
   Так как функции $f_{n,2}, \ldots, f_{n,m}$ равны между собой, то двоичное представление функции F_n может принимать значения только из множества

$$\{\langle 0, 0, 0, \ldots, 0\rangle,\ \langle 0, 1, 1, \ldots, 1\rangle,\ \langle 1, 0, 0, \ldots, 0\rangle,\ \langle 1, 1, 1, \ldots, 1\rangle\}.$$

   Функция $f_{n,1}$ равна нулю тогда и только тогда, когда $x_1 = \cdots = x_n = 0$ и хотя бы одна из переменных $y_1, \ldots, y_n$ равна 0. Но если $x_1 = \cdots = x_n = 0$, то функция $f_{n,2}$ равна единице, поэтому двоичное представление функции F_n не принимает значение $\langle 0, 0, 0 \ldots, 0\rangle$, а значения $\langle 0, 1, 1, \ldots, 1\rangle$, $\langle 1, 0, 0, \ldots, 0\rangle$, $\langle 1, 1, 1, \ldots, 1\rangle$ принимает соответственно при следующих значениях переменных:
   — $x_1 = \cdots = x_n = y_1 = \cdots = y_n = 0$;
   — $x_1 = 1,\ x_2 = \cdots = x_n = y_1 = \cdots = y_n = 0$;
   — $x_1 = \cdots = x_n = y_1 = \cdots = y_n = 1$.

Отметим, что похожее множество функций использовалось в работе [9] для доказательства континуальности семейства самодвойственных классов функций k-значной логики.

Положим $\mathcal{W} = \bigcup\limits_{n=2}^{\infty} \{F_n\}$.

Лемма 1. Для всех $n \geqslant 2$ справедливо соотношение $F_n \notin \big[\mathcal{W} \backslash \{F_n\}\big]_\beta$.

Доказательство. Предположим, что утверждение леммы 1 неверно. Это значит, что имеет место соотношение

$$\widehat{F_n} = \widehat{H}\big(\langle \omega_1, \ldots, \omega_m\rangle, \ldots, \langle \omega_{m(r-1)+1}, \ldots, \omega_{mr}\rangle\big), \tag{1}$$

где функция H принадлежит множеству $\mathcal{W} \backslash \{F_n\}$ либо получена из функций данного множества путём введения несущественных переменных; r — число переменных функции H; булевы функции $\omega_1, \ldots, \omega_{mr}$ принадлежат множеству $B(\mathcal{W} \backslash \{F_n\})$.

Поскольку функция F_n не содержит несущественных переменных, то без ограничения общности можно положить, что $H = F_p$ для некоторого числа p, $p \geqslant 2$, $p \neq n$, и $r = 2p$. Так как $B(\{F_i\}) = O^\infty$ для любого $i \geqslant 2$, то $B(\mathcal{W}\backslash\{F_n\}) = O^\infty$, и поэтому все функции $\omega_1, \ldots, \omega_{2mp}$ принадлежат классу O^∞, а так как все компоненты функции F_n существенно зависят только от переменных $x_1, \ldots, x_n, y_1, \ldots, y_n$, то без ограничения общности можно считать, что булевы функции $\omega_1, \ldots, \omega_{2mp}$ зависят только от тех же переменных.

Поскольку все компоненты функции F_p существенно зависят только от первых компонент вектор-переменных, то компоненты $\omega_2, \ldots, \omega_m, \ldots, \omega_{m(2p-1)+2}, \ldots, \omega_{2mp}$ можно заменить на константу 1, которая содержится в классе O^∞.

Для всех i, $1 \leqslant i \leqslant p$, обозначим функцию $\omega_{m(i-1)+1}$ через g_i, а функцию $\omega_{m(p+i-1)+1}$ — через h_i. В данных обозначениях соотношение (1) можно записать при помощи эквивалентной системы равенств:

$$\begin{cases} f_{n,1} = g_1 \vee \cdots \vee g_p \vee h_1 \& \ldots \& h_p; & (2) \\ f_{n,i} = h_1 \vee \cdots \vee h_p \vee d_p(g_1, \ldots, g_p) \vee \overline{g_1} \& \ldots \& \overline{g_p}, \quad 2 \leqslant i \leqslant m. & (3) \end{cases}$$

Так как $g_i \in O^\infty$, то g_i мажорирует некоторую переменную из множества $\{x_1, \ldots, x_n, y_1, \ldots, y_n\}$, $i = 1, \ldots, p$. Предположим, что для некоторого числа i, $1 \leqslant i \leqslant p$, функция g_i мажорирует переменную y_j, $1 \leqslant j \leqslant n$. Рассмотрим набор $\widetilde{\alpha} = (\alpha_1, \ldots, \alpha_{2n})$ из E_2^{2n}, у которого все компоненты равны нулю, кроме компоненты с номером $n+j$ (она соответствует переменной y_j). Тогда $g_i(\widetilde{\alpha}) = 1$. Но $f_{n,1}(\widetilde{\alpha}) = 0$, а значит, не выполняется соотношение (2). Следовательно, ни одна из функций $g_1, \ldots, g_p$ не мажорирует ни одну из переменных $y_1, \ldots, y_n$. Аналогичными рассуждениями получаем, что ни одна из функций $h_1, \ldots, h_p$ не мажорирует ни одну из переменных $x_1, \ldots, x_n$.

Далее предположим, что выполняется неравенство $p > n$. Так как каждая из функций $g_1, \ldots, g_p$ мажорирует хотя бы одну переменную из множества $\{x_1, \ldots, x_n\}$, найдётся переменная x_i, $1 \leqslant i \leqslant n$, которая мажорируется как минимум двумя функциями g_{i_1} и g_{i_2}, $1 \leqslant i_1 < i_2 \leqslant p$. Рассмотрим набор $\widetilde{\alpha} = (\alpha_1, \ldots, \alpha_{2n})$ из E_2^{2n}, у которого все компоненты равны нулю, кроме компоненты с номером i (она соответствует переменной x_i). Тогда $g_{i_1}(\widetilde{\alpha}) = g_{i_2}(\widetilde{\alpha}) = 1$, а следовательно, $d_p\big(g_1(\widetilde{\alpha}), \ldots, g_p(\widetilde{\alpha})\big) = 1$. Но $f_{n,2}(\widetilde{\alpha}) = 0$, так как $d_n(\alpha_1, \ldots, \alpha_n) = \overline{\alpha_1} \& \ldots \& \overline{\alpha_n} = \alpha_{n+1} \vee \cdots \vee \alpha_{2n} = 0$. Поэтому не выполняется соотношение (3).

Теперь предположим, что $p < n$. Для каждого числа i, $1 \leqslant i \leqslant p$, функция h_i мажорирует хотя бы одну переменную из множества $\{y_1, \ldots, y_n\}$. Это означает, что найдётся число j_i, $1 \leqslant j_i \leqslant n$, для которого $h_i \geqslant y_{j_i}$. Рассмотрим набор $\widetilde{\alpha} = (\alpha_1, \ldots, \alpha_{2n})$ из E_2^{2n}, у которого компоненты с номерами $n + j_1, \ldots, n + j_p$ равны единице (они соответствуют переменным $y_{j_1}, \ldots, y_{j_p}$), а остальные компоненты равны нулю. Тогда $h_1(\widetilde{\alpha}) = \cdots = h_p(\widetilde{\alpha}) = 1$, а следовательно, $h_1(\widetilde{\alpha}) \& \ldots, \& h_p(\widetilde{\alpha}) = 1$. Поскольку $p < n$, $\{i_1, \ldots, i_p\} \neq \{1, \ldots, n\}$. Поэтому найдётся число j, $1 \leqslant j \leqslant n$, такое, что $\alpha_{n+j} = 0$. Значит, $\alpha_1 \vee \cdots \vee \alpha_n = \alpha_{n+1} \& \ldots \& \alpha_{2n} = 0$, и на наборе $\widetilde{\alpha}$ функция $f_{n,1}$ равна нулю. Поэтому не может выполняться соотношение (2), что и требовалось доказать. ■

Теорема 1. Пусть $k = 2^m$, где $m \geqslant 2$. Тогда семейство различных β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием O^∞ является континуальным.

Доказательство. Для каждого подмножества S множества $\{2, 3, 4, \ldots\}$ определим множество функций k-значной логики $\{F_i : i \in S\}$, которое будем обозначать через $\mathcal{W}(S)$.

Рассмотрим два различных подмножества R и Q множества $\{2,3,4,\dots\}$. Для них найдётся число n, такое, что $n \in R\backslash Q$ или $n \in Q\backslash R$. Без ограничения общности будем считать, что $n \in R\backslash Q$. Тогда $F_n \in [\mathcal{W}(R)]_\beta$, но по лемме 1 выполняется соотношение $F_n \notin [\mathcal{W}\backslash\{F_n\}]_\beta$, а значит, $F_n \notin [\mathcal{W}(Q)]_\beta$.

Получили, что для различных подмножеств R и Q множества $\{2,3,4,\dots\}$ различны и β-замкнутые классы функций $[\mathcal{W}(R)]_\beta$ и $[\mathcal{W}(Q)]_\beta$. Так как семейство различных таких подмножеств имеет континуальную мощность, то семейство различных β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием O^∞ как минимум континуально.

Множество всех функций k-значной логики является счётным, а следовательно, семейство его подмножеств континуально. Поэтому семейство различных β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием O^∞ имеет мощность не более континуума. Значит, оно континуально, что и требовалось показать. ■

3. Другие семейства β-замкнутых классов функций из $P_{k|3}$

Покажем теперь, что для каждого класса $\mathcal{B}$ булевых функций, который строго содержится в классе O^∞, семейство β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием $\mathcal{B}$ является конечным. Для этого введём вспомогательные определения.

Пусть $f, f_1, \dots, f_p$ — булевы функции, зависящие от одного и того же множества переменных. Функцию f будем называть *функционально зависимой от функций* $f_1, \dots, f_p$, если существует p-местная булева функция g, такая, что $f = g(f_1, \dots, f_p)$. В ином случае функцию f будем называть *функционально независимой от функций* $f_1, \dots, f_p$. Функции, реализующие константы 0 и 1, будем считать функционально зависимыми от пустого множества функций. Если для всех чисел j, $1 \leqslant j \leqslant p$, функции f_j являются функционально независимыми от функций $f_1, \dots, f_{j-1}, f_{j+1}, \dots, f_p$, то множество функций $\{f_1, \dots, f_p\}$ назовём *функционально независимым.*

Лемма 2. Пусть $A = \{f_1, \dots, f_p\}$ — функционально независимое множество булевых функций и $d(A)$ — число значений булевой вектор-функции $(f_1, \dots, f_p)$. Тогда выполняются соотношения $\lceil \log_2 d(A) \rceil \leqslant p \leqslant d(A) - 1$.

Доказательство. Так как $d(A)$ — число значений булевой вектор-функции $(f_1, \dots, f_p)$, то очевидно, что верно неравенство $d(A) \leqslant 2^p$. Поэтому $\log_2 d(A) \leqslant p$, а так как p является натуральным числом, то $\lceil \log_2 d(A) \rceil \leqslant p$.

Доказательство соотношения $p \leqslant d(A) - 1$ будем вести индукцией по p.

Пусть $p = 1$. Тогда функция f_1 не является константой по определению функционально независимого множества. Поэтому $d(A) = 2$, что и необходимо.

Теперь предположим, что для любого функционально независимого множества $A' = \{f_1, \dots, f_s\}$ булевых функций, где $1 \leqslant s < p$ и $d(A')$ — число значений булевой вектор-функции $(f_1, \dots, f_s)$, верно соотношение $s \leqslant d(A') - 1$.

Покажем, что выполняется соотношение $p \leqslant d(A) - 1$ для функционально независимого множества $\{f_1, \dots, f_p\}$ булевых функций. Для этого рассмотрим его подмножество $A' = \{f_1, \dots, f_{p-1}\}$. Это функционально независимое множество функций, поэтому по предположению индукции верно соотношение $p-1 \leqslant d(A')-1$. Если для любого вектора $(a_1, \dots, a_{p-1})$ из E_2^{p-1} на множестве наборов, на которых вектор-функция $(f_1, \dots, f_{p-1})$ принимает значение $(a_1, \dots, a_{p-1})$, функция f_p является константой, то функция f_p является функционально зависимой от функций $f_1, \dots, f_{p-1}$, что неверно. Поэтому найдётся вектор $(a_1, \dots, a_{p-1})$ из E_2^{p-1}, такой, что на множестве наборов, на которых вектор-функция $(f_1, \dots, f_{p-1})$ принимает значение $(a_1, \dots, a_{p-1})$, функция f_p

не является константой. Значит, $d(A) \geqslant d(A') + 1$, из чего и следует необходимое соотношение. ■

Пусть $F \in P_k$ и $\widehat{F} = \langle f_1, \ldots, f_m \rangle$. Множество компонент $\{f_{i_1}, \ldots, f_{i_p}\}$ будем называть *базой функции* F, если данное множество является функционально независимым, а остальные компоненты функции F являются функционально зависимыми от функций $f_{i_1}, \ldots, f_{i_p}$.

Лемма 3. Пусть $F \in P_k$, $\widehat{F} = \langle f_1, \ldots, f_m \rangle$, $\{f_{i_1}, \ldots, f_{i_p}\}$ — база функции F и $|D(F)| = r$. Тогда выполняются соотношения $\lceil \log_2 r \rceil \leqslant p \leqslant r - 1$.

Доказательство. Так как значение функции F однозначным образом определяется по значениям функций из её базы, то доказываемые соотношения вытекают из леммы 2. ■

Лемма 4. Пусть $\mathcal{A} \subseteq P_{k|3}$, $[\mathcal{A}]_\beta = \mathcal{A}$, $R \subseteq \mathcal{A}$, $B(\mathcal{A}) = B(R)$ и класс $B(\mathcal{A})$ содержится в классе M или в классе T_{01}. Тогда для любой функции F из множества $\mathcal{A}$ найдётся функция H_F из $\mathcal{A}$, зависящая от одной переменной, и такая, что $F \in [R \cup \{H_F\}]_\beta$.

Доказательство. Рассмотрим произвольную функцию F из множества $\mathcal{A}$. Обозначим через n число переменных функции F, а через $\langle f_1, \ldots, f_m \rangle$ — её двоичное представление. Если функция F является константой, то в качестве H_F можно взять функцию, реализующую ту же константу, что и функция F, и зависящую от одной переменной.

Пусть теперь функция F принимает два или три значения. Тогда среди её компонент содержатся не только константы, а значит, класс $B(\mathcal{A})$ содержит селекторные функции [8].

Рассмотрим случай, когда функция F принимает три значения (более простой случай, когда функция F принимает два значения, рассматривается аналогично). По лемме 3 у неё найдётся база, состоящая ровно из двух функций. Без ограничения общности будем считать, что базой является множество $\{f_1, f_2\}$. Функции f_1 и f_2 не являются функционально зависимыми друг от друга. Поэтому они не равны между собой и не являются константами. Если класс $B(\mathcal{A})$ содержится в классе монотонных функций, то функции f_1 и f_2 являются монотонными, а значит, верны равенства $f_1(0, \ldots, 0) = f_2(0, \ldots, 0) = 0$ и $f_1(1, \ldots, 1) = f_2(1, \ldots, 1) = 1$. Если класс $B(\mathcal{A})$ содержится в классе T_{01}, то выполняются аналогичные равенства.

Функции f_1 и f_2 не равны между собой. Значит, найдётся набор $\widetilde{\alpha} = (\alpha_1, \ldots, \alpha_{mn})$ из E_2^{mn}, на котором значения данных функций различны. Без ограничения общности будем считать, что $f_1(\widetilde{\alpha}) = 1$, а $f_2(\widetilde{\alpha}) = 0$. Поскольку функция F принимает ровно три значения, а значения её компонент $f_3, \ldots, f_m$ однозначным образом определяются по значениям функций f_1 и f_2, то двоичное представление функции F принимает следующие значения:

$$\begin{gathered}\big\langle 0, 0, f_3(0, \ldots, 0), \ldots, f_m(0, \ldots, 0) \big\rangle, \\ \big\langle 1, 0, f_3(\alpha_1, \ldots, \alpha_{mn}), \ldots, f_m(\alpha_1, \ldots, \alpha_{mn}) \big\rangle, \\ \big\langle 1, 1, f_3(1, \ldots, 1), \ldots, f_m(1, \ldots, 1) \big\rangle.\end{gathered}$$

Для каждого числа i, $1 \leqslant i \leqslant mn$, определим булеву функцию $g_i(x_1, x_2)$ следующим образом: если $\alpha_i = 1$, то $g_i = x_1$, а если $\alpha_i = 0$, то $g_i = x_2$. Все эти функции являются селекторными, а следовательно, содержатся в классе $B(\mathcal{A})$.

Далее рассмотрим вектор-функцию $\langle f_1, \ldots, f_m \rangle (\langle g_1, \ldots, g_m \rangle, \ldots, \langle g_{m(n-1)+1}, \ldots, g_{mn} \rangle)$. Она является двоичным представлением некоторой функции H_F из P_k. Так как $m \geqslant 2$,

а функции $g_1, \ldots, g_{mn}$ зависят от булевых переменных x_1 и x_2, функция H_F зависит от одной переменной, а поскольку все функции $g_1, \ldots, g_{mn}$ содержатся в множестве $B(\mathcal{A})$ и класс $\mathcal{A}$ является β-замкнутым, функция H_F принадлежит множеству $\mathcal{A}$. Покажем, что она искомая.

Обозначим через $\langle h_1, \ldots, h_m\rangle$ двоичное представление функции H_F. Так как функции $f_3, \ldots, f_m$ являются функционально зависимыми от функций f_1 и f_2, по построению функции $h_3, \ldots, h_m$ являются функционально зависимыми от функций h_1 и h_2, причём эти функциональные зависимости могут быть заданы теми же функциями.

Докажем соотношение $\widehat{F} = \langle h_1, \ldots, h_m\rangle(\langle f_1, \ldots, f_m\rangle)$. Так как функции $f_3, \ldots, f_m$ являются функционально зависимыми от функций f_1 и f_2, а функции $h_3, \ldots, h_m$ — от функций h_1 и h_2, достаточно доказать справедливость равенств $f_1 = h_1(f_1, \ldots, f_m)$ и $f_2 = h_2(f_1, \ldots, f_m)$. А так как компоненты функции H_F существенно зависят только от переменных x_1 и x_2, достаточно показать, что $f_1 = h'_1(f_1, f_2)$ и $f_2 = h'_2(f_1, f_2)$, где функции h'_1, h'_2 получены удалением несущественных переменных $x_3, \ldots, x_m$ из функций h_1 и h_2 соответственно. Для этого рассмотрим произвольный набор $\widetilde{\gamma}$ из E_2^{mn}. Имеют место три случая:

— Если $f_1(\widetilde{\gamma}) = f_2(\widetilde{\gamma}) = 0$, то $h'_j(0,0) = f_j\big(g_1(0,0), \ldots, g_{mn}(0,0)\big) = f_j(0, \ldots, 0) = 0$, $j = 1, 2$, что и требуется.
— Случай, когда $f_1(\widetilde{\gamma}) = f_2(\widetilde{\gamma}) = 1$, рассматривается аналогично предыдущему.
— Если $f_1(\widetilde{\gamma}) = 1$, а $f_2(\widetilde{\gamma}) = 0$, то $h_1(1,0) = f_1\big(g_1(1,0), \ldots, g_{mn}(1,0)\big) = f_1(\widetilde{\alpha}) = 1$, а $h_2(1,0) = f_2(\widetilde{\alpha}) = 0$, что и необходимо.

Таким образом, верно равенство $\widehat{F} = \langle h_1, \ldots, h_m\rangle(\langle f_1, \ldots, f_m\rangle)$, а поскольку каждая из функций $f_1, \ldots, f_m$ содержится в классе $B(R)$, получим $F \in [R \cup \{H_F\}]_\beta$, что и требовалось доказать. ■

Лемма 5. Пусть $F \in P_k$, $F \notin P_{k|1}$. Тогда в β-замкнутом классе $[\{F\}]_\beta$ найдётся функция, зависящая от одной переменной, множество компонент которой содержит селекторную функцию.

Доказательство. Обозначим двоичное представление функции F через $\langle f_1, \ldots, f_m\rangle$, а число её переменных — через n. Если все компоненты функции F являются константами, то функция F принимает одно значение, что неверно по условиям леммы. Поэтому среди компонент функции F найдётся функция, не являющаяся константой. Без ограничения общности положим, что это функция f_1. Тогда класс $[\{f_1\}]$ содержит селекторные функции [8]. Поэтому $x_1 \in [\{f_1\}]$ и верно равенство $x_1 = f_1(g_1, \ldots, g_{mn})$ для некоторых функций $g_1, \ldots, g_{mn}$, которые принадлежат классу $[\{f_1\}]$. Поскольку функция x_1 зависит только от переменной x_1, без ограничения общности можно считать, что функции $g_1, \ldots, g_{mn}$ тоже зависят только от переменной x_1, а так как $f_1 \in B(\{F\})$, в силу замкнутости класса $B(\{F\})$ верно включение $\{g_1, \ldots, g_{mn}\} \subseteq B(\{F\})$.

Обозначим через H функцию, имеющую следующее двоичное представление:

$$\langle f_1, \ldots, f_m\rangle\big(\langle g_1, \ldots, g_m\rangle, \ldots, \langle g_{m(n-1)+1}, \ldots, g_{mn}\rangle\big).$$

По построению функция H содержится в классе $[\{F\}]_\beta$ и зависит от одной переменной. А так как выполняется соотношение $f_1(g_1, \ldots, g_{mn}) = x_1$, имеем $x_1 \in b(H)$, что и требовалось показать. ■

Лемма 6. Пусть $\mathcal{B}$ — замкнутый класс булевых функций, содержащийся в классе M или в классе T_{01}. Тогда существует число $r = r(\mathcal{B})$, такое, что у любого

β-замкнутого класса $\mathcal{A}$ функций из $P_{k|3}$ с булевым замыканием $\mathcal{B}$ существует конечный базис, все функции которого зависят не более чем от r переменных.

Доказательство. Пусть $\mathcal{A}$ — β-замкнутый класс функций из $P_{k|3}$, такой, что $B(\mathcal{A}) = \mathcal{B}$. Если все функции из класса $\mathcal{A}$ являются константами, то утверждение леммы 6 очевидно, а в качестве r можно взять число 1.

Пусть теперь среди функций класса $\mathcal{A}$ имеется хотя бы одна функция, принимающая не менее двух значений. Обозначим её через F. Тогда по лемме 5 в классе $[\{F\}]_\beta$ существует функция H, зависящая от одной переменной, множество компонент которой содержит селекторную функцию. Так как $F \in \mathcal{A}$ и класс $\mathcal{A}$ является β-замкнутым, верно $H \in \mathcal{A}$.

Известно, что у класса $\mathcal{B}$ существует конечный базис [8]. Обозначим его функции через $g_1, \ldots, g_s$, а переменные, от которых они зависят, — через $x_1^1, \ldots, x_1^r$, где r является максимальным числом переменных у данных функций. Для каждого j, $1 \leqslant j \leqslant s$, определим функцию G_j из P_k через её двоичное представление: $\widehat{G_j} = \widehat{H}(\langle g_j, \ldots, g_j \rangle)$. Тогда функция G_j зависит не более чем от r переменных, а так как $\{g_1, \ldots, g_s\} \subseteq B(\mathcal{A})$ и класс $\mathcal{A}$ является β-замкнутым, выполняется $G_j \in \mathcal{A}$.

Поскольку множество компонент функции H содержит селекторную функцию, $g_j \in b(G_j)$, $j = 1, \ldots, s$. Поэтому верно равенство $B(\{G_1, \ldots, G_s\}) = B(\mathcal{A})$, и по лемме 4 для любой функции F из множества $\mathcal{A}$ можно выбрать функцию H_F из $\mathcal{A}$, зависящую от одной переменной и такую, что $F \in [\{G_1, \ldots, G_s, H_F\}]_\beta$. Значит, верно включение $\mathcal{A} \subseteq [\{G_1, \ldots, G_s\} \cup \{H_F | F \in \mathcal{A}\}]_\beta$. Но так как все функции из множеств $\{G_1, \ldots, G_s\}$ и $\{H_F | F \in \mathcal{A}\}$ содержатся в $\mathcal{A}$, верно и обратное включение. Поэтому $\mathcal{A} = [\{G_1, \ldots, G_s\} \cup \{H_F | F \in \mathcal{A}\}]_\beta$.

Так как число функций из $\mathcal{A}$, зависящих от одной переменной, конечно, конечно и множество $\{H_F | F \in \mathcal{A}\}$, а поскольку функции $G_1, \ldots, G_s$ зависят не более чем от r переменных, где r зависит только от $\mathcal{B}$, получаем утверждение леммы 6. ■

Теорема 2. Пусть $k = 2^m$, $m \geqslant 2$, а $\mathcal{B}$ — замкнутый класс булевых функций, содержащийся в классе M или в классе T_{01}. Тогда число различных β-замкнутых классов $\mathcal{A}$ функций из $P_{k|3}$ с булевым замыканием $\mathcal{B}$ является конечным.

Доказательство. Пусть $\mathcal{A}$ — произвольный β-замкнутый класс функций из $P_{k|3}$, такой, что $B(\mathcal{A}) = \mathcal{B}$. Тогда по лемме 6 в классе $\mathcal{A}$ найдётся конечный базис, все функции которого зависят не более чем от r переменных, где r зависит только от класса $\mathcal{B}$. Так как число функций из $P_{k|3}$, зависящих не более чем от r фиксированных переменных, является конечным, конечно и число их подмножеств. Поэтому конечно число рассматриваемых β-замкнутых классов функций. ■

Следствие 1. Пусть $k = 2^m$, где $m \geqslant 2$. Тогда для любого замкнутого класса $\mathcal{B}$ булевых функций, такого, что $\mathcal{B} \subseteq O^\infty$, $\mathcal{B} \neq O^\infty$, семейство β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием $\mathcal{B}$ является конечным.

Доказательство. Если класс $\mathcal{B}$ содержится в O^∞ и $\mathcal{B} \neq O^\infty$, то либо $\mathcal{B} \subset M$, либо $\mathcal{B} \subset T_{01}$, либо выполняются оба соотношения [8]. Поэтому по теореме 2 семейство β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием $\mathcal{B}$ является конечным. ■

Рассмотрим далее множество Q замкнутых классов $\mathcal{B}$ булевых функций, для которых число различных β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием $\mathcal{B}$ не является конечным. По теореме 2 классы, содержащиеся в классе M или в классе T_{01}, не принадлежат множеству Q.

Несложно показать, что для любой функции F из P_k, все компоненты которой являются линейными функциями, число её значений есть степень двойки, а для любой функции F, все компоненты которой являются самодвойственными функциями, число её значений является чётным. Поэтому если $\mathcal{A} \subseteq P_{k|3}$, $[\mathcal{A}]_\beta = \mathcal{A}$ и $B(\mathcal{A}) = \mathcal{B}$, где класс $\mathcal{B}$ содержится в классе L или в классе S, то $\mathcal{A} \subseteq P_{k|2}$. В работе [6] доказано, что число различных β-замкнутых классов функций из $P_{k|2}$ с булевым замыканием $\mathcal{B}$ является конечным. Поэтому конечно и число различных β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием $\mathcal{B}$, а значит, ни один из классов линейных или самодвойственных функций не принадлежит множеству Q, и множество Q содержится в следующем множестве классов [8]:

$$\{P_2,\, T_0,\, T_1\} \cup \{O^\mu,\, I^\mu : \mu \geqslant 2\} \cup \{O^\infty,\, I^\infty\}.$$

В данном множестве классы O^∞ и I^∞ являются минимальными по включению, а по теореме 1 класс O^∞ содержится в множестве Q. Поскольку класс I^∞ является двойственным к классу O^∞, то и он, очевидно, содержится в Q. Поэтому имеет место следующая теорема

Теорема 3. Пусть $k = 2^m$, где $m \geqslant 2$. Тогда классы O^∞ и I^∞ являются минимальными по включению среди всех замкнутых классов $\mathcal{B}$ булевых функций, для которых число различных β-замкнутых классов функций из $P_{k|3}$ с булевым замыканием $\mathcal{B}$ не является конечным.

ЛИТЕРАТУРА

1. *Янов Ю. И., Мучник А. А.* О существовании k-значных замкнутых классов, не имеющих конечного базиса // Докл. АН СССР. 1959. Т. 127. № 1. С. 44–46.
2. *Яблонский С. В.* Введение в дискретную математику. М.: Высшая школа, 2001. 384 с.
3. *Нгуен Ван Хоа.* О семействах замкнутых классов k-значной логики, сохраняемых всеми автоморфизмами // Дискретная математика. 1993. Т. 5. Вып. 4. С. 87–108.
4. *Марченков С. С.* S-классификация функций многозначной логики // Дискретная математика. 1997. Т. 9. Вып. 3. С. 125–152.
5. *Тарасова О. С.* Классы функций k-значной логики, замкнутые относительно операций суперпозиции и перестановок // Матем. вопросы кибернетики. Сб. статей. Вып. 13. М.: Физматлит, 2004. С. 59–112.
6. *Подолько Д. К.* О классах функций, замкнутых относительно специальной операции суперпозиции // Вестник Московского университета. Сер. 1. Математика. Механика. 2013. № 6. С. 54–57.
7. *Кон П.* Универсальная алгебра. М.: Мир, 1968. 352 с.
8. *Угольников А. Б.* Классы Поста: учеб. пособие. М.: Изд-во ЦПИ при механико-математическом факультете МГУ им. М. В. Ломоносова, 2008. 64 с.
9. *Марченков С. С.* О замкнутых классах самодвойственных функций многозначной логики. II // Проблемы кибернетики. 1983. Т. 40. С. 261–266.

2014 Теоретические основы прикладной дискретной математики №2(24)

DOI 10.17223/20710410/24/3

УДК 519.4

ВОССТАНОВЛЕНИЕ ПОЛИНОМИАЛЬНО УСЛОЖНЁННОЙ ЛИНЕЙНОЙ РЕКУРРЕНТЫ МАКСИМАЛЬНОГО ПЕРИОДА НАД КОЛЬЦОМ ГАЛУА ПО СТАРШЕЙ КООРДИНАТНОЙ ПОСЛЕДОВАТЕЛЬНОСТИ

Е. М. Серебряков

Центр специальных разработок, г. Москва, Россия

E-mail: ryback_casey@mail.ru

Рассматриваются усложнения линейной рекуррентной последовательности (ЛРП) максимального периода над кольцом Галуа $\mathrm{GR}(q^n, p^n)$ с помощью некоторого многочлена $E(x)$ над этим кольцом, а также приводится алгоритм восстановления исходной рекурренты по старшей координатной последовательности полиномиально усложнённой ЛРП.

Ключевые слова: *ЛРП максимального периода, полиномиальное усложнение, старшая координатная последовательность, восстановление начального вектора.*

Введение

В криптографии актуальным остается вопрос о восстановлении линейной рекуррентной последовательности максимального периода (в частности, её начального отрезка) над кольцом Галуа по её старшей координатной последовательности. Ранее рядом авторов исследовалась старшая координатная последовательность u_{n-1} ЛРП максимального периода u и приводились соответствующие алгоритмы восстановления u по u_{n-1}. А. А. Нечаевым предложен алгоритм восстановления ЛРП максимального периода u над кольцом вычетов $R = \mathbb{Z}_{2^n}$ по её старшей координатной последовательности, впоследствии А. С. Кузьминым приведён алгоритм восстановления ЛРП максимального периода u над кольцом $R = \mathbb{Z}_{p^n}$ для произвольного $p \geqslant 3$.

В ряде работ отечественных [1, 2] и зарубежных авторов [3 – 6] уже исследовалась возможность восстановления усложнённой последовательности, причём усложнения задавались с помощью «инъективных отображений», то есть отображений, которые допускают однозначное восстановление исходной ЛРП по её усложнению.

Цель этой работы — для полиномиального усложнения ЛРП максимального периода u над кольцом Галуа $\mathrm{GR}(q^n, p^n)$ произвольным многочленом $E(x) \in R[x]$ степени $k \geqslant 1$ построить алгоритм восстановления начального отрезка исходной рекурренты по значениям старшей координатной последовательности усложнённой ЛРП.

1. Постановка задачи

Всюду далее $R = \mathrm{GR}(q^n, p^n)$ — кольцо Галуа характеристики p^n, состоящее из q^n элементов, где $n \in \mathbb{N}$, p простое, $q = p^r$. Согласно [7], множество pR всех делителей нуля есть идеал кольца R. Обозначим через $\bar{R} = R/pR = \mathrm{GF}(q)$ поле вычетов кольца R. Эпиморфизм $\varphi : R \to \bar{R}$, действующий по правилу $\varphi(a) = \bar{a}$ для всех $a \in R$, естественным образом можно продолжить на кольцо многочленов $R[x]$. Образ любого многочлена $F(x) \in R[x]$ при таком эпиморфизме будем обозначать через $\bar{F}(x) \in \bar{R}[x]$. Подмножество K кольца R называется *координатным множеством* кольца R, если

его элементы образуют полную систему вычетов по модулю идеала pR. Примером координатного множества является *p-адическое координатное множество* (координатное множество Тейхмюллера) $\Gamma(R) = \{\alpha \in R : \alpha^q = \alpha\}$. Для $R = \mathrm{GR}(p^n, p^n) = \mathbb{Z}_{p^n}$ ещё одним интересным примером координатного множества является *p-ичное координатное множество* $\delta(R) = \{0, 1, \ldots, p-1\}$. Известно [8], что любой элемент $a \in R$ однозначно представляется в виде

$$a = a_0 + pa_1 + ... + p^{n-1}a_{n-1}, \tag{1}$$

где $a_t \in K, t = 0, 1, \ldots, n-1$. Для всех $t \in \{0, \ldots, n-1\}$ определим функции $\gamma_t^K : R \to K$ следующим образом: $\gamma_t^K(a) = a_t$, где a_t определяется равенством (1).

Функция γ_t^K называется *t-й координатной функцией* в координатном множестве K. Естественным образом введём на множестве K операции, положив для любых $a, b \in K$

$$a \oplus b = \gamma_0^K(a + b), \quad a \otimes b = \gamma_0^K(ab).$$

Тогда $(K; \oplus, \otimes) = \mathrm{GF}(q)$ — поле, изоморфное $\bar{R}$, причём изоморфизм $\psi : K \to \bar{R}$ задаётся соотношением $\psi(a) = \bar{a}$, где $a \in K$. Изоморфизм ψ также естественным образом продолжается на множество всех последовательностей K^∞ над множеством K. Образ любой последовательности $v \in K^\infty$ при таком изоморфизме будем обозначать через $\overline{v} \in \overline{R}^\infty$. Стоит отметить, что по последовательности $\overline{v} \in \overline{R}^\infty$ можно однозначно восстановить последовательность $v \in K^\infty$.

В качестве результата умножения многочлена $F(x) = \sum\limits_{r \geqslant 0} f_r x^r \in R[x]$ на последовательность $u \in R^\infty$ определим последовательность $v = F(x)u$ из множества R^∞, знаки которой определяются следующим образом:

$$v(i) = \sum_{r \geqslant 0} f_r u(i + r), \quad i \geqslant 0.$$

Пусть $F(x) \in R[x]$ — многочлен максимального периода, имеющий степень m и период $T(F) = (q^m - 1)p^{n-1}$ [9]. Обозначим $L_R(F) = \{u \in R^\infty : F(x)u = (0)\}$ множество последовательностей, аннулируемых многочленом $F(x)$. Из работы [9] известно, что период любой рекурренты из $L_R(F)$ удовлетворяет неравенству $T(u) \leqslant (q^m - 1)p^{n-1}$. Последовательность с периодом $T(u) = (q^m - 1)p^{n-1}$ назовём *ЛРП максимального периода* (МП).

Пусть всюду далее $u \in L_R(F)$ — ЛРП максимального периода. Для любого многочлена $E(x) = \sum\limits_{j=0}^{k} e_j x^j \in R[x]$ определим последовательность $z = E(u) \in R^\infty$ со знаками $z(i) = \sum\limits_{j=0}^{k} e_j u(i)^j$ для всех $i \geqslant 0$. Последовательность $E(u)$ назовем *полиномиальным усложнением* рекурренты u с помощью многочлена $E(x)$ над кольцом R.

Для любой ЛРП $v \in R^\infty$ определим последовательность $v_t = \gamma_t^K(v) = (\gamma_t^K(v(0)), \gamma_t^K(v(1)), \ldots)$, которая называется *t-й координатной последовательностью* рекурренты v в координатном множестве K. Последовательность v_t является ЛРП над K, так как она периодическая. Последовательность v_{n-1} назовём *старшей координатной* последовательностью ЛРП v.

В данной работе будем решать следующую задачу: по известной старшей координатной последовательности $z_{n-1} = \gamma_{n-1}^K(z)$ полиномиально усложнённой рекурренты восстановить исходную ЛРП максимального периода u. Отметим, что в работе [10]

эта задача решена для произвольного кольца Галуа $R = \mathrm{GR}(q^n, p^n)$ и координатного множества K с некоторыми естественными ограничениями при $E(x) = x$.

Усложнения ЛРП u можно задавать с помощью функций $\Psi : R^n \to K$ следующим образом:

$$v(i) = \Psi(u_0(i), \ldots, u_{n-1}(i)).$$

Такие отображения индуцируют отображения семейств последовательностей $\hat{\Psi} : L_R(F) \to K^\infty$, которые называются «сжимающими отображениями». Как уже отмечалось, некоторые из таких отображений позволяют однозначно восстановить исходную ЛРП по её усложнению. Поставленную задачу можно свести к отображениям такого вида с помощью следующей леммы.

Лемма 1. Знак координатной последовательности полиномиально усложнённой ЛРП $z = E(u)$ представляется в виде $z_t(i) = \Psi(u_0(i), \ldots, u_t(i))$, $i \geqslant 0$, $t \in \{1, \ldots, n-1\}$, где

$$\Psi(x_0, x_1, \ldots, x_t) = E'(x_0)x_t + \eta(x_0, x_1, \ldots, x_{t-1}), \quad \eta(x_0, x_1, \ldots, x_{t-1}) \in K[x_0, x_1, \ldots, x_{t-1}].$$

Доказательство. Рассмотрим $E(x)$ над кольцом $R/p^{t+1}R$, где $x \equiv x_0 + px_1 + \ldots + p^t x_t \pmod{p^{t+1}}$:

$$E(x) \overset{p^{t+1}}{\equiv} \sum_{s=0}^{k} e_s \left(\sum_{j=0}^{t} p^j x_j \right)^s = \sum_{s=0}^{k} e_s \sum_{j_0+\ldots+j_t=s} \frac{s!}{j_0! \ldots j_t!} p^{j_1+2j_2+\ldots+tj_t} x_0^{j_0} \ldots x_t^{j_t}.$$

В этой сумме нас интересуют ненулевые слагаемые при p^t. Очевидно, что этому условию удовлетворяют лишь те слагаемые, у которых либо $j_t = 0$, либо $j_t = 1$, $j_0 = s - 1$, $j_1 = \ldots = j_{t-1} = 0$. Разбивая таким образом по j_t внутреннюю сумму на две части, $\Psi(x_0, \ldots, x_t)$ можно представить в виде

$$\Psi(x_0, \ldots, x_t) = \sum_{s=0}^{k} e_s \left(s p^t x_0^{s-1} x_t + \sum_{j_0+\ldots+j_{t-1}=s} \frac{s!}{j_0! \ldots j_{t-1}!} p^{j_1+2j_2+\ldots+(t-1)j_{t-1}} x_0^{j_0} \ldots x_{t-1}^{j_{t-1}} \right).$$

Значит, $\gamma_t^K(\Psi(x_0, \ldots, x_t)) = E'(x_0)x_t + \eta(x_0, \ldots, x_{t-1})$. ■

Заметим, что в [2] доказана принципиальная возможность восстановления исходной рекурренты, усложнённой с помощью функции $\Psi(x_0, \ldots, x_{n-1})$, полученной в лемме 1, однако нет конструктивного алгоритма восстановления.

2. Сечения

Одним из подходов к изучению свойств координатных последовательностей является метод сечений. Определим умножение элемента $\alpha \in K$ на элемент $\bar{r} \in \bar{R}$ равенством $\bar{r}\alpha = \gamma_0^K(r\alpha)$. Тогда K есть $\bar{R}$-алгебра и можно умножать любую последовательность над K на многочлен из $\bar{R}[x]$, а также рассматривать любую периодическую последовательность w над K как ЛРП над полем $\bar{R}$. Под сечением будем понимать умножение последовательности на некоторый многочлен, в результате чего получается более простая рекуррента. Обозначим $\tau_t = p^t(q^m - 1)$, $t = 0, 1, \ldots, n-1$, $\tau_0 = \tau$. В работе [11] доказана следующая лемма.

Лемма 2. Пусть $F(x) \in R[x]$ — многочлен максимального периода. Тогда существуют многочлены $\Phi^{(t+1)}(x) \in R[x]$, $t \in \{0, \ldots, n-1\}$, такие, что

$$\begin{gathered} x^{\tau_t} - e \equiv p^{t+1}\Phi^{(t+1)}(x) \pmod{F(x)}, \quad t \in \{0, \ldots, n-1\}, \\ \overline{\Phi}^{(t+1)}(x) \neq 0, \quad \deg \Phi^{(t+1)}(x) < m, \end{gathered} \tag{2}$$

причём, если $p \geqslant 3$, $t \geqslant 1$, то $\Phi^{(t)}(x) \equiv \Phi^{(t+1)}(x) \pmod{p^{t+1}}$.

Очевидно, что многочлены $\Phi^{(t)}(x)$, где $t \in \{1, \ldots, n\}$, определены неоднозначно, поэтому выберем их и зафиксируем. Последовательность $u^{(t)} = \Phi^{(t)}(x)u$ назовём *t-й производной последовательностью* рекурренты u. Нетрудно видеть, что $u^{(t)}$ — ЛРП максимального периода [8]. В работе [11] также доказано, что

$$\begin{array}{l} u_0^{(1)} = u_0^{(t)}, \quad t \geqslant 1, \\ u_1^{(2)} = u_1^{(t)}, \quad t \geqslant 2. \end{array} \tag{3}$$

Далее считаем, что элементы координатных множеств $K = \{\alpha_1, \ldots, \alpha_q\}$ и $\Gamma(R) = \{\beta_1, \ldots, \beta_q\}$ упорядочены так, что $\alpha_t \equiv \beta_t \pmod p$, $t = 0, \ldots, q-1$.

Известно [12], что существует единственный многочлен $\Psi_K(x) = \psi_{q-1}x^{q-1} + \ldots + \psi_1 x + \psi_0 \in R[x]$, такой, что $\Psi_K(\beta_t) = \alpha_t$, $t \in \{0, \ldots, q-1\}$, при этом $\Psi_K(x) \equiv x \pmod p$ и $\psi_0 = 0$. Многочлен $\Psi_K(x)$ называется *интерполяционным многочленом* координатного множества K. Обозначим через

$$\Psi_K^*(x) = \psi_{q-1}^* x^{q-1} + \ldots + \psi_1^* x \in R[x]$$

такой многочлен, что $\Psi_K(x) = x + p\Psi_K^*(x)$.

Пусть $F(x_1, \ldots, x_k) \in R[x_1, \ldots, x_k]$. Тогда через $\dfrac{\partial F}{\partial x_j}$ будем обозначать формальную частную производную многочлена F по переменной x_j для всех $j \in \{1, \ldots, k\}$. Эти производные являются многочленами из $R[x_1, \ldots, x_k]$. Через $\dfrac{\partial F}{\partial x_j}(a_1, \ldots, a_k)$ будем обозначать значение частной производной в точке $(a_1, \ldots, a_k) \in R^k$. В работе [12] доказана следующая лемма.

Лемма 3. Для любого $F(x_1, \ldots, x_k) \in R[x_1, \ldots, x_k]$ существует такой многочлен $F^*(x_1, \ldots, x_k)$, что $F(x_1, \ldots, x_k)^q = F(x_1^q, \ldots, x_k^q) + pF^*(x_1, \ldots, x_k)$, и для любых $a_1, \ldots, a_k \in K$

$$\begin{array}{c} \gamma_1^K(F(a_1, \ldots, a_k)) \equiv -F^*(a_1, \ldots, a_k) + \sum\limits_{i=1}^{k} \dfrac{\partial F}{\partial x_i}(a_1, \ldots, a_k)\Psi_K^*(a_i) - \\ - \Psi_K^*(F(a_1, \ldots, a_k)) \pmod p, \end{array} \tag{4}$$

в частности, для любых $a, b \in K$

$$\gamma_1^K(a+b) \equiv \sum_{j=1}^{p-1} \frac{1}{j!(p-j)!} a^{hj} b^{h(p-j)} + \Psi_K^*(a) + \Psi_K^*(b) - \Psi_K^*(a+b) \pmod p. \tag{5}$$

Для любого $s \in \{1, \ldots, n\}$ обозначим $b^{(s)} = E'(u)u^{(s)} \in R^\infty$, где $E'(x)$ — производная многочлена $E(x)$. Пусть также $b_t^{(s)} = \gamma_t^K(b^{(s)})$, $s \in \{1, \ldots, n\}$, $t = 0, 1$. Исследуем некоторые свойства последовательности $b^{(s)}$. Так как $b_0^{(s)}(i+\tau) = b_0^{(s)}(i)$ для всех $i \geqslant 0$, то $T(b_0^{(s)}) \mid \tau$ для всех $s \in \{1, \ldots, n\}$.

Лемма 4. При всех $t \in \{2, \ldots, n+1\}$ для координатной последовательности $b_1^{(t-1)} = \gamma_1^K(E'(u)u^{(t-1)})$ верно следующее равенство:

$$\begin{array}{c} b_1^{(t-1)} \equiv E'(u_0)\Psi_K^*(u_0^{(t-1)}) - \Psi_K^*(E'(u_0)u_0^{(t-1)}) - E_*'(u_0)u_0^{(t-1)} + \\ + E''(u_0)\Psi_K^*(u_0)u_0^{(t-1)} + E''(u_0)u_0^{(t-1)}u_1 + E'(u_0)u_1^{(t-1)} \pmod p, \end{array} \tag{6}$$

где $E_*'(x) \in R[x]$ — такой многочлен, что $E'(x)^q = E'(x^q) + pE_*'(x)$; $E''(x)$ — производная многочлена $E'(x)$.

Доказательство. Обозначим $v = E'(u)$, $w = u^{(t-1)}$, $v_j = \gamma_j^K(v)$, $w_j = \gamma_j^K(w)$, $j = 0, 1$. Тогда имеем

$$\begin{gathered} b_1^{(t-1)} = \gamma_1^K(vw) = \gamma_1^K((v_0 + pv_1)(w_0 + pw_1)) = \\ = \gamma_1^K(v_0w_0 + p(v_1w_0 + v_0w_1)) \equiv \gamma_1^K(v_0w_0) + v_1w_0 + v_0w_1 \pmod p. \end{gathered} \tag{7}$$

Заметим, что

$$v_0w_1 \equiv E'(u_0)u_1^{(t-1)} \pmod p. \tag{8}$$

Найдём $\gamma_1^K(v_0w_0)$. Возьмём в условиях леммы 3 в качестве $F(x_1, x_2)$ многочлен $x_1x_2 \in \in R[x_1, x_2]$, тогда из равенства (4) имеем

$$\begin{gathered} \gamma_1^K(v_0w_0) \equiv w_0\Psi_K^*(v_0) + v_0\Psi_K^*(w_0) - \Psi_K^*(v_0w_0) \equiv \\ \equiv u_0^{(t-1)}\Psi_K^*(E'(u_0)) + E'(u_0)\Psi_K^*(u_0^{(t-1)}) - \Psi_K^*(E'(u_0)u_0^{(t-1)}) \pmod p. \end{gathered} \tag{9}$$

Так как $E'(x) = \sum\limits_{j=1}^{k} je_jx^{j-1}$, то

$$\begin{gathered} v = E'(u) \equiv \sum_{j=1}^{k} je_j(u_0 + pu_1)^{j-1} \equiv \sum_{j=1}^{k} je_ju_0^{j-1} + pu_1\sum_{j=2}^{k} j(j-1)e_ju_0^{j-2} \equiv \\ \equiv E'(u_0) + pE''(u_0)u_1 \pmod{p^2}. \end{gathered} \tag{10}$$

Теперь, используя соотношения (10) и взяв в (4) в качестве $F(x_1, \ldots, x_k)$ многочлен $E'(x)$, получаем

$$\begin{gathered} v_1 \equiv \gamma_1^K(E'(u_0)) + E''(u_0)u_1 \equiv \\ \equiv -E'_*(u_0) + E''(u_0)\Psi_K^*(u_0) - \Psi_K^*(E'(u_0)) + E''(u_0)u_1 \pmod p. \end{gathered} \tag{11}$$

Умножая (11) на $w_0 = u_0^{(t-1)}$, находим

$$\begin{gathered} v_1w_0 \equiv -E'_*(u_0)u_0^{(t-1)} + E''(u_0)\Psi_K^*(u_0)u_0^{(t-1)} - \Psi_K^*(E'(u_0))u_0^{(t-1)} + \\ + E''(u_0)u_0^{(t-1)}u_1 \pmod p. \end{gathered} \tag{12}$$

Подставляя (8), (9) и (12) в соотношение (7), получаем (6). ■

Для $r, t \in \mathbb{N}$ обозначим

$$\delta(r = t) = \begin{cases} 1, & \text{если } r = t, \\ 0, & \text{если } r \neq t. \end{cases}$$

Обозначим $h = p^{r-1}$. Перейдём к построению сечений координатных последовательностей.

Теорема 1. Для координатных последовательностей ЛРП $z = E(u)$ справедливы следующие соотношения:

для $t = 1, \ldots, n-1$

$$(x^{\tau_{t-1}} - e)z_t \equiv E'(u)u_0^{(t)} \pmod p; \tag{13}$$

для $t = 2, \ldots, n-1$

$$(x^{\tau_{t-2}} - e)z_t \equiv \gamma_1^K(z_{t-1} + b^{(t-1)}) + \frac{1}{2}\delta(t = 2)E''(u)(u_0^{(t-1)})^2 \pmod p; \tag{14}$$

для $j = 1, \dots, p-1$, $t = 1, \dots, n-1$

$$(x^{\tau_{t-1}} - e)^j z_t^j \equiv j! \left(b_0^{(t)}\right)^j \equiv j! \left(E'(u_0) u_0^{(t)}\right)^j \pmod{p}; \tag{15}$$

для $j = 1, \dots, p-1$, $t = 1, \dots, n-1$

$$(x^{\tau_{t-1}} - e)^{j-1} z_t^j \equiv j! z_t \left(b_0^{(t)}\right)^{j-1} + \frac{j-1}{2} j! \left(b_0^{(t)}\right)^j \pmod{p}; \tag{16}$$

для $t = 2, \dots, n-1$

$$\begin{aligned}(x^{\tau} - e)^{p^{t-1}-p^{t-2}} z_t \equiv \left(z_{t-1} \left(b_0^{(t-1)}\right)^{p-1}\right)^h + \frac{p-2+2^{h-1}}{2^h} \left(b_0^{(t-1)}\right)^{ph} - \\ - \sum_{s=p-1}^{q-1} \psi_s^* \sum_{\substack{j_1+\dots+j_p=s,\\ j_1 \geqslant 0, j_2>0, \dots, j_p>0}} \frac{s!}{j_1! \dots j_p!} z_{t-1}^{j_1} \left(b_0^{(t-1)}\right)^{s-j_1} + \xi(p,t) \pmod{p},\end{aligned} \tag{17}$$

где

$$\xi(p,t) = \begin{cases} E'(u_0)\Phi^{(1)}(x) u_0^{(1)} + E''(u_0) \left(u_0^{(1)}\right)^2, \text{ если } p = 3, t = 2, \\ 0 \quad \text{в остальных случаях.} \end{cases}$$

Доказательство. Докажем равенство (13). Учитывая, что $T(u_t) = \tau_t$ [8], по лемме 1 получим $T(z_t) \mid \tau_t$. Тогда

$$(x^{\tau_{t-1}} - e) z \equiv p^t (x^{\tau_{t-1}} - e) z_t \pmod{p^{t+1}}. \tag{18}$$

С другой стороны, из соотношения (2) следует, что $(x^{\tau_{t-1}} - e) u \equiv p^t u_0^{(t)} \pmod{p^{t+1}}$, или

$$u(i + \tau_{t-1}) \equiv u(i) + p^t u_0^{(t)}(i) \pmod{p^{t+1}}. \tag{19}$$

Теперь, используя равенство (19), имеем для всех $i \geqslant 0$

$$\begin{aligned}(x^{\tau_{t-1}} - e) z(i) = z(i + \tau_{t-1}) - z(i) \equiv \sum_{j=0}^{k} e_j \left(u(i + \tau_{t-1})^j - u(i)^j\right) \equiv \\ \equiv \sum_{j=0}^{k} j e_j u(i)^{j-1} p^t u_0^{(t)}(i) = p^t E'(u(i)) u_0^{(t)}(i) \pmod{p^{t+1}}.\end{aligned} \tag{20}$$

Сравнивая соотношения (18) и (20), получаем равенство (13).

Докажем равенство (14). Используя (2), получаем

$$(x^{\tau_{t-2}} - e) u = p^{t-1} u^{(t-1)} \equiv p^{t-1} u_0^{(t-1)} + p^t u_1^{(t-1)} \pmod{p^{t+1}}.$$

Тогда для всех $i \geqslant 0$

$$\begin{aligned}u(i + \tau_{t-2})^j \equiv \left(u(i) + p^{t-1} u_0^{(t-1)}(i) + p^t u_1^{(t-1)}(i)\right)^j \equiv u(i)^j + j u(i)^{j-1} \left(p^{t-1} u_0^{(t-1)}(i) + \right. \\ \left. + p^t u_1^{(t-1)}(i)\right) + \binom{j}{2} u(i)^{j-2} \left(p^{t-1} u_0^{(t-1)}(i) + p^t u_1^{(t-1)}(i)\right)^2 \equiv u(i)^j + \\ + j u(i)^{j-1} \left(p^{t-1} u_0^{(t-1)}(i) + p^t u_1^{(t-1)}(i)\right) + \delta(t=2) \binom{j}{2} u(i)^{j-2} p^t (u_0^{(t-1)}(i))^2 \pmod{p^{t+1}}.\end{aligned} \tag{21}$$

Используя соотношение (21), получаем

$$(x^{\tau_{t-2}}-e)z(i)=z(i+\tau_{t-2})-z(i)\equiv\sum_{j=0}^{k}e_j\left(u(i+\tau_{t-2})^j-u(i)^j\right)\equiv$$

$$\equiv\sum_{j=0}^{k}e_j\left[ju(i)^{j-1}\left(p^{t-1}u_0^{(t-1)}(i)+p^tu_1^{(t-1)}(i)\right)+\delta(t=2)\binom{j}{2}u(i)^{j-2}p^tu_0^{(t-1)}(i)^2\right]\equiv \quad (22)$$

$$\equiv p^{t-1}E'(u(i))u_0^{(t-1)}(i)+p^tE'(u(i))u_1^{(t-1)}(i)+\frac{1}{2}\delta(t=2)p^tE''(u(i))(u_0^{(t-1)}(i))^2 \pmod{p^{t+1}}.$$

С другой стороны, используя соотношения $T(z_t)\mid\tau_t$ для всех $t\in\{0,\dots,n-1\}$ и равенство $z_t(i+\tau_{t-1})=\gamma_0^K\left(z_t(i)+E'(u(i))u^{(t)}(i)\right)$, вытекающее из (13), получаем

$$\begin{gathered}(x^{\tau_{t-2}}-e)z(i)\equiv(x^{\tau_{t-2}}-e)\left(p^{t-1}z_{t-1}(i)+p^tz_t(i)\right)\equiv\\ \equiv p^{t-1}\left(z_{t-1}(i+\tau_{t-2})-z_{t-1}(i)\right)+p^t(x^{\tau_{t-2}}-e)z_t(i)\equiv\\ \equiv p^{t-1}(\gamma_0^K\left(z_{t-1}(i)+E'(u(i))u^{(t-1)}(i)\right)-z_{t-1}(i))+p^t(x^{\tau_{t-2}}-e)z_t(i)\pmod{p^{t+1}}.\end{gathered}\quad(23)$$

Сравнивая (23) с соотношением (22), получаем

$$p(x^{\tau_{t-2}}-e)z_t\equiv$$

$$\equiv E'(u)u_0^{(t-1)}+pE'(u)u_1^{(t-1)}+\frac{1}{2}\delta(t=2)pE''(u)(u_0^{(t-1)})^2-\gamma_0^K(z_{t-1}+E'(u)u^{(t-1)})+z_{t-1}\equiv$$

$$\equiv p\gamma_1^K(z_{t-1}+E'(u)u^{(t-1)})+\frac{1}{2}\delta(t=2)pE''(u)(u_0^{(t-1)})^2\pmod{p^2}.$$

В итоге $(x^{\tau_{t-2}}-e)z_t\equiv\gamma_1^K\left(z_{t-1}+E'(u)u^{(t-1)}\right)+\frac{1}{2}\delta(t=2)E''(u)\left(u_0^{(t-1)}\right)^2\pmod{p}$, что и требовалось доказать.

Докажем сравнение (15). Для всех $i\geqslant 0$ и $s\in\overline{1,p-1}$, используя соотношение (13) и условие $T(b_0)|\tau_{t-1}$, имеем

$$\begin{gathered}(x^{\tau_{t-1}}-e)z_t^s(i)\equiv z_t^s(i+\tau_{t-1})-z_t^s(i)\equiv(z_t(i)+b_0^{(t)}(i))^s-z_t^s(i)\equiv\\ \equiv\sum_{j=0}^{s-1}\binom{s}{j}z_t^j(i)(b_0^{(t)}(i))^{s-j}\equiv g(z_t(i),b_0^{(t)}(i))\pmod{p},\end{gathered}\quad(24)$$

где $g(x,y)\in K[x,y]$; $\deg_x g\leqslant s-1$; $\deg_y g\leqslant s$.

Поэтому нетрудно видеть, что для любого $s\in\{1,\dots,p-1\}$ при $l>s$

$$(x^{\tau_{t-1}}-e)^lz_t^s\equiv 0\pmod{p}.$$

Используя (24), методом математической индукции покажем, что для всех $j\in\{1,\dots,p-1\}$ и $s\in\{1,\dots,j\}$ выполняется

$$\begin{gathered}(x^{\tau_{t-1}}-e)^sz_t^j\equiv j(j-1)\dots(j-s+1)z_t^{j-s}\left(b_0^{(t)}\right)^s+\\ +\frac{s}{2}j(j-1)\dots(j-s)z_t^{j-s-1}\left(b_0^{(t)}\right)^{s+1}+g^{(s)}\left(z_t,b_0^{(t)}\right)\pmod{p},\end{gathered}\quad(25)$$

где $g^{(s)}(x,y)\in K[x,y]$; $\deg_x g^{(s)}(x,y)\leqslant j-s-2$.

Проверим основание индукции при $s=1$. Имеем

$$(x^{\tau_{t-1}}-e)z_t^j\equiv\left(z_t+b_0^{(t)}\right)^j-z_t^j\equiv jz_t^{j-1}b_0^{(t)}+\frac{j(j-1)}{2}z_t^{j-2}\left(b_0^{(t)}\right)^2+g^{(1)}\left(z_t,b_0^{(t)}\right)\pmod{p},$$

где $\deg_x g^{(1)}(x,y)\leqslant j-3$.

Пусть равенство (25) верно для s, проверим его справедливость для $s+1$. Имеем

$$(x^{\tau_{t-1}}-e)^{s+1}z_t^j \equiv j(j-1)\dots(j-s+1)\left(b_0^{(t)}\right)^s\left((j-s)z_t^{j-s-1}b_0^{(t)}+\right.$$
$$+\frac{(j-s)(j-s-1)}{2}z_t^{j-s-2}\left(b_0^{(t)}\right)^2+g_1^{(s+1)}(z_t,b_0^{(t)})\Big)+$$
$$+\frac{s}{2}j(j-1)\dots(j-s)\left(b_0^{(t)}\right)^{s+1}\left((j-s-1)z_t^{j-s-2}b_0^{(t)}+\right.$$
$$\left.+g_2^{(s+1)}\left(z_t,b_0^{(t)}\right)\right)+g^{(s+1)}\left(z_t,b_0^{(t)}\right)\pmod p,$$

где $\deg_x g_1^{(s+1)} \leqslant j-s-3$; $\deg_x g_2^{(s+1)} \leqslant j-s-3$; $\deg_x g^{(s+1)} \leqslant j-s-3$.

Сгруппируем слагаемые с одинаковыми степенями z_t:

$$(x^{\tau_{t-1}}-e)^{s+1}z_t^j \equiv j(j-1)\dots(j-s)z_t^{j-(s+1)}\left(b_0^{(t)}\right)^{s+1}+$$
$$+\frac{s+1}{2}j(j-1)\dots(j-s-1)z_t^{j-s-2}\left(b_0^{(t)}\right)^{s+2}+\hat g^{(s+1)}\left(z_t,b_0^{(t)}\right)\pmod p.$$

Таким образом, сравнение (25) верно. Подставляя в него $s=k-1$ и $s=k$, получаем сравнения (15) и (16) соответственно.

Докажем сравнение (17). Напомним, что $b^{(t)}=E'(u)u^{(t)}$, $b_r^{(t)}=\gamma_r^K(b^{(t)})$, $r\in\{0,1\}$, $t\in\{1,\dots,n\}$. Тогда $b_0^{(t)}\equiv E'(u_0)u_0^{(t)} \pmod p$. Используя равенство (14), получаем

$$(x^{\tau_{t-2}}-e)z_t \equiv \gamma_1^K\left(z_{t-1}+b_0^{(t-1)}\right)+b_1^{(t-1)}+\frac{1}{2}\delta(t=2)E''(u_0)\left(u_0^{(t-1)}\right)^2 \pmod p. \quad (26)$$

Рассмотрим последовательное ($p-2$ раза) умножение $(x^{\tau_{t-2}}-e)$ на последовательность $\gamma_1^K\left(z_{t-1}+b_0^{(t-1)}\right)$. Из (5) имеем $\gamma_1^K\left(z_{t-1}+b_0^{(t-1)}\right)\equiv A+D \pmod p$, где

$$A=\sum_{j=1}^{p-1}\frac{1}{j!(p-j)!}z_{t-1}^{jh}\left(b_0^{(t-1)}\right)^{(p-j)h},\quad D=\Psi_K^*(z_{t-1})+\Psi_K^*\left(b_0^{(t-1)}\right)-\Psi_K^*\left(z_{t-1}+b_0^{(t-1)}\right).$$

Используя (25), методом математической индукции докажем, что для всех $s\in\{1,\dots,p-2\}$ верно

$$(x^{\tau_{t-2}}-e)^sA \equiv \frac{(p-1)^h\dots(p-s)^h}{(p-1)!}z_{t-1}^{(p-s-1)h}\left(b_0^{(t-1)}\right)^{(s+1)h}+$$
$$+\frac{(s+2^{h-1})(p-2)^h\dots(p-s-1)^h}{2^h(p-2)!}z_{t-1}^{(p-s-2)h}\left(b_0^{(t-1)}\right)^{(s+2)h}+ \quad (27)$$
$$+g^{(s)}\left(z_{t-1},b_0^{(t-1)}\right)\pmod p,$$

где $g^{(s)}(x,y)\in K[x,y]$; $\deg_x g^{(s)}\leqslant(p-s-3)h$.

Пользуясь тем, что характеристика поля K делит h, и используя сравнение (25) при $s=1$ и $k=j$, $j\in\{1,\dots,p-1\}$, получаем для всех $i\geqslant 0$

$$(x^{\tau_{t-2}}-e)z_{t-1}^{jh}(i)\equiv z_{t-1}(i+\tau_{t-2})^{jh}-z_{t-1}(i)^{jh}\equiv(z_{t-1}(i+\tau_{t-2})^j-z_{t-1}(i)^j)^h\equiv$$
$$\equiv\left((x^{\tau_{t-2}}-e)z_{t-1}^j(i)\right)^h\equiv j^hz_{t-1}(i)^{(j-1)h}b_0^{(t-1)}(i)^h+ \quad (28)$$
$$+\frac{j^h(j-1)^h}{2^h}z_{t-1}(i)^{(j-2)h}b_0^{(t-1)}(i)^{2h}+g\left(z_{t-1}(i),b_0^{(t-1)}(i)\right)\pmod p,$$

где $g(x,y)\in K[x,y]$; $\deg_x g\leqslant(j-3)h$.

Выделим в A следующие два слагаемых:

$$A = \frac{1}{(p-1)!} z_{t-1}^{(p-1)h} \left(b_0^{(t-1)}\right)^h + \frac{1}{2(p-2)!} z_{t-1}^{(p-2)h} \left(b_0^{(t-1)}\right)^{2h} + g^{(0)}\left(z_{t-1}, b_0^{(t-1)}\right),$$

где $g^{(0)}(x,y) \in K[x,y]$; $\deg_x g^{(0)} \leqslant (p-3)h$. Теперь, пользуясь тем, что $T(b_0^{(t-1)})|\tau_{t-2}$, докажем базу индукции, используя (28). Имеем

$$(x^{\tau_{t-2}} - e)A \equiv \frac{1}{(p-1)!} \left(b_0^{(t-1)}\right)^h (x^{\tau_{t-2}} - e) z_{t-1}^{(p-1)h} + \frac{1}{2(p-2)!} \left(b_0^{(t-1)}\right)^{2h} (x^{\tau_{t-2}} - e) z_{t-1}^{(p-2)h} +$$
$$+ (x^{\tau_{t-2}} - e) g^{(0)}\left(z_{t-1}, b_0^{(t-1)}\right) \equiv \frac{1}{(p-1)!} \left(b_0^{(t-1)}\right)^h \left((p-1)^h z_{t-1}^{(p-2)h} \left(b_0^{(t-1)}\right)^h + \right.$$
$$\left. + \frac{(p-1)^h (p-2)^h}{2^h} z_{t-1}^{(p-3)h} \left(b_0^{(t-1)}\right)^{2h} + g_1^{(1)}\left(z_{t-1}, b_0^{(t-1)}\right)\right) +$$
$$+ \frac{\left(b_0^{(t-1)}\right)^{2h}}{2(p-2)!} \left((p-2)^h z_{t-1}^{(p-3)h} \left(b_0^{(t-1)}\right)^h + g_2^{(1)}\left(z_{t-1}, b_0^{(t-1)}\right)\right) + \hat{g}^{(1)}\left(z_{t-1}, b_0^{(t-1)}\right) \pmod p,$$

где $g_1^{(1)}(x,y), g_2^{(1)}(x,y), \hat{g}^{(1)}(x,y) \in K[x,y]$; $\deg_x g_1^{(1)} \leqslant (p-4)h$; $\deg_x g_2^{(1)} \leqslant (p-4)h$; $\deg_x \hat{g}^{(1)} \leqslant (p-4)h$.

Группируем коэффициенты при одинаковых степенях z_{t-1}:

$$(x^{\tau_{t-2}} - e)A \equiv \frac{(p-1)^h}{(p-1)!} z_{t-1}^{(p-2)h} \left(b_0^{(t-1)}\right)^{2h} +$$
$$+ \left(\frac{(p-1)^h (p-2)^h}{2^h (p-1)!} + \frac{(p-2)^h}{2(p-2)!}\right) z_{t-1}^{(p-3)h} \left(b_0^{(t-1)}\right)^{3h} + g^{(1)}\left(z_{t-1}, b_0^{(t-1)}\right) \pmod p,$$

где $g^{(1)}(x,y) \in K[x,y]$; $\deg_x g^{(1)} \leqslant (p-4)h$.

Пользуясь тем, что h нечётное, получаем

$$(x^{\tau_{t-2}} - e)A \equiv \frac{(p-1)^h}{(p-1)!} z_{t-1}^{(p-2)h} \left(b_0^{(t-1)}\right)^{2h} + \frac{(2^{h-1}+1)(p-2)^h}{2^h (p-2)!} z_{t-1}^{(p-3)h} \left(b_0^{(t-1)}\right)^{3h} +$$
$$+ g^{(1)}\left(z_{t-1}, b_0^{(t-1)}\right) \pmod p,$$

где $g^{(1)}(x,y) \in K[x,y]$; $\deg_x g^{(1)} \leqslant (p-4)h$.

Используя сравнение (28), проверим шаг индукции:

$$(x^{\tau_{t-2}} - e)^{s+1} A \equiv \frac{(p-1)^h \dots (p-s)^h}{(p-1)!} \left(b_0^{(t-1)}\right)^{(s+1)h} (x^{\tau_{t-2}} - e) z_{t-1}^{(p-s-1)h} +$$
$$+ \frac{(s+2^{h-1})(p-2)^h \dots (p-s-1)^h}{2^h (p-2)!} \left(b_0^{(t-1)}\right)^{(s+2)h} (x^{\tau_{t-2}} - e) z_{t-1}^{(p-s-2)h} +$$
$$+ (x^{\tau_{t-2}} - e) g^{(s)}\left(z_{t-1}, b_0^{(t-1)}\right) \equiv \frac{(p-1)^h \dots (p-s)^h}{(p-1)!} \left(b_0^{(t-1)}\right)^{(s+1)h} \times$$
$$\times \left((p-s-1)^h z_{t-1}^{(p-s-2)h} \left(b_0^{(t-1)}\right)^h + \frac{(p-s-1)^h (p-s-2)^h}{2^h} z_{t-1}^{(p-s-3)h} \left(b_0^{(t-1)}\right)^{2h} + \right.$$

$$+g_1^{(s+1)}(z_{t-1},b_0^{(t-1)})\Big)+\frac{(s+2^{h-1})(p-2)^h\dots(p-s-1)^h}{2^h(p-2)!}\times$$
$$\times\left(b_0^{(t-1)}\right)^{(s+2)h}\left((p-s-2)^h z_{t-1}^{(p-s-3)h}\left(b_0^{(t-1)}\right)^h+\right.$$
$$\left.+\,g_2^{(s+1)}\left(z_{t-1},b_0^{(t-1)}\right)\right)+\hat{g}^{(s+1)}\left(z_{t-1},b_0^{(t-1)}\right)\pmod p,$$

где $g_1^{(s+1)}(x,y),g_2^{(s+1)}(x,y),\hat{g}^{(s+1)}(x,y)\in K[x,y]$; $\deg_x g_1^{(s+1)}\leqslant(p-s-4)h$; $\deg_x g_2^{(s+1)}\leqslant$ $\leqslant(p-s-4)h$; $\deg_x\hat{g}^{(s+1)}\leqslant(p-s-4)h$.

Раскрывая скобки и группируя коэффициенты при одинаковых степенях z_{t-1}, получаем

$$(x^{\tau_{t-2}}-e)^{s+1}A\equiv\frac{(p-1)^h\dots(p-s)^h(p-s-1)^h}{(p-1)!}z_{t-1}^{(p-s-2)h}\left(b_0^{(t-1)}\right)^{(s+2)h}+$$
$$+\frac{(p-1)^h\dots(p-s-2)^h}{2^h(p-1)!}z_{t-1}^{(p-s-3)h}\left(b_0^{(t-1)}\right)^{(s+3)h}+$$
$$+\frac{(s+2^{h-1})(p-2)^h\dots(p-s-2)^h}{2^h(p-2)!}z_{t-1}^{(p-s-3)h}\left(b_0^{(t-1)}\right)^{(s+3)h}+g^{(s+1)}\left(z_{t-1},b_0^{(t-1)}\right)\pmod p,$$

где $g^{(s+1)}(x,y)\in K[x,y]$; $\deg_x g^{(s+1)}\leqslant(p-s-4)h$. Отсюда следует справедливость сравнения (27).

Аналогично методом математической индукции доказывается, что для всех $s\in\{1,\dots,p-2\}$

$$(x^{\tau_{t-2}}-e)^s D\equiv-\sum_{r=s+1}^{q-1}\psi_r^*\sum_{\substack{j_1+\dots+j_{s+2}=r,\\ j_1\geqslant0,j_2>0,\dots,j_{s+2}>0}}\frac{r!}{j_1!\dots j_{s+2}!}z_{t-1}^{j_1}\left(b_0^{(t-1)}\right)^{r-j_1}\pmod p.\qquad(29)$$

Подставляя $s=p-2$ в равенства (27) и (29) и используя теорему Вильсона, получаем

$$(x^{\tau_{t-2}}-e)^{p-2}A\equiv\left(z_{t-1}\left(b_0^{(t-1)}\right)^{p-1}\right)^h+\frac{p-2+2^{h-1}}{2^h}\left(b_0^{(t-1)}\right)^q\pmod p,$$
$$(x^{\tau_{t-2}}-e)^{p-2}D\equiv-\sum_{r=p-1}^{q-1}\psi_r^*\sum_{\substack{j_1+\dots+j_p=r,\\ j_1\geqslant0,j_2>0,\dots,j_p>0}}\frac{r!}{j_1!\dots j_p!}z_{t-1}^{j_1}\left(b_0^{(t-1)}\right)^{r-j_1}\pmod p.$$

Таким образом,

$$(x^{\tau_{t-2}}-e)^{p-2}\gamma_1^K\left(z_{t-1}+b_0^{(t-1)}\right)\equiv\left(z_{t-1}\left(b_0^{(t-1)}\right)^{p-1}\right)^h+\frac{p-2+2^{h-1}}{2^h}\left(b_0^{(t-1)}\right)^{ph}-$$
$$-\sum_{r=p-1}^{q-1}\psi_r^*\sum_{\substack{j_1+\dots+j_p=r,\\ j_1\geqslant0,j_2>0,\dots,j_p>0}}\frac{r!}{j_1!\dots j_p!}z_{t-1}^{j_1}\left(b_0^{(t-1)}\right)^{r-j_1}\pmod p.\qquad(30)$$

Теперь вычислим $(x^{\tau_{t-2}}-e)^{p-2}b_1^{(t-1)}$. С учётом леммы 4 видно, что в соотношении (6) все слагаемые, кроме последних двух, имеют период, делящий τ, поэтому

$$(x^{\tau_{t-2}}-e)^{p-2}b_1^{(t-1)}\equiv E''(u_0)u_0^{(t-1)}(x^{\tau_{t-2}}-e)^{p-2}u_1+$$
$$+E'(u_0)(x^{\tau_{t-2}}-e)^{p-2}u_1^{(t-1)}\pmod p.\qquad(31)$$

Пользуясь равенствами (3) и соотношениями $T(u_1)\,|\tau_1$ и $(x^\tau - e)u_1 \equiv u_0^{(1)} \pmod p$, получаем

$$(x^{\tau_{t-2}} - e)u_1 = \begin{cases} u_0^{(1)}, & \text{если } t = 2, \\ 0, & \text{если } t > 2. \end{cases}$$

Так как $T(u_0^{(1)}) \mid \tau_0$, имеем

$$(x^{\tau_{t-2}} - e)^{p-2}u_1 = \begin{cases} u_0^{(1)}, & \text{если } p = 3, t = 2, \\ 0 & \text{в противном случае.} \end{cases} \tag{32}$$

В итоге из равенств (32) вытекает

$$E''(u_0)u_0^{(t-1)}(x^{\tau_{t-2}} - e)^{p-2}u_1 = \begin{cases} E''(u_0)(u_0^{(1)})^2, & \text{если } p = 3, t = 2, \\ 0 & \text{в противном случае.} \end{cases} \tag{33}$$

Рассмотрим $(x^{\tau_{t-2}} - e)u_1^{(t-1)}$:

$$\begin{gathered}(x^{\tau_{t-2}} - e)u_1^{(t-1)} \equiv (x^{\tau_{t-2}} - e)\gamma_1^K\left(u^{(t-1)}\right) \equiv (x^{\tau_{t-2}} - e)\gamma_1^K\left(\Phi^{(t-1)}(x)u\right) \equiv \\ \equiv (x^{\tau_{t-2}} - e)\gamma_1^K\left(\Phi^{(1)}\,(x)u\right) \equiv (x^{\tau_{t-2}} - e)\gamma_1^K\left(\Phi^{(1)}(x)(u_0 + pu_1)\right) \equiv \\ \equiv (x^{\tau_{t-2}} - e)\gamma_1^K\left(\Phi^{(1)}(x)u_0 + p\Phi^{(1)}(x)u_1\right) \equiv (x^{\tau_{t-2}} - e)\gamma_1^K\left(\Phi^{(1)}(x)u_0\right) + \\ +(x^{\tau_{t-2}} - e)\left(\Phi^{(1)}(x)u_1\right) \pmod p.\end{gathered}$$

Так как период последовательности $\gamma_1^K\left(\Phi^{(1)}(x)u_0\right)$ делит τ, то, используя соотношения $T(u_1) \mid \tau_1$ и $(x^\tau - e)u_1 \equiv u_0^{(1)} \pmod p$, получаем

$$(x^{\tau_{t-2}} - e)u_1^{(t-1)} \equiv (x^{\tau_{t-2}} - e)\Phi^{(1)}(x)u_1 \equiv \begin{cases} \Phi^{(1)}(x)u_0^{(1)}, & \text{если } t = 2, \\ 0 & \text{в противном случае.} \end{cases} \tag{34}$$

Пользуясь (34) и соотношением $T(u_0^{(1)}) \mid \tau$, получаем

$$E'(u_0)(x^{\tau_{t-2}} - e)^{p-2}u_1^{(t-1)} \equiv \begin{cases} E'(u_0)\Phi^{(1)}(x)u_0^{(1)}, & \text{если } p = 3, t = 2, \\ 0 & \text{в противном случае.} \end{cases} \tag{35}$$

С учётом (33) и (35) равенство (31) примет вид

$$(x^{\tau_{t-2}} - e)^{p-2}b_1^{(t-1)} \equiv \begin{cases} E'(u_0)\Phi^{(1)}(x)u_0^{(1)} + E''(u_0)(u_0^{(1)})^2, & \text{если } t = 2,\ p = 3, \\ 0 & \text{в противном случае.} \end{cases} \tag{36}$$

Таким образом, с использованием сравнения (26) имеем

$$(x^{\tau_{t-2}} - e)^{p-1}z_t \equiv (x^{\tau_{t-2}} - e)^{p-2}\gamma_1^K(z_t + b_0^{(t-1)}) + (x^{\tau_{t-2}} - e)^{p-2}b_1^{(t-1)} \pmod p. \tag{37}$$

Объединяя результаты (30), (36) и (37), получаем

$$\begin{gathered}(x^\tau - e)^{p^{t-1}-p^{t-2}}z_t \equiv \left(z_{t-1}\left(b_0^{(t-1)}\right)^{p-1}\right)^h + \frac{p - 2 + 2^{h-1}}{2^h}\left(b_0^{(t-1)}\right)^{ph} - \\ - \sum_{r=p-1}^{q-1}\psi_r^* \sum_{\substack{j_1+\ldots+j_p=r,\\ j_1\geqslant 0, j_2>0,\ldots,j_p>0}} \frac{r!}{j_1!\ldots j_p!}z_{t-1}^{j_1}\left(b_0^{(t-1)}\right)^{r-j_1} + \xi(p,t),\end{gathered}$$

где

$$\xi(p,t) = \begin{cases} E'(u_0)\Phi^{(1)}(x)u_0^{(1)} + E''(u_0)(u_0^{(1)})^2, & \text{если } t = 2, p = 3, \\ 0 & \text{в противном случае.} \end{cases}$$

Теорема доказана. ■

3. Алгоритм восстановления

Прежде чем перейти к алгоритму восстановления, приведем еще несколько результатов. Для всех $t \in \mathbb{N}$ обозначим

$$t = \sum_{i \geqslant 0} \nu_i(t) p^i, \quad \nu_i(t) \in \{0, \ldots, p-1\}, \quad w_p(t) = \sum_{i \geqslant 0} \nu_i(t),$$
$$t = \sum_{i \geqslant 0} \rho_i(t) q^i, \quad \rho_i(t) \in \{0, \ldots, q-1\}, \quad w_q(t) = \sum_{i \geqslant 0} \rho_i(t),$$

где $\nu_i(t)$ — коэффициент при p^i в p-ичном разложении; $\rho_i(t)$ — коэффициент при q^i в q-ичном разложении; $w_p(t)$ — *p-ичный вес* числа t; $w_q(t)$ — *q-ичный вес* числа t.

Для доказательства основного результата потребуется ещё одна лемма.

Лемма 5 [13]. Пусть $\alpha \in \mathbb{N}$, $k_1, \ldots, k_\alpha \in \mathbb{N}_0$, $k = k_1 + \ldots + k_\alpha$, ε — максимальная степень простого числа p, делящая полиномиальный коэффициент $\dfrac{k!}{k_1! \ldots k_\alpha!}$. Тогда

а) $\varepsilon = \dfrac{w_p(k_1) + \ldots + w_p(k_\alpha) - w_p(k)}{p-1}$;

б) $\dfrac{k!}{k_1! \ldots k_\alpha!} \equiv (-p)^\varepsilon \prod\limits_{m \geqslant 0} \dfrac{\nu_m(k)!}{\nu_m(k_1)! \ldots \nu_m(k_\alpha)!} \pmod{p^{\varepsilon+1}}$.

Так как $\bar{F}(x) \in \bar{R}[x]$ — примитивный многочлен, то существует корень ω этого многочлена в расширении Галуа $\bar{S} = \mathrm{GF}(q^m)$ поля $\bar{R}$. Известно, что любая последовательность из $L_{\bar{R}}(\bar{F})$ представляется в виде линейной комбинации биномиальных последовательностей с корнями из множества корней многочлена $\bar{F}(x) \in \bar{R}[x]$.

Лемма 6. Пусть $E(x) = \sum\limits_{r=0}^{k} e_r x^r$ и $\deg \bar{E}(x) = k' \in \{1, \ldots, q-1\}$. Тогда последовательность $b_0^{(1)}$ представляется в виде

$$b_0^{(1)} = (s+1)\bar{e}_{s+1} u_0^s \otimes u_0^{(1)} \oplus l, \quad s \in \{0, \ldots, k'-1\}, \quad (\mu_l(x), \mu_s(x)) = \bar{e},$$

где $l \in K^\infty$; $\mu_l(x)$ — минимальный многочлен ЛРП l; $\mu_s(x)$ — минимальный многочлен ЛРП $u_0^s u_0^{(1)}$.

Доказательство. Из определения $b_0^{(1)}$ получаем

$$b_0^{(1)} = \bar{E}'(u_0) \otimes u_0^{(1)} = \sum_{r=0}^{k'-1} \oplus (r+1)\bar{e}_{r+1} u_0^r \otimes u_0^{(1)}.$$

Для доказательства леммы достаточно показать, что для двух разных $r_1, r_2 \in \{0, \ldots, k'-1\}$ выполняется $(\mu_{r_1}(x), \mu_{r_2}(x)) = \bar{e}$, где $\mu_{r_i}(x)$ — минимальный многочлен последовательности $u_0^{r_i} u_0^{(1)}$, $i = 1, 2$. Воспользуемся биномиальным представлением линейных рекуррент u_0 и $u_0^{(1)}$:

$$u_0(i) \equiv \sum_{j=0}^{m-1} (\xi \omega^i)^{q^j} \pmod{p}, \quad u_0^{(1)}(i) \equiv \sum_{j=0}^{m-1} (\xi \omega^d \omega^i)^{q^j} \pmod{p},$$

где ω — корень $\bar{F}(x)$ в поле $\bar{S}$; $\xi \in \bar{S}$; $d \in \{0, \ldots, \tau-1\}$ — такое, что $x^d \equiv \overline{\Phi}^{(1)}(x) \pmod{\bar{F}(x)}$. Для любого $r \in \{0, \ldots, k'-1\}$ представим ЛРП $u_0^r \otimes u_0^{(1)}$ в виде сумм биномиальных последовательностей:

$$u_0^r(i) \otimes u_0^{(1)}(i) \equiv \sum_{\substack{j_0 + \ldots + j_{m-1} = r, \\ t \in \{0, \ldots, m-1\}}} \frac{r!}{j_0! \ldots j_{m-1}!} \xi^{\sum\limits_{\alpha=0}^{m-1} j_\alpha q^\alpha + q^t} \omega^{q^t d} \omega^{i\left(\sum\limits_{\alpha=0}^{m-1} j_\alpha q^\alpha + q^t\right)} \pmod{p}, \ i \geqslant 0.$$

Отсюда следует, что $w_q\left(\sum\limits_{\alpha=0}^{m-1} j_\alpha q^\alpha + q^t\right) = r+1$. Значит, $u_0^r \otimes u_0^{(1)}$ представляется в виде линейной комбинации биномиальных последовательностей с коэффициентами из $\bar{S}$ и корнями вида ω^t, где $t \in \{0, \ldots, \tau-1\}$, $w_q(t) = r+1$. Таким образом, множества корней в поле разложения многочленов $\mu_{r_1}(x)$ и $\mu_{r_2}(x)$ не пересекаются при $r_1 \neq r_2$; значит, они взаимно просты. ■

Лемма 7. Пусть знаки последовательности u_0 представляются в виде функции след $u_0(i) \equiv \operatorname{tr}(\xi\omega^i) \pmod p$, где $\xi \in \bar{S}$, $i \geqslant 0$. Тогда для любого $s \in \{0, \ldots, q-2\}$ последовательность $u_0^s \otimes u_0^{(1)}$ над координатным множеством K представляется в виде

$$u_0^s \otimes u_0^{(1)} = l_1 \oplus l_2,$$

где $l_1, l_2 \in K^\infty$; $\mu_{l_1}(x)$ — минимальный многочлен последовательности l_1; $\mu_{l_2}(x)$ — минимальный многочлен последовательности l_2; $l_1 \equiv \operatorname{tr}(\xi^{s+1}\omega^{i(s+1)+d}) \pmod p$ и $(\mu_{l_1}(x), \mu_{l_2}(x)) = \bar{e}$.

Доказательство. Из доказательства леммы 6 следует, что для любого $s \in \{0, \ldots, q-2\}$ последовательность $u_0^s \otimes u_0^{(1)}$ представляется в виде

$$u_0^s(i) \otimes u_0^{(1)}(i) \equiv \sum_{\substack{j_0+\ldots+j_{m-1}=s,\\ t\in\{0,\ldots,m-1\}}} \frac{s!}{j_0!\ldots j_{m-1}!} \xi^{\sum\limits_{\alpha=0}^{m-1} j_\alpha q^\alpha + q^t} \omega^{q^t d} \omega^{i\left(\sum\limits_{\alpha=0}^{m-1} j_\alpha q^\alpha + q^t\right)} \pmod p, \quad i \geqslant 0.$$

С другой стороны, период последовательности $u_0^s \otimes u_0^{(1)}$ делит $\tau_0 = q^m - 1$. Так как эта последовательность над полем K и q взаимно просто с τ_0, то её минимальный многочлен $\mu(x)$ есть произведение попарно различных унитарных неприводимых многочленов, степени которых делят m, а сама последовательность представляется в виде суммы

$$u_0^s(i) \otimes u_0^{(1)}(i) \equiv \sum_{r\in C} \operatorname{tr}(a_r\omega^{ri}) \pmod p, \qquad i \geqslant 0, \tag{38}$$

где C — множество минимальных представителей q-циклотомических классов, на которые разбивается множество $\{0, \ldots, \tau_0-1\}$. Слагаемое вида $a\omega^{(s+1)i}$ появляется в (38) при условии

$$q^t + j_0 + qj_1 + \ldots + q^{m-1}j_{m-1} = s+1 \pmod{q^m-1}.$$

Такое возможно тогда и только тогда, когда $t = 0$, $j_0 = s$, $j_1 = \ldots = j_{m-1} = 0$. Значит, $a = \xi^{s+1}\omega^d$. ■

Индексом нелинейности ненулевого одночлена lx^t будем называть число $\operatorname{ind} lx^t = w_p(t)$ [10]. Для нулевого многочлена индекс нелинейности положим равным $-\infty$. Индексом нелинейности $\operatorname{ind}\psi(x)$ ненулевого многочлена $\psi(x)$ будем называть максимум индексов нелинейности его мономов. Поскольку интерполяционный многочлен произвольного координатного множества K имеет степень, не превышающую $q-1$, то

$$1 \leqslant \operatorname{ind}\Psi_K(x) \leqslant (p-1)r.$$

Отметим, что, согласно [14], многочлен $\bar{E}(x)$ как функция над конечным полем $\bar{R}$ из q элементов может быть представлен многочленом $\tilde{E}(x)$ над этим же полем, степень которого не превышает $q-1$. Поэтому в дальнейшем считаем, что $\deg\bar{E}(x) \leqslant q-1$.

Получим теперь основной результат.

Теорема 2. Пусть $E(x) = \sum\limits_{s=0}^{k} e_s x^s \in R[x]$, $u \in L_R(F)$ — ЛРП максимального периода, $z = E(u)$ и выполнены следующие условия:

$$\bar{E}'(x) \text{ не имеет корней в поле } \bar{R}; \tag{39}$$

$$\operatorname{ind} \Psi_K(x) \leqslant p-1; \tag{40}$$

существует такое $s \in \{0, \ldots, k-1\}$, что $(s+1, q^m-1) = 1$ и $\bar{e}_{s+1} \neq \bar{0}$.

Тогда по последовательности $z_{n-1} = \gamma_{n-1}^K(E(u))$ однозначно восстанавливается начальный отрезок $u(0), \ldots, u(m-1)$.

Доказательство. Приведём конструктивный алгоритм восстановления $u(0), \ldots, u(m-1)$ по z_{n-1}.

Э т а п I. Полагая в соотношении (13) $t = n-1$, находим последовательность $b_0^{(1)}$. По условию существует такое $s \in \{0, \ldots, k-1\}$, что коэффициент многочлена $\bar{e}_{s+1} \neq \bar{0}$. Согласно лемме 6, ЛРП $b_0^{(1)}$ представляется в виде $b_0^{(1)} = (s+1)\bar{e}_{s+1} u_0^s \otimes u_0^{(1)} \oplus l$, причём минимальный многочлен $\mu_l(x)$ ЛРП l взаимно прост с минимальным многочленом ЛРП $u_0^s \otimes u_0^{(1)}$. Отсюда по известной последовательности $b_0^{(1)}$ однозначно восстанавливается последовательность $u_0^s \otimes u_0^{(1)}$.

Согласно лемме 7, по ЛРП $u_0^s \otimes u_0^{(1)}$ можно восстановить последовательность $\operatorname{tr}(\xi^{s+1}\omega^d \omega^{(s+1)i})$, $i \geqslant 0$, а по ней ξ^{s+1}. Пользуясь тем, что $(s+1, q^m-1) = 1$, находим величину ξ. Таким образом, последовательность u_0 восстановлена.

Э т а п II. Поскольку u_0 — ЛРП МП, то последовательность $u_0^{(1)}$ — её сдвиг, и она также является рекуррентой максимального периода с периодом $\tau_0 = q^m - 1$. Тогда существует такое $i_0 \in \mathbb{N}_0$, что $u_0^{(1)}(i_0 + j) \neq 0$ для всех $j \in \{0, \ldots, m-1\}$.

Э т а п III. Из условия (40) следует, что $\operatorname{ind} \Psi_K^*(x) \leqslant p-1$, т. е. $\psi_s^* \neq 0$ только при $w_p(s) \leqslant p-1$. Согласно лемме 5, полиномиальный коэффициент во внутренней сумме сечения (17) не равен 0 по модулю p при условии $w_p(j_1) + \ldots + w_p(j_p) = w_p(s)$. Это возможно только при $j_1 = 0$, $j_2 = 1$, …, $j_p = 1$. Тогда в двойной сумме в (17) остаются только слагаемые вида $s!\psi_s^* \left(b_0^{(t-1)}\right)^s$.

Пользуясь тем, что для всех $t \in \{2, \ldots, n-1\}$ и $j \in \{0, \ldots, m-1\}$ верно $b_0^{(t)}(i_0+j) = b_0^{(1)}(i_0+j) = \bar{E}'\left(u_0(i_0+j)\right) u_0^{(1)}(i_0+j) \neq 0$, так как $\bar{E}'(u_0(i_0+j)) \neq 0$ в силу условия (39) теоремы и $u_0^{(1)}(i_0+j) \neq 0$ по выбору i_0, и полагая в сечении (17) последовательно $t = n-1, n-2, \ldots, 2$, находим $z_s(i_0+j)$ для всех $s \in \{1, \ldots, n-2\}$ и $j \in \{0, \ldots, m-1\}$. Знаки последовательности $\xi(p,t)$ и двойной суммы однозначно вычисляются по известным знакам последовательностей u_0 и $u_0^{(1)}$.

Находим $z_0 = \bar{E}(u_0)$. Таким образом, узнаем значения ЛРП z над кольцом R в тактах $i_0, i_0+1, \ldots, i_0+m-1$.

Э т а п IV. Теперь для каждого $j \in \{0, \ldots, m-1\}$ по известным знакам $z_t(i_0+j)$, $t \in \{1, \ldots, n-1\}$, используя результат леммы 1 и условие (39), находим знаки последовательностей $u_t(i_0+j)$, $t \in \{1, \ldots, n-1\}$.

Э т а п V. Начальный вектор $(u(0), \ldots, u(m-1))$ вычисляется по известному вектору $(u(i_0), u(i_0+1), \ldots, u(i_0+m-1))$ с использованием закона рекурсии. ■

Заметим, что условие $u_0^{(1)}(i_0+j) \neq 0$ для всех $j \in \{0, \ldots, m-1\}$, введённое на этапе II алгоритма, можно ослабить: для восстановления начального вектора ЛРП u достаточно найти знаки $u(i)$ в тактах $i_1, \ldots, i_m$, таких, что система вычетов по модулю $\bar{F}(x)$ многочленов $x^{i_1}, \ldots, x^{i_m}$ линейно независима над $\bar{R}$. Поэтому в последова-

тельности $u_0^{(1)}$ достаточно найти m ненулевых значений $u_0^{(1)}(i_j)$, $j = 0, \ldots, m-1$, для которых система вычетов по модулю $\bar{F}(x)$ многочленов $x^{i_1}, \ldots, x^{i_m}$ линейно независима над $\bar{R}$. При этом на этапе V начальный вектор $(u(0), \ldots, u(m-1))$ вычисляется по известному вектору $(u(i_1), u(i_2), \ldots, u(i_m))$ следующим образом:

$$(u(0), \ldots, u(m-1)) = (u(i_1), u(i_2), \ldots, u(i_m))A^{-1},$$

где $A = (S(F)^{i_1}\mathbf{e}, \ldots, S(F)^{i_m}\mathbf{e})$; $S(F)$ — сопровождающая матрица многочлена $F(x)$; $\mathbf{e} = (e, 0, \ldots 0)$. Матрица A обратима в указанных условиях.

Следствие 1. Пусть $K = \Gamma(R)$, $E(x) = \sum\limits_{s=0}^{k} e_s x^s \in R[x]$, $u \in L_R(F)$ — ЛРП максимального периода, $z = E(u)$ и выполнены следующие условия:

1) $\bar{E}'(x)$ не имеет корней в поле $\bar{R}$;
2) существует такое $s \in \{0, \ldots, k-1\}$, что $(s+1, q^m - 1) = 1$ и $\bar{e}_{s+1} \neq \bar{0}$.

Тогда по последовательности $z_{n-1} = \gamma_{n-1}^K(E(u))$ однозначно восстанавливается начальный отрезок $u(0), \ldots, u(m-1)$.

Следствие 2. Пусть $R = \mathbb{Z}_{p^n}$, K — произвольное координатное множество кольца R, $E(x) = \sum\limits_{s=0}^{k} e_s x^s \in R[x]$, $u \in L_R(F)$ — ЛРП максимального периода, $z = E(u)$ и выполнены следующие условия:

1) $\bar{E}'(x)$ не имеет корней в поле $\bar{R}$;
2) существует такое $s \in \{0, \ldots, k-1\}$, что $(s+1, q^m - 1) = 1$ и $\bar{e}_{s+1} \neq \bar{0}$.

Тогда по последовательности $z_{n-1} = \gamma_{n-1}^K(E(u))$ однозначно восстанавливается начальный отрезок $u(0), \ldots, u(m-1)$.

Отметим, что при дополнительных требованиях ограничение (39) на $\bar{E}'(x)$ в условиях теоремы 2 можно ослабить.

Для любого $i \geqslant 0$ знак производной последовательности можно представить в виде $u_0^{(1)}(i) = \mathrm{tr}(\xi\Phi^{(1)}(\omega)\omega^i) = \mathrm{tr}(\xi\omega^d\omega^i)$, где $d \in \{0, \ldots, \tau-1\}$. Найдём такие $d \in \{0, \ldots, \tau-1\}$, при которых $\omega^d \in \bar{R}$. Очевидно, что это включение выполняется только при условии $(q^m - 1) \mid d(q-1)$. Этому соотношению удовлетворяют только числа вида $d = y(1 + {} + q + \ldots + q^{m-1})$, $y \in \{1, \ldots, q-1\}$. Для таких d можно воспользоваться свойством линейности функции след и представить рекурренту $u_0^{(1)}$ в виде $u_0^{(1)} = cu_0$, где c — некоторая константа из $\bar{R}$. Таким образом, верно следующее следствие.

Следствие 3. Пусть в условиях теоремы 2 для многочлена максимального периода $F(x) \in R[x]$ определено число $d \in \{0, \ldots, \tau-1\}$, такое, что $\bar{\Phi}^{(1)}(x) \equiv x^d \pmod{\bar{F}(x)}$, и выполнены следующие условия:

1) существует такое $\bar{r} \in \bar{R}$, что $\bar{E}'(\bar{r}) \neq \bar{0}$;
2) d представляется в виде $d = y(1 + q + \ldots + q^{m-1})$, $y \in \{1, \ldots, q-1\}$;
3) $\mathrm{ind}\,\Psi_K(x) \leqslant p - 1$;
4) существует такое $s \in \{0, \ldots, k-1\}$, что $(s+1, q^m - 1) = 1$ и $\bar{e}_{s+1} \neq \bar{0}$.

Тогда по последовательности $z_{n-1} = \gamma_{n-1}^K(E(u))$ однозначно восстанавливается начальный отрезок $u(0), \ldots, u(m-1)$.

Автор выражает благодарность Д. Н. Былкову за помощь в написании статьи и уточнении некоторых результатов.

ЛИТЕРАТУРА

1. *Кузьмин А. С., Маршалко Г. Б., Нечаев А. А.* Восстановление линейной рекурренты над примарным кольцом вычетов по её усложнению // Математические вопросы криптографии. 2010. Т. 1. № 2. С. 31–56.
2. *Былков Д. Н.* Класс усложнений линейных рекуррент над кольцом Галуа, не приводящий к потере информации // Проблемы передачи информации. 2010. Т. 46. № 3. С. 51–59.
3. *Tian T. and Wen-Feng Q.* Injectivity of compressing map on primitive sequences over $\mathbb{Z}/(p^e)$ // IEEE Trans. Inform. Theory. 2007. V. 53. No. 8. P. 2960–2966.
4. *Xuan-Yong Z. and Wen-Feng Q.* Uniqueness of the distribution of zeros of primitive level sequences over $\mathbb{Z}/(p^e)$ // Finite Fields Their Appl. 2005. V. 11. No. 1. P. 30–44.
5. *Xuan-Yong Z. and Wen-Feng Q.* Compression mappings on primitive sequences over $\mathbb{Z}/(p^e)$ // IEEE Trans. Inform. Theory. 2004. V. 50. No. 10. P. 2442–2448.
6. *Xuan-Yong Z. and Wen-Feng Q.* Further result of compressing maps on primitive sequences modulo odd prime powers // IEEE Trans. Inform. Theory. 2007. V 53. No. 8. P. 2985–2990.
7. *Нечаев А. А.* Код Кердока в циклической форме // Дискретная математика. 1989. Т. 4. № 1. С. 123–139.
8. *Кузьмин А. С., Нечаев А. А.* Линейные рекуррентные последовательности над кольцами Галуа // Алгебра и логика. 1995. Т. 3. № 2. С. 169–189.
9. *Нечаев А. А.* Цикловые типы линейных подстановок над конечными коммутативными локальными кольцами // Математич. сб. 1993. Т. 184. № 3. С. 21–56.
10. *Кузьмин А. С., Нечаев А. А.* Восстановление линейной рекуррентной последовательности над кольцом Галуа по её старшей координатной последовательности // Дискретная математика. 2011. Т. 23. № 2. С. 3–31.
11. *Кузьмин А. С., Куракин В. Л., Нечаев А. А.* Псевдослучайные и полилинейные последовательности // Труды по дискретной математике. 1997. Т. 1. С. 139–202.
12. *Куракин В. Л.* Функция переноса в первый разряд в кольце Галуа // Дискретная математика. 2012. Т. 24. № 2. С. 21–36.
13. *Куракин В. Л.* Представления над кольцом $\mathbb{Z}_{p^n}$ линейной рекуррентной последовательности максимального периода над полем $\mathrm{GF}(p)$ // Дискретная математика. 1992. Т. 4. № 4. С. 96–116.
14. *Глухов М. М., Елизаров В. П., Нечаев А. А.* Алгебра. Т. 2. М.: Гелиос, 2003.

2014 Теоретические основы прикладной дискретной математики №2(24)

DOI 10.17223/20710410/24/4

УДК 519.719.325

ВЫЧИСЛЕНИЕ СТЕПЕНИ НЕЛИНЕЙНОСТИ ФУНКЦИИ НА ЦИКЛИЧЕСКОЙ ГРУППЕ ПРИМАРНОГО ПОРЯДКА

А. В. Черемушкин

Институт криптографии, связи и информатики, г. Москва, Россия

E-mail: avc238@mail.ru

Предлагается способ вычисления степени нелинейности дискретных функций, заданных на циклической группе примарного порядка, основанный на свойствах разложения Ньютона. Найдены значения степени нелинейности для базисных функций этого разложения. Для циклических групп порядков p^2 и p^3 приводится распределение числа функций с заданным значением степени нелинейности.

Ключевые слова: *дискретные функции, степень нелинейности, разложение Ньютона.*

Настоящая работа является продолжением [1]. Напомним определения. Рассмотрим функции $F : G^m \to H$, где G и H — циклические группы. Для удобства будем считать, что циклические группы — это аддитивные группы колец вычетов. Всюду ниже запись $N \bmod M$ обозначает остаток от деления натурального числа N на натуральное число M, а запись (N, M) — их наибольший общий делитель. *Производная по направлению* $\Delta_a F$, $a \in G^m$, функции F определяется равенствами

$$\Delta_a F(x) = F(x + a) - F(x),$$

где $x \in G^m$. *Степенью нелинейности* функции F (обозначается $\mathrm{dl}\, F$) называется минимальное натуральное число t, такое, что

$$\Delta_{a_1} \dots \Delta_{a_{t+1}} F(x) = 0$$

при всех $a_1, \dots, a_{t+1}, x \in G^m$. Если такого числа t не существует, то полагаем $\mathrm{dl}\, F = \infty$. Если функция F тождественно равна нулю, то для удобства полагаем $\mathrm{dl}\, F = -1$.

В работе [1] показано, что степень нелинейности у всех функций из этого множества конечна в том и только в том случае, когда $|G| = p^n$, $|H| = p^k$ при некотором простом p и натуральных $n \geqslant 1$ и $k \geqslant 1$[1], а также $\mathrm{dl}\, F \leqslant m(p^n + (k-1)(p-1)p^{n-1} - 1)$ и верхняя граница степени нелинейности достигается для функций, в табличном задании которых имеется одно ненулевое значение равное 1.

Пусть D_t — множество функций со степенью нелинейности равной t:

$$D_t = \{F : \mathrm{dl}\, F = t\}, \quad 1 \leqslant t \leqslant t_{\max}.$$

Из работы [1] следует, что $D_t \subseteq \mathrm{Ker}\, \Delta_1^{t+1}$, в частности $D_t = \mathrm{Ker}\, \Delta_1^{t+1} \setminus \mathrm{Ker}\, \Delta_1^t$,

$$\mathrm{Ker}\, \Delta_1^1 \subseteq \mathrm{Ker}\, \Delta_1^2 \subseteq \mathrm{Ker}\, \Delta_1^3 \subseteq \dots \subseteq \mathrm{Ker}\, \Delta_1^{t_{\max}} \subseteq \mathrm{Ker}\, \Delta_1^{t_{\max}+1}. \quad (1)$$

[1]Заметим, что в формулировке [1, теорема 1] допущена неточность: в условии пропущены слова «для любой функции» степень нелинейности конечна. Следует также заметить, что свойства значений биномиальных коэффициентов по примарному модулю, приведённые в [1], могут быть выведены в качестве следствия из обобщённой теоремы Люка [2].

Если F — такая функция, что $\Delta_1^{t_{\max}} F \neq 0$, то функции $\Delta_1^{t_{\max}-t} F \neq 0$ имеют степень нелинейности t. Поэтому классы D_t при всех числах t из интервала $1 \leqslant t \leqslant t_{\max}$ не являются пустыми, и в цепочке (1) все включения являются строгими.

В данной работе рассматриваются подходы к описанию классов D_t на основе разложения Ньютона.

Пусть $n = k \geqslant 2$ и $m \geqslant 1$. Как известно, всякую функцию $F : \mathbb{Z}_{p^n}^m \to \mathbb{Z}_{p^n}$ можно представить в виде

$$F(x_1, \ldots, x_m) = \sum_{i_1,\ldots,i_m=0}^{n-1} h(i_1, \ldots, i_m) \binom{x_1}{i_1} \cdots \binom{x_m}{i_m} \mod p^n, \tag{2}$$

где $0 \leqslant h(i_1, \ldots, i_m) \leqslant p^n - 1$, $0 \leqslant i_1, \ldots, i_m \leqslant p^n - 1$, и $\binom{x}{i} = \dfrac{x(x-1)\ldots(x-i+1)}{i!}$ при $0 < i \leqslant p^n - 1$ и $\binom{x}{0} = 1$.

Подсчитаем степени нелинейности для базисных функций, входящих в это разложение. Пусть

$$F_i(x) = \binom{x}{i} \bmod p^n,$$

$x \in \mathbb{Z}_{p^n}$, $0 \leqslant i \leqslant p^n - 1$.

Для дальнейшего потребуются следующие простые свойства.

Свойство 1. Если $\operatorname{dl} F > \operatorname{dl} F'$, то $\operatorname{dl}(F \pm F') = \operatorname{dl} F$.

Свойство 2. Пусть $m \geqslant 1$ и $F : G^m \to H$, где G и H — циклические группы порядков p^n и p^k соответственно, $n \geqslant 2$ и $k \geqslant 2$. Если $k > s \geqslant 1$ и функция F имеет максимальную степень нелинейности $\operatorname{dl} F = m(p^n + (k-1)(p-1)p^{n-1} - 1)$, то степень нелинейности функции $F' : G^m \to H$ вида $F'(x) = p^s F(x) \bmod p^k$ не превосходит величины

$$\operatorname{dl} F' \leqslant m(p^n + (k-s-1)(p-1)p^{n-1} - 1). \tag{3}$$

Действительно, в силу изоморфизма колец $p^s\mathbb{Z}_{p^k} \cong \mathbb{Z}_{p^{k-s}}$ функцию F' можно рассматривать как функцию $F' : G^m \to H'$, где H' — циклическая группа порядка p^{k-s}. Поэтому степень нелинейности функции F' не превосходит максимальной степени нелинейности для функций такого вида, которая в точности равна указанной величине. (Далее будет показано, что при $k = n$ в (3) выполняется равенство.)

Свойство 3. Функция $F_{p^n-1}(x) = \binom{x}{p^n-1} \bmod p^n$ при $x \in \mathbb{Z}_{p^n}$ принимает значение 1 только при $x = p^n - 1$, а в остальных точках равна нулю. Поэтому она имеет максимальную степень нелинейности $p^n + (n-1)(p-1)p^{n-1} - 1$.

Это утверждение является прямым следствием способа доказательства теоремы 2 из [1]. Еще одним следствием этой теоремы является

Свойство 4. Пусть $m \geqslant 1$ и G и H — циклические группы порядков p^n и p^k соответственно, $1 \leqslant s < k \leqslant n$. Тогда функция $F : G^m \to H$ вида

$$F(x_1, \ldots, x_m) = p^s \binom{x_1}{p^n-1} \cdots \binom{x_m}{p^n-1} \bmod p^k$$

имеет степень нелинейности $m(p^n + (k-s-1)(p-1)p^{n-1} - 1)$.

Действительно, при $x_1 = \cdots = x_m = p^n - 1$ функция F принимает значение p^s, а в остальных точках равна нулю. Если теперь функцию F рассматривать как функцию $F' : G^m \to H'$, где H' — циклическая группа порядка p^{k-s}, то она будет иметь единственное ненулевое значение равное 1. Поэтому она имеет максимальную степень нелинейности.

Лемма 1. Пусть $M \geqslant 2$ — натуральное число и

$$F_i(x) = \binom{x}{i} \bmod M,$$

$x \in \mathbb{Z}_M$, $0 \leqslant i \leqslant M-1$. Тогда при $1 \leqslant i \leqslant M-1$ и $x \in \mathbb{Z}_M$ выполняется тождество

$$\Delta_1 F_i \equiv F_{i-1} - F_i(M) F_{M-1} \,(\mathrm{mod}\, M). \tag{4}$$

Доказательство. В силу целочисленного тождества

$$\Delta_1 \binom{x}{i} \equiv \binom{x}{i-1}, \quad x > i \geqslant 1,$$

при $0 \leqslant x < M-1$ имеем

$$\Delta_1 F_i(x) \equiv F_i((x+1) \bmod M) - F_i(x) \equiv F_i(x+1) - F_i(x) \equiv F_{i-1}(x) \pmod M.$$

Для $x = M-1$ получаем

$$\Delta_1 F_i(M-1) \equiv F_i(0) - F_i(x) \equiv F_i(M) - F_i(M-1) + F_i(0) - F_i(M) \equiv \\ \equiv F_{i-1}(M-1) - F_i(M) \pmod M.$$

Так как функция $F_{M-1}(x)$ принимает значение 1 только при $x = M-1$, а в остальных точках равна нулю, то эти равенства можно переписать в виде (4). ■

Лемма 2. Пусть $n \geqslant 2$, p простое и $1 \leqslant i \leqslant p^n - 1$. Тогда значения производных функции $F_i(x) = \binom{x}{i} \bmod p^n$ при $1 \leqslant x \leqslant p^n - 1$ удовлетворяют равенствам

$$\Delta_1 \binom{x}{i} \equiv \begin{cases} \binom{x}{i-1} \pmod{p^n}, & \text{если } (p^n, i) = 1, \\ \binom{x}{i-1} - \binom{p^n}{i}\binom{x}{p^n-1} \pmod{p^n}, & \text{если } (p^n, i) \neq 1. \end{cases}$$

Доказательство вытекает из того факта, что $\binom{p^n}{i} \bmod p^n \neq 0$ только при $(p^n, i) \neq 1$ (см., например, [1, лемма 1]).

Теорема 1. Пусть $n \geqslant 1$ и p простое. Тогда степень нелинейности функции $F_i(x) = \binom{x}{i} \bmod p^n$, $1 \leqslant i \leqslant p^n - 1$, равна

$$\mathrm{dl}\, F_i = \begin{cases} i + (t-1)(p-1)p^{n-1} + p^n - p^t, & \text{если } p^t \leqslant i \leqslant p^{t+1} - 1,\ 1 \leqslant t \leqslant n-1, \\ i, & \text{если } 1 \leqslant i \leqslant p-1. \end{cases}$$

Доказательство. Определим рекурсивно семейство деревьев $D_p(n,k)$, $k = 1,2,\ldots$ При $k=1$ дерево $D_p(n,1)$ представляет собой цепь с вершинами, помеченными последовательно числами $0,1,\ldots,p^n-1$, и корнем в вершине с пометкой p^n-1. Если деревья при $k=1,2,\ldots,t-1$ уже построены, то дерево $D_p(n,t)$ получается из дерева $D_p(n,1)$ путём присоединения к каждой вершине с пометкой вида $p^s a$, $(a,p)=1$, при $1 \leqslant s \leqslant t-1$ ребра, соединяющего эту вершину с корнем дерева $D_p(n,s)$ (см. рис. 1).

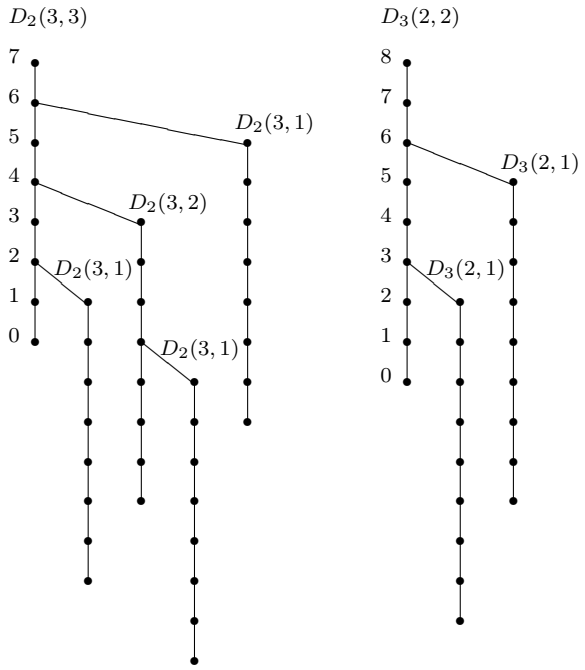

Рис. 1. Деревья $D_2(3,3)$ и $D_3(2,2)$

В терминах свойств этого дерева удобно интерпретировать значения степеней нелинейности. Поставим в соответствие i-й вершине исходного дерева $D_p(n,1)$ биномиальный коэффициент $\binom{x}{i}$. Тогда переходу от вершины i к $i-1$ в дереве $D_p(n,k)$ соответствует взятие производной Δ_1, причём точкам присоединения деревьев соответствует появление дополнительных слагаемых, возникающих в соответствии с формулой

$$\Delta_1\binom{x}{i} \equiv \binom{x}{i-1} - \binom{p^n}{i}\binom{x}{p^n-1} \pmod{p^k}.$$

Нетрудно видеть, что степень нелинейности функции $\binom{x}{i} \bmod p^k$ при $x \in \mathbb{Z}_n$ равна высоте поддерева с корнем в вершине i, т. е. длине максимального пути от i-й вершины до листьев дерева $D_p(n,k)$.

Рассмотрим теперь дерево $D_p(n,n)$ и вычислим значения $\operatorname{dl} F_i$, $1 \leqslant i \leqslant p^n-1$. Воспользуемся индукцией по t. При $t=0$ все значения i из интервала $0 \leqslant i \leqslant p-1$ взаимно просты с p, $(i,p)=1$. По лемме 2 получаем $\operatorname{dl} F_{i-1} = \operatorname{dl} F_i - 1$. Так как при $i=1$ имеем $\operatorname{dl} F_1 = 1$, то для значений i в этом интервале выполнено равенство $\operatorname{dl} F_i = i$.

Предположим, что для значений t, меньших s, равенство выполняется. Докажем его для $t = s$. В силу лемм 5 и 6 из [1] достаточно рассмотреть случай, когда используется только производная Δ_1, так как остальные производные могут быть выражены через неё. При $i = p^s$ имеем

$$\Delta_1 F_{p^s}(x) \equiv \Delta_1 \binom{x}{p^s} \equiv \binom{x}{p^s - 1} - \binom{p^n}{p^s}\binom{x}{p^n - 1} \pmod{p^n}.$$

В силу леммы 1 из [1] и того факта, что умножение на обратимый элемент не изменяет значения степени нелинейности функции, получаем

$$\operatorname{dl}\left[\binom{p^n}{p^s}\binom{x}{p^n - 1}\right] = \operatorname{dl}\left[p^{n-s}\binom{x}{p^n - 1}\right].$$

Это значение совпадает с высотой дерева $D_p(n, s)$ и в силу свойства 4 равно

$$\operatorname{dl}\left[p^{n-s}\binom{x}{p^n - 1}\right] = p^n + (s-1)(p-1)p^{n-1} - 1.$$

По предположению индукции

$$\operatorname{dl}\binom{x}{p^s - 1} = p^s - 1 + (s-2)(p-1)p^{n-1} + p^n - p^{s-1}.$$

Значит, по свойству 1 имеем $\operatorname{dl} F_{p^s} = p^n + (s-1)(p-1)p^{n-1}$.

Рассмотрим теперь случай $p^s < i \leqslant p^{s+1} - 1$. В этом интервале встречаются только числа вида $i = p^r a$, $(p, a) = 1$, при $r \leqslant s$. В графе $D_p(n, n)$ точке $i = p^r a$ при $r \geqslant 1$ соответствует присоединение к дереву $D_p(n, 1)$ дерева $D_p(n, r)$. Так как все такие вершины находятся выше вершины с пометкой p^s, а высота дерева $D_p(n, r)$ не превосходит высоты дерева $D_p(n, s)$, то присоединение этих деревьев не изменит длину максимального пути от i-й вершины до листьев дерева $D_p(n, n)$, который будет проходить через вершину с пометкой p^s. Значит, $\operatorname{dl} F_i = \operatorname{dl} F_{p^s} + i - p^s = p^n + (s-1)(p-1)p^{n-1} + i - p^s$. ■

Следствие 1. Пусть $n \geqslant 1$, p простое. Тогда для степеней нелинейности функций $F_i(x) = \binom{x}{i} \bmod p^n$, $1 \leqslant i \leqslant p^n - 1$, выполняются равенства

$$\operatorname{dl} F_i - \operatorname{dl} F_{i-1} = \begin{cases} (p-1)p^{n-1} + p^t - p^{t+1}, & \text{если } i = p^t, \quad 1 \leqslant t \leqslant n-1, \\ 1, & \text{если } i \neq p^t, \quad 1 \leqslant t \leqslant n-1. \end{cases}$$

Следствие 2. Пусть $n \geqslant 1$, $p \geqslant 2$ и разложение функции $F : \mathbb{Z}_{p^n} \to \mathbb{Z}_{p^n}$ имеет вид

$$F(x) = \sum_{i=0}^{p^n - 1} a_i \binom{x}{i} \bmod p^n.$$

Тогда следующие условия эквивалентны:

1) функция F имеет максимальную степень нелинейности;

2) коэффициент a_{p^n-1} обратим в кольце $\mathbb{Z}_{p^n}$, т. е. $(a_{p^n-1}, p) = 1$;

3) сумма значений функции F является обратимым элементом в кольце $\mathbb{Z}_{p^n}$:

$$\left(\sum_{x=0}^{p^n-1} F(x) \bmod p^n,\ p\right) = 1.$$

Доказательство. Эквивалентность первого и второго условий вытекает из теоремы 1 и свойства 1. Для доказательства эквивалентности второго и третьего условий воспользуемся следующим свойством суммы биномиальных коэффициентов:

$$\sum_{i=0}^{x}\binom{i}{y}=\binom{x+1}{y+1},$$

справедливым при всех $x\geqslant y\geqslant 0$. Его доказательство легко проводится индукцией по x:

$$\sum_{i=0}^{x}\binom{i}{y}=\sum_{i=0}^{x-1}\binom{i}{y}+\binom{x}{y}=\binom{x}{y+1}+\binom{x}{y}=\binom{x+1}{y+1}.$$

С помощью этого равенства получаем

$$\sum_{x=0}^{p^n-1}\binom{x}{i}=\binom{p^n}{i+1}\equiv\begin{cases}0 \pmod{p^n}, & 0\leqslant i\leqslant p^n-2,\\ 1 \pmod{p^n}, & i=p^n-1,\end{cases}$$

откуда $\sum\limits_{x=0}^{p^n-1}F(x)\equiv a_{p^n-1} \pmod{p^n}$. ■

Следствие 3. Пусть $m\geqslant 2$, $n\geqslant 1$, $p\geqslant 2$ и разложение функции $F:\mathbb{Z}_{p^n}^m\to\mathbb{Z}_{p^n}$ имеет вид (2). Тогда следующие условия эквивалентны:

1) функция F имеет максимальную степень нелинейности равную

$$\operatorname{dl}F=m(p^n+(k-1)(p-1)p^{n-1}-1);$$

2) коэффициент $h(p^n-1,\dots,p^n-1)$ обратим в кольце $\mathbb{Z}_{p^n}$, т. е.

$$(h(p^n-1,\dots,p^n-1),p)=1;$$

3) сумма значений функции F является обратимым элементом в кольце $\mathbb{Z}_{p^n}$:

$$\left(\sum_{x_1,\dots,x_m}F(x_1,\dots,x_m)\bmod p^n,\,p\right)=1.$$

Доказательство проводится полностью аналогично.

Следствие 4. Пусть $m\geqslant 1$ и $F:\mathbb{Z}_{p^n}^m\to\mathbb{Z}_{p^n}$, где $n\geqslant 2$. Если $n>s\geqslant 1$ и функция F имеет максимальную степень нелинейности $m(p^n+(n-1)(p-1)p^{n-1}-1)$, то степень нелинейности функции $F':G^m\to H$ вида $F'(x)=p^sF(x) \bmod p^n$ равна

$$\operatorname{dl}F'=m(p^n+(n-s-1)(p-1)p^{n-1}-1).$$

Доказательство. Рассмотрим разложение Ньютона функции F вида (2). В силу следствия 3 и свойства 1 можно считать, что функция F имеет в своём задании единственное ненулевое значение, причём оно должно быть взаимно просто с p. Пусть, например, $F(0)=1$ (в противном случае можно перейти к функции $F''=$ $=F(0)^{-1}F \bmod p^n$). Значит, у функции $F'(x)$ также одно ненулевое значение $F(0)=p^s$. При гомоморфизме $\mathbb{Z}_{p^n}\to\mathbb{Z}_{p^{n-s}}$ элементу p^s соответствует элемент $1\in\mathbb{Z}_{p^{n-s}}$. Поэтому для степени нелинейности функции $F'(x)$ в неравенстве (3) на самом деле должно стоять равенство. ■

Следствие 5. При $n \geqslant 1$, $p \geqslant 2$ число функций максимальной степени нелинейности среди функций вида $F : \mathbb{Z}_{p^n} \to \mathbb{Z}_{p^n}$ равно

$$|D_{t_{\max}}| = \varphi(p^n)p^{n(p^n-1)} = (p-1)p^{np^n-1} = \left(1 - \frac{1}{p}\right) p^{np^n}.$$

Следствие 6. При $m \geqslant 1$, $n \geqslant 1$, $p \geqslant 2$ число функций максимальной степени нелинейности среди функций вида $F : \mathbb{Z}_{p^n}^m \to \mathbb{Z}_{p^n}$ равно

$$|D_{t_{\max}}| = \varphi(p^n)p^{n(p^{mn}-1)} = (p-1)p^{np^{mn}-1} = \left(1 - \frac{1}{p}\right) p^{np^{mn}}.$$

Следствие 7. Пусть $n \geqslant 2$, $p \geqslant 2$. Тогда число функций степени нелинейности i при $0 \leqslant i \leqslant p-1$ среди функций вида $F : \mathbb{Z}_{p^n} \to \mathbb{Z}_{p^n}$ равно

$$|D_i| = (p^n - 1)p^{ni}.$$

Следствие 8. Пусть $n \geqslant 2$, $p \geqslant 2$. Тогда число функций степени нелинейности $t_{\max} - i$ при $1 \leqslant i \leqslant p-1$ среди функций вида $F : \mathbb{Z}_{p^n} \to \mathbb{Z}_{p^n}$ равно

$$|D_{t_{\max}-i}| = (p-1)p^{np^n-i-1}.$$

Доказательство. В силу следствия 3 при $i = 1$

$$\mathrm{dl}\left[p\binom{x}{p^n-1}\right] = p^n + (n-2)(p-1)p^{n-1} - 1.$$

Поскольку при $n \geqslant 2$ выполняется неравенство

$$\mathrm{dl}\left[p\binom{x}{p^n-1}\right] \leqslant p^n + (n-2)(p-1)p^{n-1} - 1 < p^n + (n-1)(p-1)p^{n-1} - 2 = \mathrm{dl}\binom{x}{p^n-2},$$

то в этот класс попадают только функции, у которых в разложении Ньютона коэффициент при $\binom{x}{p^n-2}$ взаимно прост с p, а коэффициент при $\binom{x}{p^n-1}$ кратен p. Число таких функций равно

$$(p-1)p^{2(n-1)}p^{n(p^n-2)} = (p-1)p^{np^n-2}.$$

Далее для $2 \leqslant i \leqslant p-1$ можно воспользоваться индукцией. ■

Пример 1. Пусть $G = H = \mathbb{Z}_4$. Подсчитаем число функций $F : G \to H$ для каждого возможного значения степени нелинейности. Всего имеется 256 таких функций. Рассмотрим разложение Ньютона функции F

$$F(x) = a_0 + a_1\binom{x}{1} + a_2\binom{x}{2} + a_3\binom{x}{3} \bmod 4, \quad a_i \in \mathbb{Z}_4, \quad 0 \leqslant i \leqslant 3.$$

С помощью теоремы 1 и следствия 3 подсчитаем степени нелинейности функций $2^i\binom{x}{j} \bmod 2^2$, $0 \leqslant i \leqslant 1$, $2 \leqslant j \leqslant 3$ (табл. 1):

Таблица 1

i	$\binom{x}{2}$	$\binom{x}{3}$
0	4	5
1	2	3

Пользуясь табл. 1, найдём возможные значения коэффициентов разложения Ньютона для каждого значения степени нелинейности (табл. 2):

Таблица 2

$\deg F$	a_0	a_1	a_2	a_3	Число F
5	*	*	*	1,3	128
4	*	*	1,3	2	64
3	*	*	0,2	2	32
2	*	*	2	0	16
1	*	1–3	0	0	12
0	1–3	0	0	0	3
-1	0	0	0	0	1

Примечание. Символ «*» означает, что соответствующий коэффициент может принимать любое значение из $\mathbb{Z}_4$.

Пример 2. Для $G = H = \mathbb{Z}_9$ уже будет 387 420 489 функций. Рассмотрим разложение Ньютона функции F:

$$F(x) = a_0 + a_1\binom{x}{1} + a_2\binom{x}{2} + \ldots a_8\binom{x}{8} \bmod 9, \quad a_i \in \mathbb{Z}_9.$$

С помощью теоремы 1 и следствия 3 находим степени нелинейности функций $3^i\binom{x}{j} \bmod 3^2$, $0 \leqslant i \leqslant 1$, $2 \leqslant j \leqslant 8$ (табл. 3, см. также рис. 1):

Таблица 3

i	$\binom{x}{2}$	$\binom{x}{3}$	$\binom{x}{4}$	$\binom{x}{5}$	$\binom{x}{6}$	$\binom{x}{7}$	$\binom{x}{8}$
0	2	9	10	11	12	13	14
1	2	3	4	5	6	7	8

Теперь выпишем возможные значения коэффициентов разложения Ньютона для каждого значения степени нелинейности (табл. 4):

Таблица 4

$\mathrm{dl}\, F$	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	Число F
14	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$\neg\, 0{,}3{,}6$	258 280 326
13	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$\neg\, 0{,}3{,}6$	0,3,6	86 093 442
12	$*$	$*$	$*$	$*$	$*$	$*$	$\neg\, 0{,}3{,}6$	0,3,6	0,3,6	28 697 814
11	$*$	$*$	$*$	$*$	$*$	$\neg\, 0{,}3{,}6$	0,3,6	0,3,6	0,3,6	9 565 938
10	$*$	$*$	$*$	$*$	$\neg\, 0{,}3{,}6$	0,3,6	0,3,6	0,3,6	0,3,6	3 188 646
9	$*$	$*$	$*$	$\neg\, 0{,}3{,}6$	0,3,6	0,3,6	0,3,6	0,3,6	0,3,6	1 062 882
8	$*$	$*$	$*$	0,3,6	0,3,6	0,3,6	0,3,6	0,3,6	3, 6	354 294
7	$*$	$*$	0,3,6	0,3,6	0,3,6	0,3,6	0,3,6	3, 6	0	118 098
6	$*$	$*$	0,3,6	0,3,6	0,3,6	0,3,6	3, 6	0	0	39 366
5	$*$	$*$	0,3,6	0,3,6	0,3,6	3, 6	0	0	0	13 122
4	$*$	$*$	0,3,6	0,3,6	3, 6	0	0	0	0	4 374
3	$*$	$*$	$*$	3, 6	0	0	0	0	0	1 458
2	$*$	$*$	1–8	0	0	0	0	0	0	648
1	$*$	1–8	0	0	0	0	0	0	0	72
0	1–8	0	0	0	0	0	0	0	0	8
-1	0	0	0	0	0	0	0	0	0	1

Примечание. Обозначение «$\neg\, 0{,}3{,}6$» использовано для множества $\{1, 2, 4, 5, 7, 8\}$.

Данный пример можно обобщить следующим образом.

Теорема 2. Пусть $p \geqslant 2$. Тогда число функций степени нелинейности i среди функций вида $F : G \to H$, $G = H = \mathbb{Z}_{p^2}$, равно

$$|D_i| = \begin{cases} 1, & \text{если } i = -1, \\ p^{2i}(p^2 - 1), & \text{если } 0 \leqslant i \leqslant p - 1, \\ p^{i+p}(p - 1), & \text{если } p \leqslant i \leqslant p^2 + (p - 1)p - 1. \end{cases}$$

Доказательство вытекает из табл. 5 со значениями степеней нелинейности функций $p^i \binom{x}{j} \bmod p^2$, $0 \leqslant i \leqslant 1$, $2 \leqslant j \leqslant p^2 - 1$.

Таблица 5

i	$\binom{x}{2}$	$\ldots$	$\binom{x}{p-1}$	$\binom{x}{p}$	$\binom{x}{p+1}$	$\ldots$	$\binom{x}{p^2-1}$
0	2	$\ldots$	$p-1$	p^2	p^2+1	$\ldots$	$p^2+(p-1)p-1$
1	2	$\ldots$	$p-1$	p	$p+1$	$\ldots$	p^2-1

Пример 3. Для $G = H = \mathbb{Z}_8$ имеется 16 777 216 функций. Рассмотрим разложение Ньютона функции F:

$$F(x) = a_0 + a_1\binom{x}{1} + a_2\binom{x}{2} + \cdots + a_7\binom{x}{7} \bmod 8, \quad a_i \in \mathbb{Z}_8.$$

Выпишем степени нелинейности функций $2^i \binom{x}{j} \bmod 2^3$, $0 \leqslant i \leqslant 2$, $2 \leqslant j \leqslant 7$ (табл. 6, см. также рис. 1):

Таблица 6

i	$\binom{x}{2}$	$\binom{x}{3}$	$\binom{x}{4}$	$\binom{x}{5}$	$\binom{x}{6}$	$\binom{x}{7}$
2^0	8	9	12	13	14	15
2^1	2	3	8	9	10	11
2^2	2	3	4	5	6	7

Отсюда вытекают возможные значения коэффициентов разложения Ньютона для каждого значения степени нелинейности (табл. 7):

Таблица 7

$\deg F$	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	Число F
15	$*$	$*$	$*$	$*$	$*$	$*$	$*$	1,3,5,7	8 388 608
14	$*$	$*$	$*$	$*$	$*$	$*$	1,3,5,7	0,2,4,6	4 194 304
13	$*$	$*$	$*$	$*$	$*$	1,3,5,7	0,2,4,6	0,2,4,6	2 097 152
12	$*$	$*$	$*$	$*$	1,3,5,7	0,2,4,6	0,2,4,6	0,2,4,6	1 048 576
11	$*$	$*$	$*$	$*$	0,2,4,6	0,2,4,6	0,2,4,6	2, 6	524 288
10	$*$	$*$	$*$	$*$	0,2,4,6	0,2,4,6	2, 6	0, 4	262 144
9	$*$	$*$	$*$	$*$	0,2,4,6	2, 6	0, 4	0, 4	131 072
9	$*$	$*$	$*$	1,3,5,7	0,2,4,6	0, 4	0, 4	0, 4	65 536
8	$*$	$*$	$*$	0,2,4,6	2, 6	0, 4	0, 4	0, 4	32 768
8	$*$	$*$	1,3,5,7	0,2,4,6	0, 4	0, 4	0, 4	0, 4	16 384
7	$*$	$*$	0,2,4,6	0,2,4,6	0, 4	0, 4	0, 4	4	8 192
6	$*$	$*$	0,2,4,6	0,2,4,6	0, 4	0, 4	4	0	4 096
5	$*$	$*$	0,2,4,6	0,2,4,6	0, 4	4	0	0	2 048
4	$*$	$*$	0,2,4,6	0,2,4,6	4	0	0	0	1024
3	$*$	$*$	0,2,4,6	2, 4, 6	0	0	0	0	768
2	$*$	$*$	2, 4, 6	0	0	0	0	0	192
1	$*$	1–7	0	0	0	0	0	0	56
0	1–7	0	0	0	0	0	0	0	7
-1	0	0	0	0	0	0	0	0	1

Этот пример также можно обобщить в виде следующей теоремы.

Теорема 3. Пусть $p \geqslant 2$. Тогда число функций степени нелинейности i среди функций вида $F : G \to H$, $G = H = \mathbb{Z}_{p^3}$, равно

$$|D_i| = \begin{cases} 1, & \text{если } i = -1, \\ p^{3i}(p^3-1), & \text{если } 0 \leqslant i \leqslant p-1, \\ p^{2i+p}(p^2-1), & \text{если } p \leqslant i \leqslant p^2-1, \\ p^{i+p^2+p}(p-1), & \text{если } p^2 \leqslant i \leqslant p^3-1, \\ p^{2i-p^3+p^2+p}(p^2-1), & \text{если } p^3 \leqslant i \leqslant p^3+p^2-p-1, \\ p^{i+2p^2}(p-1), & \text{если } p^3+p^2-p \leqslant i \leqslant p^3+2(p-1)p^2-1. \end{cases}$$

Доказательство вытекает из табл. 8 со значениями степеней нелинейности функций $p^i\binom{x}{j} \bmod p^3$, $0 \leqslant i \leqslant 2$, $2 \leqslant j \leqslant p^3-1$:

Таблица 8

i	$\binom{x}{2}$	$\dots$	$\binom{x}{p-1}$	$\binom{x}{p}$	$\dots$	$\binom{x}{p^2-1}$	$\binom{x}{p^2}$	$\dots$	$\binom{x}{p^3-1}$
0	2	$\dots$	$p-1$	p^3	$\dots$	p^3+p^2-p-1	$p^2+2(p-1)p^2$	$\dots$	$p^3+2(p-1)p^2-1$
1	2	$\dots$	$p-1$	p	$\dots$	p^2-1	p^3	$\dots$	$p^3+(p-1)p^2-1$
2	2	$\dots$	$p-1$	p	$\dots$	p^2-1	p^2	$\dots$	p^3-1

Теперь нетрудно выписать возможные значения коэффициентов разложения Ньютона для каждого значения степени нелинейности (табл. 9):

Таблица 9

$\deg F$	$a_0 \dots a_{p-1}$	a_p	$\dots$	a_{p^2-1}	a_{p^2}	$\dots$	a_{2p^2-p-1}	a_{2p^2-p}	$\dots$	a_{p^3-1}	Число F
$p^3+2(p-1)p^2-1$	$*\cdots*$	$*$	$\dots$	$*$	$*$	$\dots$	$*$	$*$	$\dots$	n	$p^{3p^3-1}(p-1)$
$p^3+2(p-1)p^2-2$	$*\cdots*$	$*$	$\dots$	$*$	$*$	$\dots$	$*$	$*$	$\dots$	p	$p^{3p^3-2}(p-1)$
$\dots$	$\dots$		$\dots$			$\dots$			$\dots$		$\dots$
p^3+p^2-p	$*\cdots*$	$*$	$\dots$	$*$	$*$	$\dots$	$*$	n	$\dots$	p	$p^{p^3+3p^2-p}(p-1)$
p^3+p^2-p-1	$*\cdots*$	$*$	$\dots$	$*$	p	$\dots$	$!p$	p^2	$\dots$	p^2	$p^{p^3+3p^2-p-2}(p-1)$
p^3+p^2-p-1	$*\cdots*$	$*$	$\dots$	n	p	$\dots$	p^2	p^2	$\dots$	p^2	$p^{p^3+3p^2-p-3}(p-1)$
$\dots$	$\dots$		$\dots$			$\dots$			$\dots$		$\dots$
p^3	$*\cdots*$	$*$	$\dots$	p	$!p$	$\dots$	p^2	p^2	$\dots$	p^2	$p^{p^3+p^2+p+1}(p-1)$
p^3	$*\cdots*$	n	$\dots$	p	p^2	$\dots$	p^2	p^2	$\dots$	p^2	$p^{p^3+p^2+p}(p-1)$
p^3-1	$*\cdots*$	$*$	$\dots$	p	p	$\dots$	p	p^2	$\dots$	$!p^2$	$p^{p^3+p^2+p-1}(p-1)$
p^3-2	$*\cdots*$	$*$	$\dots$	p	p	$\dots$	p	p^2	$\dots$	0	$p^{p^3-2p^2-p-2}(p-1)$
$\dots$	$\dots$		$\dots$			$\dots$			$\dots$		$\dots$
$2p^2-p$	$*\cdots*$	p	$\dots$	p	p	$\dots$	p	$!p^2$	$\dots$	0	$p^{4p^2-2p}(p-1)$
$2p^2-p-1$	$*\cdots*$	p	$\dots$	p	p	$\dots$	$!p^2$	0	$\dots$	0	$p^{4p^2-2p-1}(p-1)$
$\dots$	$\dots$		$\dots$			$\dots$			$\dots$		$\dots$
p^2+1	$*\cdots*$	p	$\dots$	p	p	$\dots$	0	0	$\dots$	0	$p^{2p^2+p+1}(p-1)$
p^2	$*\cdots*$	p	$\dots$	p	$!p^2$	$\dots$	0	0	$\dots$	0	$p^{2p^2+p}(p-1)$
p^2-1	$*\cdots*$	p	$\dots$	$p\setminus 0$	0	$\dots$	0	0	$\dots$	0	$p^{2p^2+p-2}(p^2-1)$
p^2-2	$*\cdots*$	p	$\dots$	0	0	$\dots$	0	0	$\dots$	0	$p^{2p^2+p-4}(p^2-1)$
$\dots$	$\dots$		$\dots$			$\dots$			$\dots$		$\dots$
$p+1$	$*\cdots*$	p	$\dots$	0	0	$\dots$	0	0	$\dots$	0	$p^{3p+2}(p^2-1)$
p	$*\cdots*$	$p\setminus 0$	$\dots$	0	0	$\dots$	0	0	$\dots$	0	$p^{3p}(p^2-1)$
$p-1$	$*\cdots\neq 0$	0	$\dots$	0	0	$\dots$	0	0	$\dots$	0	$p^{3(p-1)}(p^3-1)$
$p-2$	$*\dots 0$	0	$\dots$	0	0	$\dots$	0	0	$\dots$	0	$p^{3(p-2)}(p^3-1)$
$\dots$	$\dots$		$\dots$			$\dots$			$\dots$		$\dots$
0	$\neq 0\dots 0$	0	$\dots$	0	0	$\dots$	0	0	$\dots$	0	p^3-1
-1	$0\dots 0$	0	$\dots$	0	0	$\dots$	0	0	$\dots$	0	1

Примечание. $*$ — произвольное значение; p — кратные p; p^2 — кратные p^2; $!p$ — кратные p, но не p^2, без нуля; $!p^2$ — кратные p^2 без нуля; $p\setminus 0$ — кратные p без нуля; $\neq 0$ — произвольное значение, не равное нулю.

ЛИТЕРАТУРА

1. *Черемушкин А. В.* Аддитивный подход к определению степени нелинейности дискретной функции на циклической группе примарного порядка // Прикладная дискретная математика. 2013. № 2(20). С. 26–38.
2. *Granville A.* Arithmetic properties of binomial coefficients. I. Binomial coefficients modulo prime powers // Organic Math. (Burnaby, BC, 1995), CMS Conf. Proc., 20, Amer. Math. Soc., Providence, RI, 1997. P. 253–276.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/24/5

УДК 004.94

АНАЛИЗ УСЛОВИЙ ПРЕДОСТАВЛЕНИЯ И ПОЛУЧЕНИЯ ПРАВ ДОСТУПА В МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ MS SQL SERVER

В. Ю. Смольянинов

Учебно-методическое объединение по информационной безопасности, г. Москва, Россия

E-mail: VladimirSmall@gmail.com

Рассматривается модель управления доступом MS SQL Server, построенная на основе СУБД ДП-модели. Для учёта особенностей управления доступом в СУБД MS SQL Server 2012 в модель добавлены роли, права доступа к учётным записям пользователей и ролям, цепочки владения, а также возможность олицетворения пользователей и активации триггеров и процедур от имени заданных учётных записей пользователей. Доказывается утверждение об эквивалентности возможности выполнения произвольного SQL-кода от имени заданной учётной записи и возможности получения к ней права олицетворения. Обосновываются необходимые и достаточные условия получения и предоставления прав доступа к сущностям при отсутствии кооперации между субъект-сессиями.

Ключевые слова: *компьютерная безопасность, модель управления доступом MS SQL Server, система управления базами данных.*

Введение

Современные реляционные системы управления базами данных (СУБД) зарекомендовали себя как надёжное и эффективное решение задач поиска, хранения и обработки информации. Разработчики автоматизированных информационных систем используют СУБД не только для решения этих задач, но и для защиты информации и, в частности, для управления доступом к обрабатываемым данным. Существующие формальные модели логического управления доступом [1, 2] ориентированы в основном на применение в ОС и не учитывают некоторых существенных особенностей функционирования автоматизированных информационных систем, построенных с использованием реляционных СУБД, которые могут быть использованы нарушителем для реализации запрещённых доступов и информационных потоков. Таким образом, теоретический анализ условий, при которых возможно несанкционированное распространение прав доступа в автоматизированных информационных системах, использующих штатные механизмы управления доступом СУБД, является важной и актуальной задачей.

В [3] предпринята попытка построения СУБД ДП-модели, в которой учитывается наследование прав доступа к дочерним сущностям, существование прав на предоставление прав доступа, а также возможность автоматического выполнения SQL-кода при реализации доступа к данным и изменения учётной записи пользователя при активации хранимых процедур и триггеров. Тем не менее данная модель не учитывает

в полной мере некоторых существенных особенностей организации управления доступом в новой версии СУБД MS SQL Server 2012. В частности, игнорируются роли, наличие прав доступа к учётным записям пользователей и ролям, цепочки владения, возможность олицетворения и выполнения триггеров и процедур от имени заданных учётных записей.

В настоящей работе на базе СУБД ДП-модели [3] построена модель управления доступом MS SQL Server, в которой учитываются перечисленные особенности управления доступом и проведено необходимое уточнение правил преобразования состояний. Описываются элементы модели, предназначенной для анализа безопасности управления доступом в СУБД MS SQL Server. Построенная модель может быть в дальнейшем адаптирована для анализа безопасности других реляционных СУБД.

1. Определения и обозначения

В предлагаемой модели управления доступом *MS SQL Server* структура множества сущностей задаётся следующим образом:

- C — множество контейнеров;
- O_t — множество триггеров;
- O_p — множество процедур;
- U — множество учётных записей пользователей;
- R — множество ролей;
- $P = U \cup R$ — множество принципалов, при этом считается, что $U \cap R = \varnothing$;
- $E = C \cup O_p \cup P$ — множество сущностей, при этом считается, что множества процедур, триггеров, контейнеров и принципалов попарно не пересекаются.

Множество контейнеров реальных реляционных СУБД включает такие сущности, как экземпляр СУБД, экземпляры баз данных, схемы и таблицы. В рамках модели не рассматривается управление доступом к отдельным записям таблиц и поэтому в модель не включено множество записей. Причиной этому служит отсутствие в СУБД MS SQL Server штатных механизмов контроля доступа на уровне отдельных записей. Кроме того, согласно [4], присутствующие в СУБД MS SQL Server 2012 штатные механизмы управления доступом к отдельным полям таблиц не рекомендованы к использованию и, возможно, будут исключены в будущем.

Элементы множества триггеров соответствуют триггерам таблиц в СУБД, которые автоматически срабатывают при осуществлении операций вставки, удаления или обновления записей таблиц, с которыми они связаны. Элементы множества процедур соответствуют хранимым процедурам СУБД, выполнение которых может быть инициировано в рамках взаимодействия с системой.

Считается, что триггеры и процедуры содержат SQL-код, выполнение которого приводит к реализации преобразований системы, задаваемых правилами, описываемыми в п. 2. В реальных системах процедуры могут принимать формальные параметры. При вызове хранимой процедуры ей передаются фактические значения формальных параметров, что приводит к выполнению некоторой фиксированной последовательности операторов. В рамках модели считается, что каждой хранимой процедуре, принимающей входные параметры, соответствует множество процедур. При этом каждой допустимой комбинации значений фактических параметров соответствует отдельная процедура, содержащая SQL-код, являющийся результатом их подстановки в операторы хранимой процедуры. Если внутри хранимой процедуры имеются операторы вызова других хранимых процедур, принимающих параметры, то каждый такой вызов

заменяется оператором активации соответствующей процедуры, полученной подстановкой фактических параметров.

Большинство современных СУБД реализуют, как минимум, дискреционное управление доступом учётных записей пользователей к контейнерам и процедурам. В СУБД MS SQL Server 2012 пользователи представлены на двух уровнях [4]: в виде логинов (`logins`) на уровне экземпляра СУБД и в виде пользователей (`users`) на уровне экземпляра базы данных. Для логинов могут быть заданы права к экземпляру СУБД, а для пользователей — права доступа к сущностям, находящимся в базе данных. Каждому пользователю базы данных соответствует единственный логин. В рамках модели различия между логинами и пользователями игнорируются и считается, что каждый пользователь имеет уникальную учётную запись.

Для повышения гибкости управления правами доступа СУБД MS SQL Server реализует элементы ролевого управления доступом. Роли являются совокупностью прав доступа к сущностям. В СУБД MS SQL Server 2012 роли присутствуют на двух уровнях [4]: на уровне экземпляра СУБД (серверные роли, `server roles`) и на уровне базы данных (роли базы данных, `database roles`). В рамках модели различия между серверными ролями и ролями базы данных игнорируются. В СУБД роли бывают двух видов: фиксированные роли (`fixed roles`) и гибкие роли (`flexible roles`). Отличие фиксированных ролей от гибких состоит в том, что у последних можно управлять не только членством в роли, но и совокупностью предоставленных ей прав доступа к сущностям. Для управления авторизацией на фиксированную роль требуется наличие авторизации на неё у учётной записи инициировавшего операцию пользователя. В рамках модели игнорируется отличие между фиксированными и гибкими ролями.

Для упрощения процедуры администрирования в СУБД поддерживается управление правами доступа с учётом иерархических отношений между сущностями, а также выполняется неявное предоставление всех возможных прав доступа владельцам сущностей. В СУБД обладать правами доступа к сущностям и быть их владельцами могут не только учётные записи пользователей, но и роли. Таким образом, множество принципалов P задаёт участников безопасности, которым могут быть предоставлены права доступа к сущностям. В СУБД права доступа могут быть заданы не только в отношении контейнеров и процедур, но и в отношении самих принципалов, и, следовательно, множество сущностей задаётся как $E = C \cup O_p \cup P$. Заметим, что в рамках модели триггеры не рассматриваются как сущности, так как в СУБД MS SQL Server отсутствуют механизмы управления доступом к ним и, значит, они не являются объектами доступа.

Обозначим $E_H = E \cup O_t$, элементы которого будем называть сущностями (технически это не совсем корректно, так как триггеры не считаются сущностями; $E \cap O_t = \varnothing$). В дальнейшем будем использовать термин «сущность» как в широком смысле (в отношении элементов множества E_H), так и в узком (в отношении элементов множества E); использование термина будет понятно из контекста. Тогда принципал владельца сущности задаётся с помощью функции $owner\colon E_H \to P$. По определению будем считать, что владельцем учётной записи является она сама, то есть для каждого $u \in U$ верно $owner(u) = u$.

Заметим, что в СУБД MS SQL Server триггеры не имеют владельцев. Тем не менее в рамках модели удобно считать, что у любого триггера есть владелец, совпадающий с владельцем таблицы, к которой он относится. У каждой сущности задан единственный владелец, который определяется в момент её создания. Для простоты в модели игнорируется присутствующая в СУБД возможность задания принципала владельца

сущности с использованием SQL-выражения *ALTER AUTHORIZATION ON* (для баз данных, схем, таблиц и процедур), и поэтому выполняется следующее предположение.

Предположение 1. Для любой сущности e учётная запись пользователя владельца $owner(e)$ после создания сущности e не изменяется.

В СУБД MS SQL Server владельцами сущностей могут являться не только учётные записи пользователей, но и роли. Для учёта данной особенности функционирования СУБД областью значений функции *owner* является множество принципалов P.

Функция *owner* введена в модель в связи с тем, что в современных СУБД владелец сущности получает к ней все права доступа. Если же сущность является контейнером, то владелец неявно получает все возможные права доступа не только к самому контейнеру, но и ко всем сущностям, которые в нём содержатся.

Используем также следующие обозначения и определения:

- S — множество субъект-сессий пользователей, при этом считается, что $S \cap E = \varnothing$;
- $user\colon S \to U$ — функция, задающая для каждой субъект-сессии учётную запись пользователя, от имени которой она выполняется;
- U^* — множество всех конечных последовательностей учётных записей пользователей;
- $user\_stack\colon S \to U^*$ — функция, задающая для каждой субъект-сессии последовательность учётных записей пользователей, от имени которых она выполнялась.

Субъект-сессии соответствуют сессиям пользователей с СУБД, которые, в отличие от процессов ОС, не являются объектами доступа и потому не считаются сущностями. Субъект-сессии инициируют выполнение SQL-кода, что приводит к реализации заданных в нём преобразований состояний системы.

Субъект-сессии применяют правила преобразования состояний системы от имени учётных записей пользователей из множества U. В реальных СУБД для создания новой сессии требуется указать в строке подключения данные, аутентифицирующие подключающегося пользователя. Изначально субъект-сессия начинает применять правила от имени учётной записи пользователя, инициировавшей её создание. Однако при переходе системы из состояния в состояние допускается изменение значений функции *user*, что связано с наличием в реальных СУБД возможности выполнения кода хранимых процедур и триггеров от имени учётной записи, не являющейся инициатором их выполнения. Отметим, что возможность изменения значений функции *user* является существенным отличием от моделей ролевого управления доступом и ДП-моделей, в которых предполагается неизменность её значений.

В СУБД пользователь имеет возможность с применением оператора $REVERT$ восстановить учётную запись пользователя, от имени которой субъект-сессия выполнялась до момента предыдущего изменения контекста выполнения. Функция $user\_stack$ добавлена в модель для обеспечения возможности восстановления субъект-сессией s всех предыдущих значений функции $user(s)$.

Функция $cmode\colon C \to \{\texttt{creator}, \texttt{parent}\}$ задаёт для каждого контейнера режим установки владельца вновь создаваемых дочерних сущностей. Пусть для некоторого контейнера $c \in C$ в рамках сессии $s \in S$ было инициировано создание дочерней сущности контейнера c. Тогда, если значение $cmode(c) = \texttt{creator}$, то владельцем вновь созданной дочерней сущности становится учётная запись пользователя $user(s)$. В противном случае, если значение $cmode(c) = \texttt{parent}$, то владельцем вновь созданной дочерней сущности становится принципал $owner(c)$.

Режим установки владельца определяется в момент создания контейнера и не изменяется при выполнении преобразований системы, а значит, верно предположение 2.

Предположение 2. Для любого контейнера $c \in C$ режим установки владельца вновь создаваемых дочерних сущностей $cmode(c)$ после его создания не изменяется.

Необходимость во введении функции *cmode* связана с тем, что в СУБД MS SQL Server владельцем таблиц и процедур является владелец схемы, которой они принадлежат. С другой стороны, владельцем базы данных становится учётная запись создавшего её пользователя.

Для учёта возможности изменения учётной записи пользователя при выполнении триггеров и процедур введена функция $execute\_as\colon O_p \cup O_t \to U \cup \{\texttt{as\_caller}\}$, значение которой задаёт режим выполнения процедур и триггеров, а также определяет учётную запись пользователя, от имени которой субъект-сессии будут выполнять их SQL-код. Отметим, что случай $execute\_as(e) = u$, где $u = owner(e)$ и $u \in U$, соответствует механизму повышения полномочий $SUID$ в ОС семейства UNIX, а случай $execute\_as(e) = \texttt{as\_caller}$ — механизму имперсонации (олицетворения) в ОС семейства Windows. Если некоторая субъект-сессия $s \in S$ инициировала выполнение процедуры или триггера $e \in O_p \cup O_t$ и $execute\_as(e) = \texttt{as\_caller}$, то код сущности e будет выполняться от имени учётной записи пользователя $user(s)$, от имени которой s выполнялась непосредственно до запуска SQL-кода сущности e. В противном случае, если $execute\_as(e) = u$, где $u \in U$, то код сущности e будет выполняться субъект-сессией s от имени учётной записи пользователя u.

Заметим, что в модели не рассматриваются два присутствующих в СУБД режима выполнения триггеров и процедур: от имени владельца (*AS OWNER*) и от имени учётной записи пользователя, последним изменившим их SQL-код (*AS SELF*). Оба этих режима могут рассматриваться как частные случаи выполнения SQL-кода процедур и триггеров от имени заданной учётной записи пользователя. Заметим, что в СУБД режим выполнения от имени владельца может быть задан только для триггеров и процедур, владельцем которых являются учётные записи, а не роли.

Выше уже отмечалось, что СУБД MS SQL Server поддерживает базовые механизмы ролевого управления доступом, для учёта которых в модель добавлен ряд элементов, в частности функция $UA\colon U \to 2^R$ — функция авторизованных ролей учётных записей пользователей, задающая для каждой учётной записи пользователя множество ролей, на которые она может быть авторизована.

В дальнейшем будем считать, что выполняется следующее предположение.

Предположение 3. Для каждой роли $r \in R$ существует учётная запись пользователя u, такая, что $r \in UA(u)$. Таким образом, для каждой роли существуют авторизованные на неё пользователи.

Будем считать, что существует выделенная роль $public \in R$, такая, что для каждого $u \in U$ верно $public \in UA(u)$, то есть на роль *public* авторизованы учётные записи всех пользователей.

В СУБД MS SQL Server учётные записи пользователей нельзя лишить авторизации на роль *public*. Роль *public* считается фиксированной, однако в отличие от других фиксированных ролей существует возможность управлять совокупностью предоставленных ей прав доступа. Именно поэтому в рамках модели данная роль считается гибкой и отнесена с указанными выше ограничениями к множеству ролей R.

Будем считать, что на множестве ролей R задано отношение частичного порядка, которое удовлетворяет следующему определению.

Определение 1. Иерархией ролей называется заданное на множестве ролей R отношение частичного порядка $\leqslant$. При этом по определению выполняется: если для пользователя $u \in U$ роли $r, r' \in R$ такие, что $r \in UA(u)$ и $r' \leqslant r$, то $r' \in UA(u)$.

В случае, когда для двух ролей $r_1, r_2 \in R$ выполняются условия $r_1 \leqslant r_2$ и $r_1 \neq r_2$, будем использовать обозначение $r_1 < r_2$.

Будем считать, что существует выделенная роль $sysadmin \in R$, такая, что для каждой роли $r \in R$ выполняется $r \leqslant sysadmin$.

В СУБД MS SQL Server фиксированная роль $sysadmin$ считается ролью системных администраторов. Пользователи, авторизованные на данную роль, могут выполнять произвольные действия с любой сущностью СУБД.

В рамках модели считается, что субъект-сессия, выполняемая от имени некоторой учётной записи пользователя, авторизована на все её роли. Следовательно, можно считать, что учётная запись пользователя обладает правами доступа всех ролей, на которые она может быть авторизована. Так как на множестве ролей задана иерархия, учётная запись пользователя обладает правами доступа всех ролей, «иерархически подчиненных» роли, на которую она может быть авторизована.

В СУБД MS SQL Server иерархия на множествах гибких и фиксированных ролей задаётся путём предоставления «членства в роли»: считается, что если роль r_2 является членом роли r_1, то $r_1 \leqslant r_2$. Отметим, что авторизация пользователей на роль также осуществляется путём предоставления пользователю «членства в роли». Рассмотрим следующий пример. Пусть роль $Hackers$ является членом роли $Users$ и роли $Users$ предоставлено право доступа на чтение к таблице $Table$, а роли $Hackers$ предоставлено к ней право доступа на запись. Тогда если пользователь $Alice$ является членом группы $Users$, то ей предоставлено право доступа на чтение к таблице $Table$, в то время как члену группы $Hackers$ пользователю Bob предоставлено право доступа к таблице $Table$ как на чтение, так и на запись.

В современных СУБД на множестве контейнеров неявно задаётся иерархия сущностей. Как правило, с экземпляром СУБД связаны экземпляры баз данных, которые, в свою очередь, содержат схемы данных, являющиеся контейнерами для хранимых процедур, а также таблиц вместе с содержащимися в них записями и триггерами. Для задания иерархических отношений сущностей введём следующие определения.

Определение 2. Определим $H\colon C \to 2^{E_H}$ — функцию иерархии сущностей, сопоставляющую каждому контейнеру $c \in C$ множество $H(c) \subset E_H$ и удовлетворяющую следующим условиям.

У с л о в и е 1. Пусть $Pr(e) = \{c \in C : e \in H(c)\}$ — множество родительских контейнеров сущности e. Тогда существует единственный корневой контейнер $c_r \in C$, такой, что $|Pr(c_r)| = 0$, $owner(c_r) = sysadmin$ и для каждой сущности $e \in E_H \setminus \{c_r\}$ выполняется $|Pr(e)| = 1$.

У с л о в и е 2. Для каждой сущности $e \in E_H$ либо $e = c_r$, либо существуют контейнеры $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, такие, что $c_1 = c_r$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$, при этом если $e \in P$, то $n = 1$.

У с л о в и е 3. Пусть $T = \{c \in C : \exists o_t \in O_t\,(o_t \in H(c))\}$ — множество таблиц. Тогда для каждой таблицы $t \in T$ по определению выполняется $H(t) \subseteq O_t$ и для каждого $o_t \in H(t)$ верно $owner(o_t) = owner(t)$.

У с л о в и е 4. Для каждой сущности $e \in O_p \cup T$ выполняется равенство $cmode(Pr(e)) = \texttt{parent}$.

Определение 3. Если для сущности $e \in E_H$ существует контейнер $c \in C$, такой, что $e \in H(c)$, то будем говорить, что контейнер c содержит сущность e, а сущность e содержится в контейнере c; также будем говорить, что c является родительским контейнером сущности e, а e — дочерней сущностью контейнера c.

Будем считать, что единственный корневой контейнер c_r, у которого отсутствуют родительские контейнеры, соответствует экземпляру СУБД. Владельцем корневого контейнера c_r является роль *sysadmin*.

Согласно условию 2 определения 2, родительским контейнером всех принципалов является корневой контейнер c_r. В реальных СУБД роли не считаются частью иерархии сущностей, тем не менее владелец экземпляра СУБД обладает всеми правами доступа к принципалам. Следовательно, удобно считать, что все принципалы являются дочерними сущностями c_r.

Помимо корневого контейнера c_r, в рамках модели выделяется особый вид контейнеров — таблицы. Согласно условию 3 определения 2, для каждой таблицы $t \in T$ выполняется $H(t) \subseteq O_t$, из чего следует, что таблицы не могут иметь дочерних контейнеров и содержат только триггеры. В реальных СУБД триггеры связаны с таблицами, а не содержатся в них, при этом наличие права доступа к таблице на изменение её способа задания (*ALTER*) позволяет также изменять способ задание её триггеров. Следовательно, в рамках модели удобно считать триггеры частью иерархии и дочерними сущностями таблиц.

Выше отмечалось, что в СУБД MS SQL Server владельцем таблиц и процедур является владелец схемы, в которой они содержатся. Для того чтобы модель соответствовала данному ограничению, в определение 2 введено условие 4: режим установки владельца для контейнеров, содержащих таблицы и процедуры, должен быть равен `parent`.

Наконец отметим, что в отличие от реальных СУБД модель не накладывает ограничений на число уровней вложенности в иерархии.

В СУБД принципалы, имеющие права доступа к контейнерам, получают также права ко всем сущностям, расположенным ниже по иерархии. Для учёта этой особенности управления доступом введём определение отношения иерархической подчинённости сущностей.

Определение 4. Будем говорить, что сущность $e \in E_H$ иерархически подчинена контейнеру $c \in C$, если существуют контейнеры $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, такие, что $c_1 = c$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$.

В случае, если сущность $e \in E_H$ иерархически подчинена контейнеру $c \in C$, будем использовать обозначение $e < c$. В случае, если сущность $e \in E_H$ иерархически подчинена контейнеру $c \in C$ или равна ему, будем использовать обозначение $e \leqslant c$.

В зависимости от контекста будет понятно, используется ли «$\leqslant$» для обозначения иерархической подчинённости роли контейнеру или иерархических отношений на множестве ролей.

Замечание 1. Из определений 2 и 4 следует, что для любой сущности $e \in E_H \setminus \{c_r\}$ существует контейнер $c \in C$, такой, что $e < c$. Очевидно, что каждая сущность либо является корневой, либо иерархически подчинена корневой сущности c_r и содержится ровно в одном из её дочерних контейнеров.

Итак, для каждой сущности, за исключением корневой, существует единственный контейнер, в котором она содержится. Кроме того, в соответствии с условием 3 определения 2 в множество таблиц включаются все контейнеры, содержащие хотя бы один

триггер. Следовательно, для каждого триггера $o_t \in O_t$ существует ровно одна таблица $t \in T$, в которой он содержится, то есть $o_t \in H(t)$.

Замечание 2. Легко показать, что если для некоторой сущности $e \in E_H$ существуют контейнеры $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, такие, что $c_1 = c_r$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$, и контейнеры $c'_1, \ldots, c'_{n'} \in C$, где $n' \geqslant 1$, такие, что $c'_1 = c_r$, $c'_{i+1} \in H(c'_i)$ для $i = 1, \ldots, n'-1$ и $e \in H(c'_{n'})$, то $n = n'$ и $c'_i = c_i$ для $i = 1, \ldots, n$. Это означает, что для любой некорневой сущности $e \in E_H$ существует единственная последовательность, начинающаяся с c_r и заканчивающаяся контейнером, содержащим e, такая, что каждый её элемент является родительским для последующего.

Определение 5. Функция $holders\colon E_H \to 2^P$ задаёт для каждой сущности принципалы её иерархических владельцев, при этом для каждой сущности $e \in E_H$ по определению $p \in holders(e)$ тогда и только тогда, когда либо $p = owner(e)$, либо существует контейнер $c \in Pr(e)$, такой, что $p \in holders(c)$.

Принципалы иерархических владельцев сущности обладают к ней всеми возможными правами доступа. Заметим, что значения функции $holders$ однозначно определяются функцией иерархии сущностей H и функцией задания владельцев сущностей $owner$, и поэтому она не рассматривается в качестве самостоятельного элемента состояния системы.

Введём множество видов прав доступа к сущностям R_r, которые соответствуют по смыслу и назначению традиционным для СУБД разрешениям *SELECT*, *UPDATE*, *INSERT*, *DELETE*, *ALTER*, *EXECUTE* и *IMPERSONATE*:

$$R_r = \{select_r, insert_r, update_r, delete_r, alter_r, execute_r, impersonate_r\}.$$

Считается, что наличие у учётной записи пользователя прав доступа $select_r$, $insert_r$, $update_r$ или $delete_r$ к таблице позволяет реализовать к ней соответствующее право доступа (то есть выполнить операции выборки, вставки, обновления и удаления соответственно). Обладание правом доступа $alter_r$ к процедуре позволяет задавать её SQL-код и режим выполнения. Право доступа $alter_r$ к таблице позволяет создавать в ней триггеры, а также изменять SQL-код связанных с ней триггеров и режим их выполнения. Наличие права доступа $execute_r$ к процедуре позволяет выполнять её SQL-код. Наличие права доступа $impersonate_r$ к учётной записи пользователя позволяет инициировать выполнение SQL-кода от её имени. Заметим, что право доступа $impersonate_r$ неприменимо к ролям, и поэтому область значений функции $execute\_as$ задана как $U \cup \{\texttt{as\_caller}\}$, а не как $P \cup \{\texttt{as\_caller}\}$. Кроме того, наличие права доступа $alter_r$ к процедуре $o_p \in O_p$ и $impersonate_r$ к учётной записи $u \in U$ позволяет установить режим выполнения процедур o_p от имени учётной записи пользователя u. Аналогично, для задания режима выполнения триггеров таблицы $t \in T$ от имени учётной записи пользователя $u \in U$ требуется наличие права доступа $alter_r$ к таблице, а также права доступа $impersonate_r$ к учётной записи пользователя u.

В СУБД MS SQL Server существует разрешение *TAKE OWNERSHIP*, позволяющее стать владельцем сущности. По умолчанию, разрешение *TAKE OWNERSHIP* к сущности может быть предоставлено только её владельцем и администраторами СУБД. В рамках модели возможность изменения владельцев сущностей игнорируется.

Введём $R_d \subseteq P \times E \times R_r$ — множество непосредственно заданных прав доступа принципалов к сущностям. Права доступа к сущностям добавляются в множество R_d в результате явного применения правил преобразования системы, а не в результате обладания ролью или правами доступа к контейнеру, которому иерархически подчинена сущность.

Отметим, что способ задания множества R_d учитывает, что большинство современных СУБД предоставляют возможность устанавливать права доступа к контейнерам и процедурам, не позволяя определять их на уровне записей таблиц и триггеров.

Введём $R_{own} = \{(p, e, \alpha_r) : e \in E, p = owner(e), \alpha_r \in R_r\}$ — множество прав доступа принципалов владельцев к сущностям. В соответствии со способом задания R_{own} принципалы владельцев сущностей обладают к ним всеми правами доступа.

Для учёта того, что принципалы, имеющие право доступа к контейнеру, также обладают им ко всем его иерархически подчинённым сущностям, вводится множество иерархических прав доступа принципалов к сущностям:

$$R_H = \{(p, e, \alpha_r) : p \in P, e \in E, \alpha_r \in R_r, \exists c \in C((p, c, \alpha_r) \in R_d \cup R_{own} \,\&\, e < c)\}.$$

Обозначим $PR = R_d \cup R_{own} \cup R_H$. Введём R_e — множество действующих прав доступа с учётом того, что на множестве ролей задана иерархия и каждая субъект-сессия авторизована на все роли, на которые авторизована учётная запись пользователя, от имени которой она выполняется. Заметим, что для удобства множество R_e задано таким образом, чтобы роли имели права доступа всех иерархически подчинённых им ролей:

$$\begin{gathered} R_e = \{(u, e, \alpha_r) : u \in U, e \in E, \alpha_r \in R_r((u, e, \alpha_r) \in PR)\} \cup \\ \cup\{(u, e, \alpha_r) : u \in U, e \in E, \alpha_r \in R_r, \exists r \in R((r, e, \alpha_r) \in PR \,\&\, r \in UA(u))\} \cup \\ \cup\{(r, e, \alpha_r) : r \in R, e \in E, \alpha_r \in R, \exists r' \in R((r', e, \alpha_r) \in PR, r' \leqslant r)\}. \end{gathered}$$

При определении возможности применения правил преобразования состояний учитываются права доступа, содержащиеся в множестве R_e. Полагается, что субъект-сессии s, функционирующей от имени учётной записи пользователя $u = user(s)$, разрешено реализовать доступ вида α_r к сущности e в случае, если $(u, e, \alpha_r) \in R_e$, то есть у учётной записи пользователя u есть действующее право доступа α_r к сущности e (далее в этом случае будем говорить о наличии права доступа).

Отметим, что множество R_e включает права доступа учётных записей пользователей с учётом ролей, на которые они авторизованы.

Очевидно, что из способа задания множеств R_H и R_e следует

Замечание 3. Если $u \in U$, $e, e' \in C \cup O_p$, $\alpha_r \in R_r$, $e \leqslant e'$ и $(u, e', \alpha_r) \in R_e$, то $(u, e, \alpha_r) \in R_e$.

Множества прав доступа R_{own}, R_H и R_e однозначно определяются множеством непосредственно заданных прав доступа R_d, функцией иерархии сущностей H, функцией авторизованных ролей пользователей UA, а также функцией задания владельцев сущностей *owner* и поэтому не рассматриваются в качестве самостоятельных элементов состояния системы.

Как правило, в ОС, реализующих механизмы дискреционного управления доступом, назначать права доступа к сущности может только её владелец и, возможно, пользователь, обладающий привилегиями администратора. Указанные пользователи могут предоставить произвольное право доступа к сущности любому другому пользователю системы. В то же время в современных СУБД присутствуют штатные механизмы, позволяющие делегировать предоставление прав доступа к сущности заданному для неё кругу учётных записей пользователей. При этом СУБД позволяют ограничить для каждой учётной записи из этого круга набор прав доступа к сущности, которые она может предоставлять.

Обозначим:

— $Gr_d \subseteq P \times E \times R_r$ — множество непосредственно заданных прав принципалов на предоставление прав доступа к сущностям, удовлетворяющее условию $Gr_d \subseteq R_d$;

— $Gr_{hld} = \{(p, e, \alpha_r) : \alpha_r \in R_r, e \in E, p \in holders(e)\}$ — множество прав доступа принципалов на предоставление прав доступа иерархическими владельцами сущностей;

— $PGr = Gr_d \cup Gr_{hld}$. Тогда, с учётом того, что учётная запись пользователя обладает правами всех ролей, на которые она может быть авторизована, а также с учётом того, что роль обладает правами всех иерархически подчинённых ей ролей, введём множество действующих прав доступа учётных записей пользователей на предоставление прав доступа к сущностям:

$$Gr_e = \{(u, e, \alpha_r) : u \in U, e \in E, \alpha_r \in R_r((u, e, \alpha_r) \in PGr)\} \cup$$
$$\cup\{(u, e, \alpha_r) : u \in U, e \in E, \alpha_r \in R_r, \exists r \in R((r, e, \alpha_r) \in PGr \,\&\, r \in UA(u))\} \cup$$
$$\cup\{(r, e, \alpha_r) : r \in R, e \in E, \alpha_r \in R, \exists r' \in R((r', e, \alpha_r) \in PGr, r' \leqslant r)\}.$$

Множество Gr_e включает права доступа на предоставление прав доступа учётных записей пользователей, с учётом ролей, на которые они могут быть авторизованы. Заметим, что для удобства множество Gr_e задано таким образом, чтобы роли имели возможность предоставлять права доступа к сущностям, к которым могут предоставлять права их иерархически подчинённые роли.

Множества Gr_{hld} и Gr_e однозначно определяются множеством непосредственно заданных прав доступа учётных записей пользователей на предоставление прав доступа к сущностям Gr_d, функцией иерархии сущностей H, функцией авторизованных ролей пользователей UA, а также функцией задания владельцев сущностей *owner* и поэтому не рассматриваются в качестве самостоятельных элементов состояния системы.

Для удобства введём обозначение $Gr(e, \alpha_r) = \{p \in P : (p, e, \alpha_r) \in Gr_e\}$ — множество принципалов, которые могут предоставлять право доступа $\alpha_r \in R_r$ к сущности $e \in E$. В рамках модели полагается, что субъект-сессии s, функционирующей от имени учётной записи пользователя $u = user(s)$, разрешено предоставить право доступа вида α_r к сущности e, если $u \in Gr(e, \alpha_r)$, то есть учётная запись пользователя u обладает действующим правом доступа на предоставление права доступа α_r к сущности e (далее в этом случае будем говорить о наличии права доступа на предоставление права доступа).

В современных СУБД для того чтобы пользователь имел возможность предоставить право доступа к сущности, требуется, чтобы он сам обладал к ней этим правом доступа. Обоснуем, что в рамках модели необходимым условием наличия у учётной записи пользователя $u \in U$ права на предоставление права доступа α_r к сущности $e \in E$ является наличие у u права доступа α_r к e.

Утверждение 1. $Gr_e \subseteq R_e$.

Доказательство. Заметим, что, согласно способу задания множеств R_e и Gr_e, достаточно показать, что $PGr \subseteq PR$. По способу задания $PGr = Gr_d \cup Gr_{hld}$ и $Gr_d \subseteq R_d \subseteq PR$. Покажем, что $Gr_{hld} \subseteq PR$. Пусть $(p, e, \alpha_r) \in Gr_{hld}$, где $e \in E$, $\alpha_r \in R_r$ и $p \in holders(e)$. В случае $p = owner(e)$, согласно способу задания множества R_{own}, выполняется $(p, e, \alpha_r) \in R_{own} \subseteq PR$. Пусть $p \neq owner(e)$. Согласно условию 3 определения 2, для каждой сущности $e \in E$ либо $e = c_r$, либо существуют контейнеры $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, такие, что $c_1 = c_r$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$. Согласно замечанию 2, такая последовательность единственна. Следовательно, в соответствии со способом задания множество $holders(e)$ состоит из элемен-

тов $owner(c_1), \ldots, owner(c_n), owner(e)$. Так как $p \neq owner(e)$, существует $i \in \{1, \ldots, n\}$, такое, что $p = owner(c_i)$ и $e < c_i$. Так как $p = owner(c_i)$, по способу задания множества R_{own} выполняется $(p, c_i, \alpha_r) \in R_{own}$. Таким образом, выполняется $p \in P$, $e \in E$, $\alpha_r \in R_r$ и существует контейнер $c_i \in C$, такой, что $(p, c_i, \alpha_r) \in R_{own}$ и $e < c_i$. Тогда по способу задания множества R_H выполняется $(p, e, \alpha_r) \in R_H \subseteq PR$. ■

Используем следующие обозначения:

— OP_{int} — множество внутренних правил преобразования состояний, заданных в табл. 1;
— OP_{ext} — множество внешних правил преобразования состояний, заданных в табл. 2;
— $OP = OP_{\text{int}} \cup OP_{\text{ext}}$ — множество правил преобразования состояний;
— OP^*_{ext} — множество всех конечных последовательностей внешних правил преобразования состояний;
— OP^* — множество всех конечных последовательностей правил преобразования состояний;
— $tr \in OP^*$ — последовательность всех применённых правил, из которой исключены правила, условия применения которых не были выполнены;
— $assoc_f(s)\colon S \to O_p \cup O_t \cup \{\varnothing\}$ — функция, задающая для каждой субъект-сессии $s \in S$ функционально ассоциированную с ней сущность. Для сущностей, функционально-ассоциированных с субъект-сессией s, используется обозначение $[s]$;
— $operations\colon O_p \cup O_t \to OP^*_{\text{ext}}$ — функция, задающая для каждой процедуры и триггера соответствующие им последовательности внешних правил преобразования состояний, реализуемых при выполнении субъектами-сессиями их SQL-кода.

Таблица 1

Внутренние правила преобразования состояний

Правило	Исходное состояние G	Результирующее состояние G'
$do\_switch(s, o, u)$	$s \in S, o \in O_p \cup O_t \cup \{\varnothing\}, u \in U$	$G'\{user', assoc'_f, tr'\} \sim G$, $user'\{s \mapsto u\} \sim user$, $assoc'_f\{s \mapsto o\} \sim assoc_f$, $tr' = (tr, do\_switch(s, o, u))$
$execute(s, o)$	$s \in S, o \in O_p \cup O_t, e = [s]$, $operations(o) = (op_1, \ldots, op_k)$, $k \geqslant 0$, $\forall i \in \{1, \ldots, k\}(op_i \in OP_{\text{ext}})$	$G'' = G(do\_switch(s, o, u), op_1, \ldots, op_k, do\_switch(s, e, user(s)))$, где $p = \begin{cases} user(s) \text{ при } execute\_as(o) = \texttt{as\_caller}, \\ \text{иначе } execute\_as(o) \end{cases}$ $G'\{user\_stack'\} \sim G''$, где $user\_stack' = user\_stack$

Таблица 2

Внешние правила преобразования

Правило	Исходное состояние G	Результирующее состояние G'
$create\_session(s, u)$	$u \in U, s \notin S$	$G'\{S', user', user\_stack', assoc'_f, tr'\} \sim G$, $S' = S \cup \{s\}$, $user'\{s \mapsto u\} \sim user$, $user\_stack'\{s \mapsto (u)\} \sim user\_stack$, $assoc'_f\{s \mapsto \varnothing\} \sim assoc_f$, $tr' = (tr, create\_session(s, u))$
$switch(s, u)$	$s \in S$, $u \in U$, $(user(s), u, impersonate_r) \in R_e$	$G'' = G(do\_switch(s, [s], u))$, $G'\{user\_stack', tr'\} \sim G''$, $user\_stack'\{s \mapsto (user\_stack(s), u)\} \sim user\_stack''$, $tr' = (tr'', switch(s, u))$
$revert(s)$	$s \in S$, $user\_stack(s) = (u_1, \ldots, u_{k-1}, u_k)$, $k \geqslant 1$	$G'\{user', user\_stack', tr'\} \sim G$, $user'\{s \mapsto \mapsto u_{\max(1,k-1)}\} \sim user$, $user\_stack'\{s \mapsto (u_1, \ldots, u_{\max(1,k-1)})\} \sim user\_stack$, $tr' = (tr, revert(s))$

Окончание таблицы 2

Правило	Исходное состояние G	Результирующее состояние G'
$grant\_right$ $(s, p, e, \alpha_r,$ $with\_grant)$	$s \in S, p \in P, e \in E, \alpha_r \in R_r,$ $user(s) \in Gr(e, \alpha_r),$ $with\_grant \in \{\texttt{yes}, \texttt{no}\}$	$G'\{R'_d, Gr'_d, tr'\} \sim G$, $R'_d = R_d \cup \{(p, e, \alpha_r)\}$, $Gr'_d = \begin{cases} Gr_d \cup \{(p, e, \alpha_r)\} \text{ при } with\_grant = \texttt{yes}, \\ Gr_d \text{ при } with\_grant = \texttt{no} \end{cases}$ $tr' = (tr, grant\_right(s, p, e, \alpha_r, with\_grant))$
add_member (s, r, u)	$s \in S$, $r \in R$, $u \in U$ $(user(s), r, alter_r) \in R_e$	$G'\{UA', tr'\} \sim G$, $UA'\{u \mapsto UA(u) \cup \{r' \in R : r' \leqslant r\}\} \sim UA$, $tr' = (tr, add\_role\_member(s, r, u))$
$create\_container$ $(s, c_p, c, mode)$	$s \in S$, $c_p \in C$, $c \notin C$, $(user(s), c_p, alter_r) \in R_e$ $mode \in \{\texttt{creator}, \texttt{parent}\}$	$G'\{E'_H, H', owner', cmode', tr'\} \sim G$, $E'_H = C' \cup O_p \cup P \cup O_t$, $C' = C \cup \{c\}$, $H'\{c \mapsto \varnothing, c_p \mapsto H(c_p) \cup \{c\}\} \sim H$, $owner'\{c \mapsto p\} \sim owner$, где $p = \begin{cases} user(s) \text{ при } cmode(c_p) = \texttt{creator}, \\ owner(c_p) \text{ при } cmode(c_p) = \texttt{parent} \end{cases}$ $cmode'\{c \mapsto mode\} \sim cmode$, $tr' = (tr, create\_container(s, c_p, c, mode))$
$create\_procedure$ $(s, c, o_p, md,$ $op_1, \ldots, op_k)$	$s \in S, c \in C \setminus T$, $o_p \notin O_p$, $cmode(c) = \texttt{parent}$, $md \in U \cup \{\texttt{as\_caller}\}$, $k \geqslant 0$, $\forall i \in \{1, \ldots k\}(op_i \in OP_{\text{ext}})$, $(user(s), c, alter_r) \in R_e$; если $md \in U$, то $(user(s),$ $md, impersonate_r) \in R_e$	$G'\{E'_H, H', owner', execute\_as', operations',$ $tr'\} \sim G$, $E'_H = C \cup O'_p \cup P \cup O_t$, $O'_p = O_p \cup \{o_p\}$, $H'\{c \mapsto H(c) \cup \{o_p\}\} \sim H$, $owner'\{o_p \mapsto owner(c)\} \sim owner$, $execute\_as'\{o_p \mapsto md\} \sim execute\_as$, $operations'\{o_p \mapsto (op_1, \ldots, op_k)\} \sim operations$, $tr' = (tr, create\_procedure(s, c, o_p, md, op_1, \ldots, op_k))$
$alter\_procedure$ $(s, o_p, md,$ $op_1, \ldots, op_k)$	$s \in S$, $o_p \in O_p$, $md \in U \cup \{\texttt{as\_caller}\}$, $k \geqslant 0$, $\forall i \in \{1, \ldots k\}(op_i \in OP_{\text{ext}})$, $(user(s), o_p, alter_r) \in R_e$; если $md \in U$, то $(user(s),$ $md, impersonate_r) \in R_e$	$G'\{execute\_as', operations', tr'\} \sim G$, $execute\_as'\{o_p \mapsto m\} \sim execute\_as$, $operations'\{o_p \mapsto (op_1, \ldots, op_k)\} \sim operations$, $tr' = (tr, alter\_procedure(s, o_p, md, op_1, \ldots, op_k))$
$create\_trigger$ $(s, o_t, t, \alpha_r, md,$ $op_1, \ldots, op_k)$	$s \in S$, $t \in T$, $o_t \notin O_t$, $\alpha_r \in \{insert_r, update_r, delete_r\}$, $md \in U \cup \{\texttt{as\_caller}\}$, $k \geqslant 0$, $\forall i \in \{1, \ldots k\}(op_i \in OP_{\text{ext}})$, $(user(s), t, alter_r) \in R_e$; если $md \in U$, то $(user(s),$ $md, impersonate_r) \in R_e$	$G'\{E'_H, H', owner', execute\_as', triggers',$ $operations', tr'\} \sim G$, $E'_H = E_H \cup O'_t$, $O'_t = O_t \cup \{o_t\}$, $H'\{t \mapsto H(t) \cup \{o_t\}\} \sim H$, $owner'\{o_t \mapsto owner(t)\} \sim owner$, $execute\_as'\{o_t \mapsto md\} \sim execute\_as$, $triggers'\{(t, \alpha_r) \mapsto (triggers(t, \alpha_r), o_t)\} \sim$ $\sim triggers$, $operations'\{o_t \mapsto (op_1, \ldots, op_k)\} \sim$ $\sim operations$, $tr' = (tr, create\_trigger(s, o_t, t, \alpha_r,$ $md, op_1, \ldots, op_k))$
$alter\_trigger$ $(s, o_t, t, md,$ $op_1, \ldots, op_k)$	$s \in S, t \in T, o_t \in H(t)$, $md \in U \cup \{\texttt{as\_caller}\}$, $k \geqslant 0$, $\forall i \in \{1, \ldots k\}(op_i \in OP_{\text{ext}})$, $(user(s), t, alter_r) \in R_e$; если $md \in U$, то $(user(s),$ $md, impersonate_r) \in R_e$	$G'\{execute\_as', operations', tr'\} \sim G$, $execute\_as'\{o_t \mapsto md\} \sim execute\_as$, $operations'\{o_t \mapsto (op_1, \ldots, op_k)\} \sim operations$, $tr' = (tr, alter\_trigger(s, t, o_t, md, op_1, \ldots, op_k))$
$execute\_procedure$ (s, o_p)	$s \in S, o_p \in O_p$, $(user(s), o_p, execute_r) \in R_e$ или $[s] \neq \varnothing$ и $owner([s]) =$ $= owner(o_p)$	$G' = G(execute(s, o_p))$
$access(s, t, \alpha_r)$	$s \in S$, $t \in T$, $\alpha_r \in \{insert_r,$ $update_r, delete_r\}$, $triggers(t, \alpha_r) =$ $= (o_{t,1}, \ldots, o_{t,k})$, $k \geqslant 0$, $\forall i \in \{1, \ldots, k\}(o_{t,i} \in O_t)$, $(user(s), t, \alpha_r) \in R_e$ или $[s] \neq \varnothing$ и $owner([s]) =$ $= owner(t)$	$G' = G(execute(s, o_{t,1}), \ldots, execute(s, o_{t,k}))$

Необходимость во введении нового обозначения $assoc_f(s)$ обусловлена тем, что в отличие от других моделей полагается, что в каждый момент времени множество функционально-ассоциированных с субъект-сессией s сущностей $assoc_f(s)$ либо содержит ровно один выполняемый им в текущий момент времени триггер или процедуру, либо не содержит элементов. Последнее означает, что субъект-сессия выполняет SQL-код, который не относится ни к одной процедуре или триггеру. В последнем случае считается, что выполнение такого SQL-кода эквивалентно непосредственному применению субъект-сессией правил преобразования состояний из множества OP_{ext}. В дальнейшем будем полагать, что по определению выполняется равенство $[s] = assoc_f(s)$.

SQL-код может либо относиться к одной из процедур или триггеров, либо не относиться ни к одной из них. Считается, что SQL-код содержит инструкции, выполнение которых непосредственно приводит к применению внешних правил из множества OP_{ext}. При этом предполагается, что SQL-код не может содержать инструкции, выполнение которых непосредственно приводит к применению внутренних правил из множества OP_{int}. Выполнение внутренних правил инициируется опосредованно как составная часть преобразований системы, задаваемых внешними правилами. Например, выполнение SQL-кода триггеров осуществляется только при применении внешних правил, связанных со вставкой, удалением и обновлением записей таблиц.

В рамках модели предполагается, что триггеры и процедуры не содержат динамически генерируемого SQL-кода и условий, а их выполнение приводит к применению фиксированной последовательности правил, задаваемой соответствующим значением функции *operations*.

В СУБД MS SQL Server существует специальный режим проверки прав доступа при наличии «цепочки владения» (*ownership chaining*). При выполнении DML-инструкций (Data Manipulation Language) SQL-кода процедуры не выполняется проверка прав доступа при обращении к сущностям, владелец которых совпадает с владельцем выполняемой процедуры. Аналогичные проверки не производятся и в отношении DML-инструкций SQL-кода триггеров. Модель учитывает данную особенность функционирования СУБД при задании условий применения внешних правил преобразования состояний системы.

Определение 6. Обозначим через O_t^* множество всех конечных последовательностей триггеров. Определим $triggers\colon T \times \{insert_r, update_r, delete_r\} \to O_t^*$ — функцию, задающую для каждой таблицы связанную с ней последовательность триггеров и удовлетворяющую следующим условиям.

У с л о в и е 1. Для каждой таблицы $t \in T$ и каждого права доступа $\alpha_r \in \{insert_r, update_r, delete_r\}$ выполняется $triggers(t, \alpha_r) \subseteq H(t)$.

У с л о в и е 2. Для каждого триггера $o_t \in O_t$ существует единственная таблица $t \in T$ и единственное право доступа $\alpha_r \in \{insert_r, update_r, delete_r\}$, такое, что $o_t \in triggers(t, \alpha_r)$.

Если существуют таблица $t \in T$, право доступа $\alpha_r \in \{insert_r, update_r, delete_r\}$ и триггер $o_t \in triggers(t, \alpha_r)$, то будем говорить, что триггер o_t имеет тип α_r и связан с таблицей t.

Каждый триггер имеет тип, задаваемый функцией *triggers* и соответствующий определённому виду доступа. Реализация субъект-сессией права доступа вида $\alpha_r \in \{insert_r, update_r, delete_r\}$ к таблице $t \in T$ с применением правила $access(s, t, \alpha_r)$ (см. табл. 2) автоматически инициирует выполнение ею SQL-кода всех триггеров из последовательности $triggers(t, \alpha_r)$.

Введём следующие обозначения:

— $G = (E_H,\ H,\ UA,\ S,\ R_d,\ Gr_d,\ user,\ user\_stack,\ assoc_f,\ owner,\ cmode,\ execute\_as,\ triggers, operations, tr)$ — состояние системы;
— $\Sigma(G^*, OP)$ — система, где G^* — множество всех возможных состояний;
— $\Sigma(G^*, OP, G_0)$ — система $\Sigma(G^*, OP)$ с начальным состоянием G_0.

Отметим, что из состояния системы исключены элементы, которые не изменяются либо могут быть определены на основании других элементов.

В дальнейшем будем использовать следующее предположение.

Предположение 4. В начальном состоянии системы $\Sigma(G^*, OP, G_0)$ отсутствуют субъект-сессии, то есть $S_0 = \varnothing$, и последовательность tr_0 не содержит элементов.

2. Правила преобразования состояний

Будем использовать запись вида $G'\{I'_1, \ldots, I'_k\} \sim G$ для обозначения того, что все элементы состояния G' равны соответствующим элементам состояния G, за исключением элементов $I'_1, \ldots, I'_k$ состояния G'. Будем также использовать запись вида $f'\{x_1, \ldots, x_k\} \sim f$ для обозначения того, что значения функции $f'(x)$ равны значениям функции $f(x)$ для всех аргументов x, за исключением $x_1, \ldots, x_k$. Кроме того, будем использовать запись вида $f'\{x_1 \mapsto y_1, \ldots, x_k \mapsto y_k\} \sim f$ для обозначения того, что значения функции $f'(x)$ равны значениям функции $f(x)$ для всех аргументов x, кроме $x_1, \ldots, x_k$, и при этом $f'(x_1) = y_1$, …, $f'(x_k) = y_k$. Очевидно, что если выполняется $G'\{I'_1, \ldots, I'_k\} \sim G$, то также верно и $G\{I_1, \ldots, I_k\} \sim G'$.

Используем обозначение: $G \vdash_{op} G'$ — переход системы $\Sigma(G^*, OP)$ из состояния G в состояние G' с применением правила преобразования состояний $op \in OP$, при этом если условия применения правила op не выполняются, то по определению справедливо равенство $G' = G$.

Запись вида $G' = G(op_1, \ldots, op_k)$ означает, что состояние G' есть результат последовательного применения правил преобразования состояний $op_1, \ldots, op_k$, начиная с состояния G, то есть существуют состояния $G_1, \ldots, G_k$, такие, что $G \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_k} G_k$ и $G' = G_k$.

В модели определены приведённые в табл. 1 и 2 правила преобразования состояний из множества OP, в которых учтены особенности управления доступом СУБД MS SQL Server 2012.

Замечание 4. Будем считать, что все правила из табл. 1 и 2, за исключением правила $create\_session(s, u)$, получают в качестве первого параметра субъект-сессию, инициировавшую их выполнение.

Будем использовать обозначение $op\langle s\rangle$ в случае, если применение правила инициировано субъект-сессией s. Будем считать, что применение правила $create\_session(s, u)$ также инициировано субъект-сессией s.

Определение 7. Будем говорить, что применение правила $op \in OP$ инициировано учётной записью пользователя $u \in U$, если либо $op = create\_session(s, u)$, либо op имеет в качестве первого параметра субъект-сессию s, такую, что $u = user(s)$.

Поясним приведённые в табл. 1 условия и результаты применения внутренних правил преобразования системы.

В отличие от внешних, внутренние правила применяются субъект-сессиями опосредованно в ходе выполнения SQL-кода как составная часть преобразований системы, задаваемых внешними (см. описание правил $execute\_procedure$, $access$, $switch$ в табл. 2) или другими внутренними правилами (см. описание правила $execute$).

В результате применения внутреннего правила $do\_switch(s, o, u)$ в качестве учётной записи, от имени которой функционирует субъект-сессия s, устанавливается u, а o становится функционально-ассоциированной с s сущностью. Запись вида (tr, op), использованная при определении нового значения tr', означает, что в возможно пустую последовательность tr добавляются данные о применяемом правиле op.

Внутреннее правило $execute(s, o)$ задаёт состояние системы после выполнения SQL-кода, содержащегося в процедуре или триггере $o \in O_p \cup O_t$ субъект-сессией $s \in S$. Перед выполнением SQL-кода сущности o производится изменение учётной записи пользователя, от имени которой субъект-сессия s будет последовательно применять внешние правила $op_1, \ldots, op_k$. Учётная запись пользователя, от имени которой функционирует субъект-сессия s, устанавливается в соответствии с режимом выполнения кода процедуры или триггера o, задаваемых значением функции $execute\_as(o)$. Кроме того, в результате применения правила do_switch перед выполнением SQL-кода сущности o в качестве функционально-ассоциированной с субъект-сессией s сущностью устанавливается сущность o. После завершения описанных подготовительных действий производится выполнение SQL-кода сущности o, что приводит к реализации преобразований состояний системы, задаваемых последовательностью правил $operations(o) = (op_1, \ldots, op_k)$. После выполнения кода с использованием правила do_switch производится восстановление исходной функционально-ассоциированной сущности с субъект-сессией s, а также учётной записи пользователя, от имени которой она выполнялась до применения внутреннего правила $execute(s, o)$. После завершения описанных действий производится восстановление исходной последовательности учётных записей, от имени которых выполнялась субъект-сессия s, до применения правила $execute(s, o)$.

В табл. 2 приведены внешние правила преобразования состояний из множества OP_{ext}, которые могут быть непосредственно применены субъект-сессиями. Поясним условия и результаты применения внешних правил преобразования состояний.

В результате применения правила $create\_session(s, u)$ создаётся новая субъект-сессия s, которая в дальнейшем может выполнять действия от имени учётной записи пользователя $u \in U$. Заметим, что после создания субъект-сессии множество функционально-ассоциированных с ней сущностей пусто. Отметим, что изначально в последовательность учётных записей, от имени которых выполнялась субъект-сессия, заносится учётная запись пользователя u, инициировавшего создание субъект-сессии s. В реальных СУБД при подключении к базе данных пользователь должен указать информацию, аутентифицирующую его учётную запись.

Определение 8. Пусть $G \vdash_{op} G'$ — переход системы $\Sigma(G^*, OP)$ из состояния G в состояние G'. Назовём переход $G \vdash_{op} G'$ инициированным извне, если $op \in OP_{\text{ext}}$ и либо $op = create\_session(s, u)$, либо правило op получает в качестве первого параметра субъект-сессию s, такую, что $[s] = \varnothing$. Если переход $G \vdash_{op} G'$ инициирован извне, то будем называть правило op инициированным извне, при этом если $user(s) = u$, то будем использовать обозначение $op[u]$.

В дальнейшем будем считать, что выполняется следующее предположение.

Предположение 5. Применение правила $create\_session(s, u)$ может быть инициировано только извне.

В результате применения правила $switch(s, u)$ в качестве учётной записи, от имени которой функционирует субъект-сессия s, устанавливается u. Необходимым условием применения данного правила является наличие у учётной записи, от имени кото-

рой функционирует субъект-сессия s, права доступа $impersonate_r$ к учётной записи пользователя u. Запись вида $(user\_stack(s), u)$, использованная при определении нового значения $user\_stack'(s)$, означает, что в возможно пустую последовательность $user\_stack(s)$ добавляется учётная запись пользователя u. Отметим, что в результате применения правила $switch(s, u)$ функционально ассоциированная с s сущность не изменяется. Заметим, что правилу $switch$ соответствует SQL-оператор *EXECUTE AS*, используемый в реальных СУБД для выполнения действий от имени учётной записи заданного пользователя.

Правило $revert(s)$ предназначено для восстановления учётной записи, от имени которой функционировала субъект-сессия s до момента предыдущего применения ею правила $switch$. Заметим, что восстановление организовано по принципу *LIFO* (*Last In — First Out*). Единственное отличие состоит в том, что учётная запись пользователя, от имени которого была создана субъект-сессия s, никогда не удаляется из последовательности $user\_stack(s)$. В СУБД для восстановления контекста выполнения до применения оператора *EXECUTE AS* используется оператор *REVERT*.

Замечание 5. Из способа задания внешних правил в табл. 2 видно, что при применении правил учитываются права учётной записи пользователя, от имени которой в текущем состоянии выполняется субъект-сессия s, а не права учётной записи пользователя, инициировавшего создание субъект-сессии s. Следовательно, наличие права $impersonate_r$ к учётной записи u позволяет с использованием $switch$ инициировать применение любого внешнего правила преобразования состояний системы с правами учётной записи пользователя u. Это означает, что наличие права $impersonate_r$ к учётной записи u равносильно обладанию информации, аутентифицирующей учётную запись пользователя u. Кроме того, можно считать, что наличие права $impersonate_r$ к учётной записи u равносильно функционированию в кооперации с субъект-сессией, создание которой было инициировано ею.

В результате применения правила $grant\_right(s, p, e, \alpha_r, with\_grant)$ принципалу p предоставляется право доступа α_r к сущности e. Необходимым условием является наличие у учётной записи пользователя, от имени которой выполняется субъект-сессия s, права давать право доступа α_r к сущности e, то есть $user(s) \in Gr(e, \alpha_r)$. Если параметр $with\_grant$ равен `yes`, то принципалу p даётся также право предоставлять право α_r к сущности e. Правило $grant\_right$ соответствует SQL-оператору *GRANT*, позволяющему предоставлять права пользователям и ролям к сущностям базы данных.

Назначение правила $add\_member(s, r, u)$ состоит в предоставлении учётной записи пользователя u авторизации на роль r, а также на все роли, которые «иерархически подчинены» роли r. Необходимым условием применения данного правила является наличие у учётной записи, от имени которой выполняется субъект-сессия s, права доступа $alter_r$ к роли r.

Замечание 6. Исходя из способа задания внутренних и внешних правил преобразования состояний, можно сделать вывод, что единственным способом изменения множества действующих прав доступа учётных записей пользователей на предоставление прав доступа к сущностям Gr_e является применение правила $grant\_right$. В то же время изменение множества действующих прав доступа R_e возможно с применением либо правила $grant\_right$, либо add_member.

Правило $create\_container(s, c_p, c, mode)$ позволяет субъект-сессии s создать новый контейнер c, включив его в существующий контейнер c_p. Необходимым условием применения данного правила является наличие у учётной записи пользователя $user(s)$

права доступа $alter_r$ к контейнеру c_p. Владельцем вновь созданного контейнера c назначается в зависимости от режима установки владельца вновь создаваемых дочерних сущностей контейнера c_p либо учётная запись пользователя, инициировавшая применение правила (режим $cmode(c_p) = \texttt{creator}$), либо принципал владельца контейнера c_p (режим $cmode(c_p) = \texttt{parent}$). Режим установки владельца создаваемого контейнера устанавливается равным параметру $mode$.

В СУБД MS SQL Server способ установки владельца экземпляра СУБД и экземпляров баз данных соответствует режиму `creator`, то есть владельцем становится пользователь, инициировавший их создание. С другой стороны, у схем способ установки соответствует режиму `parent`. Таким образом, владельцем создаваемых таблиц и триггеров становится владелец схемы, в которой выполняется операция создания.

В результате применения правила $create\_procedure(s, c, o_p, md, op_1, \ldots, op_k)$ в контейнере c, не являющемся таблицей, создаётся новая процедура o_p, содержащая SQL-код, выполнение которого приводит к реализации правил преобразования состояний $op_1, \ldots, op_k$, при этом владельцем вновь созданной процедуры o_p становится принципал владельца контейнера c, а режим выполнения o_p устанавливается равным md. Необходимым условием выполнения данного правила является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к контейнеру c. Для назначения режима выполнения процедуры от имени заданной учётной записи пользователя требуется наличие к ней у $user(s)$ права доступа $impersonate_r$.

В результате применения правила $alter\_procedure(s, o_p, md, op_1, \ldots, op_k)$ производится изменение режима выполнения существующей процедуры o_p, а также замена её SQL-кода на новый, выполнение которого приводит к реализации правил преобразования состояний $op_1, \ldots, op_k$. Необходимым условием выполнения данного правила является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к процедуре o_p. Для назначения режима выполнения процедуры от имени учётной записи пользователя требуется наличие к ней права доступа $impersonate_r$. Отметим, что, в отличие от правила $create\_procedure$, владелец процедуры o_p не изменяется.

В результате применения правила $create\_trigger(s, t, o_t, \alpha_r, md, op_1, \ldots, op_k)$ в таблице t создаётся новый триггер o_t, содержащий SQL-код, выполнение которого приводит к реализации правил преобразования состояний $op_1, \ldots, op_k$, при этом владельцем вновь созданного триггера o_t становится принципал владельца таблицы t, а режим выполнения o_t задаётся равным md. Созданный триггер активизируется только при реализации к таблице t права доступа α_r с использованием правила $access(s, t, \alpha_r)$. Необходимым условием выполнения правила $create\_trigger$ является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к таблице t. Для назначения режима выполнения триггера от имени заданной учётной записи пользователя требуется наличие к ней у $user(s)$ права доступа $impersonate_r$.

В результате применения правила $alter\_trigger(s, t, o_t, md, op_1, \ldots, op_k)$ в таблице t производится модификация режима выполнения существующего триггера o_t, а также замена его SQL-кода на новый, выполнение которого приводит к реализации правил преобразования состояний $op_1, \ldots, op_k$. Необходимым условием выполнения данного правила является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к таблице t. Для назначения режима выполнения триггера от имени учётной записи пользователя требуется наличие к ней у $user(s)$ права доступа $impersonate_r$. Отметим, что, в отличие от правила $create\_trigger$, не выполняется изменение учётной записи владельца триггера o_t, а также вида права доступа, при реализации которого активизируется триггер o_t.

Замечание 7. В результате применения правил *create_container*, *create_procedure* и *create_trigger* не выполняется добавление в множества R_d и Gr_d прав доступа учётной записи владельца к вновь созданной сущности. Это связано с тем, что при задании учётной записи владельца в соответствии с правилами задания в множества R_e и Gr_e включаются права доступа всех её иерархических владельцев.

Будем использовать обозначение $op \prec expression$ в случае, если операция op имеет вид $expression$. Например, если операция op имеет вид $create\_session(s, \ldots)$, то будем использовать обозначение $op \prec create\_session(s, \ldots)$. Будем использовать обозначение $op \not\prec expression$ в случае, если операция op не имеет вид $expression$.

Предположение 6. Будем считать, что выполнение SQL-кода триггеров и процедур не приводит к применению правил *create_container*, *create_procedure*, *alter_procedure*, *create_trigger* или *alter_trigger*, то есть перечисленные правила могут быть применены только извне. Следовательно, при применении правила $op \prec create\_procedure(\ldots, op_1, \ldots, op_k)$, либо $op \prec alter\_procedure(\ldots, op_1, \ldots, op_k)$, либо $op \prec create\_trigger(\ldots, op_1, \ldots, op_k)$, либо $op \prec alter\_trigger(\ldots, op_1, \ldots, op_k)$ для каждого $i = 1, \ldots, k$ правило $op_i \not\prec create\_procedure$, $op_i \not\prec alter\_procedure$, $op_i \not\prec create\_trigger$ и $op_i \not\prec alter\_trigger$.

Замечание 8. Заметим, что если $o \in O_p \cup O_t$, $operations(o) = (op_1, \ldots, op_k)$, где $k \geqslant 0$, то $op_i \not\prec create\_session$ для каждого $i = 1, \ldots, k$. Предположим противное: существует $i \in \{1, \ldots, k\}$, такое, что $op_i \prec create\_session$. В соответствии со способом задания внутреннего правила $execute(s, o)$ на время применения правил сущности o, она становится функционально-ассоциированной с субъект-сессией s. Это противоречит предположению о том, что применение правила $op_i \prec create\_session$ может быть инициировано только извне. Следовательно, правила, связанные с процедурами и триггерами, не могут включать правила вида *create_session*. Кроме того, при применении правил вида $create\_procedure(\ldots, op_1, \ldots, op_k)$, $alter\_procedure(\ldots, op_1, \ldots, op_k)$, $create\_trigger(\ldots, op_1, \ldots, op_k)$ и $alter\_trigger(\ldots, op_1, \ldots, op_k)$ для каждого $i = 1, \ldots, k$ выполняется $op_i \not\prec create\_session$.

Допустим, что $o \in O_p \cup O_t$, $operations(o) = (op_1, \ldots, op_k)$, где $k \geqslant 0$. Будем использовать обозначение $op \in operations(o)$ в случае, если существует $i \in \{1, \ldots, k\}$, такое, что $op = op_i$.

Замечание 9. Пусть $o \in O_p \cup O_t$ и $op \in operations(o)$. В соответствии с предположением 6 и замечанием 8, правило op не может иметь вид *create_container*, *create_procedure*, *alter_procedure*, *create_trigger*, *alter_trigger* и *create_session*. Учитывая это, а также то, что $op \in OP_{\text{ext}}$, по определению функции *operations* получаем, что либо $op \prec grant\_right$, либо $op \prec add\_member$, либо $op \prec execute\_procedure$, либо $op \prec access$.

Для упрощения анализа условий несанкционированного распространения прав доступа будем считать, что выполняется следующее предположение.

Предположение 7. Считается, что в начальном состоянии $\Sigma(G^*, OP, G_0)$ для каждого $o \in O_{p_0} \cup O_{t_0}$ и каждого $op \in operations_0(o)$ выполняется $op \not\prec grant\_right$ и $op \not\prec add\_member$.

Это предположение должно соблюдаться разработчиками автоматизированных информационных системы, использующих данную модель. Таким образом, предоставления прав доступа и управление членством в группах должно производиться вне процедур и триггеров.

Правило $execute\_procedure(s, o_p)$ позволяет субъект-сессии s выполнить SQL-код процедуры o_p. Необходимым условием применения данного правила является либо наличие у учётной записи пользователя, от имени которой выполняется s, права доступа $execute_r$ к процедуре o_p, либо совпадение принципала владельца процедуры o_p и принципала владельца процедуры или триггера, в рамках выполнения SQL-кода которых предпринята попытка применения правила $execute\_procedure(s, o_p)$. Последний случай соответствует специальному режиму проверки прав доступа при наличии «цепочки владения». Отметим, что при выполнении SQL-кода процедуры может осуществляться активация других процедур и триггеров.

Применение правила $access(s, t, \alpha_r)$ приводит к активации триггеров типа α_r таблицы t субъект-сессией s. Необходимым условием применения данного правила является либо наличие у учётной записи пользователя, от имени которой выполняется s, права доступа α_r к таблице t, либо совпадение принципала владельца таблицы t и принципала владельца процедуры или триггера, в рамках выполнения SQL-кода которых предпринята попытка применения правила $access(s, t, \alpha_r)$. Отметим, что при выполнении SQL-кода триггера может осуществляться активация других процедур и триггеров.

Замечание 10. Заметим, что в реальных СУБД активация триггеров таблицы выполняется только при внесении в неё изменений.

Замечание 11. Будем считать, что при активации правил преобразования состояний, связанных с процедурами и триггерами, не возникает циклов, являющихся результатом повторного применения правил ранее активированных сущностей.

Замечание 12. В соответствии со способом задания применение внешних и внутренних правил не приводит к удалению элементов из множеств, являющихся элементами состояния системы $\Sigma(G^*, OP)$. Следовательно, все правила системы $\Sigma(G^*, OP)$ являются монотонными.

Учитывая монотонность правил системы, а также то, что в условиях применения правил $grant\_right$, add_member $alter\_procedure$, $alter\_trigger$, $execute\_procedure$, $access$ нет проверок на отсутствие элементов в состоянии системы $\Sigma(G^*, OP)$, отметим, что верно следующее замечание.

Замечание 13. Пусть G — состояние системы $\Sigma(G^*, OP)$ и $op_1, \ldots, op_k \in OP$, где $k \geqslant 1$, $G' = G(op_1, \ldots, op_k)$, op является одним из перечисленных выше правил. Пусть в состоянии G выполнены условия применения правила op субъект-сессией, функционирующей от имени учётной записи пользователя $u \in U$. Тогда в состоянии G' также будут выполнены условия применения правила op субъект-сессией, функционирующей от имени учётной записи пользователя u.

3. Функция *vestige*

В соответствии со способом задания правил преобразования состояний в результате применения правил $execute\_procedure$ и $access$ может быть инициировано выполнение правил триггеров и процедур. Введём функцию $vestige(G, op_1, \ldots, op_k)$, значением которой является последовательность всех правил преобразований состояний системы, применение которых является результатом последовательного применения правил $op_1, \ldots, op_k$ в состоянии G.

Определение 9. Пусть заданы состояния системы $G, G_1, \ldots, G_k$, где $k \geqslant 1$, и правила преобразования состояний $op_1, \ldots, op_k$, такие, что $G \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_k} G_k$ и для каждого $i = 1, \ldots, k$ переход $G_{i-1} \vdash_{op_i} G_i$ инициирован извне и $tr_k = (tr, \widehat{op}_1, \ldots, \widehat{op}_m)$,

где $m \geqslant 0$. Обозначим через $vestige(G, op_1, \ldots, op_k)$ последовательность правил преобразования состояний $(\widehat{op}_1, \ldots, \widehat{op}_m)$.

Замечание 14. Заметим, что последовательность правил $vestige(G, op_1, \ldots, op_k)$ однозначно определяется состоянием системы G и последовательностью инициированных извне правил $op_1, \ldots, op_k$, поэтому в дальнейшем $vestige$ будет рассматриваться как функция $G^* \times OP^*_{\text{ext}} \to OP^*$.

Замечание 15. Отметим, что в соответствии со способом задания перехода системы из состояния в состояние последовательность правил преобразования состояний $vestige(G, op_1, \ldots, op_k)$ не включает правила, условия применения которых не выполняются.

Пусть заданы состояния системы $G_0, \ldots, G_k$, где $k \geqslant 1$, и правила преобразования состояний $op_1, \ldots, op_k$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_k} G_k$ и для каждого $i = 1, \ldots, k$ переход $G_{i-1} \vdash_{op_i} G_i$ инициирован извне и $vestige(G_0, op_1, \ldots, op_k) = (\widehat{op}_1, \ldots, \widehat{op}_m)$. Пусть состояния $\widehat{G}_0, \widehat{G}_1, \ldots, \widehat{G}_m$ таковы, что $\widehat{G}_0 = G_0$ и $\widehat{G}_0 \vdash_{\widehat{op}_1} \widehat{G}_1 \vdash_{\widehat{op}_2} \ldots \vdash_{\widehat{op}_m} \widehat{G}_m$. Функция $vestige$ обладает следующими свойствами.

Свойство 1. $G_k = \widehat{G}_m$.

Свойство 2. $vestige(G_0, op_1, \ldots, op_k) = (vestige(G_0, op_1, \ldots, op_{k-1}), vestige(G_{k-1}, op_k)) = (vestige(G_0, op_1), vestige(G_1, op_2), \ldots, vestige(G_{k-1}, op_k))$ (для удобства будем считать, что запись $((\widehat{op}_1, \ldots, \widehat{op}_m), (\widehat{op}_{m+1}, \ldots, \widehat{op}_n))$ эквивалентна последовательности $(\widehat{op}_1, \ldots, \widehat{op}_m, \widehat{op}_{m+1}, \ldots, \widehat{op}_n)$).

Свойство 3. Если в состоянии G_0 не выполнены условия применения правила $op \in OP_{\text{ext}}$, то $vestige(G_0, op) = ()$.

Свойство 4. Если выполнены условия применения правила $op \in OP_{\text{ext}}$ в состоянии G_0 и $op \not\prec access$ и $op \not\prec execute\_procedure$, то $vestige(G, op) = (op)$.

Свойство 5. Для каждого $i = 1, \ldots, m$ либо $\widehat{op}_i \prec switch$, либо $\widehat{op}_i \prec revert$, либо $\widehat{op}_i \prec create\_session$, либо $\widehat{op}_i \prec create\_procedure$, либо $\widehat{op}_i \prec alter\_procedure$, либо $\widehat{op}_i \prec create\_trigger$, либо $\widehat{op}_i \prec alter\_trigger$, либо $\widehat{op}_i \prec grant\_right$, либо $\widehat{op}_i \prec add\_member$.

Свойство 6. Для каждого $i = 1, \ldots, m$ выполняется $\widehat{op}_i \not\prec access$ и $\widehat{op}_i \not\prec execute\_procedure$.

Выполнение указанных свойств следует из определения функции $vestige$ и способа задания внутренних и внешних правил преобразования состояний в табл. 1 и 2.

4. Анализ условий выполнения SQL-кода от имени заданной учётной записи

Несанкционированное выполнение нарушителем произвольного SQL-кода от имени другой учётной записи является одной из главных угроз безопасности автоматизированной информационной системы. Наличие у нарушителя возможности выполнения произвольного SQL-кода от имени административной учётной записи может привести к компрометации всей системы в целом. Следовательно, необходимо провести всесторонний теоретический анализ условий, при выполнении которых возможно выполнение пользователем произвольного SQL-кода от имени заданной учётной записи. В рамках модели возможность выполнение произвольного SQL-кода от имени заданной учётной записи рассматривается как возможность применения от её имени произвольного правила.

Введём определение возможности применения правила от имени заданной учётной записи.

Определение 10. Пусть G — состояние системы $\Sigma(G^*, OP)$, $u, u' \in U$, $op \in OP_{\text{ext}}$, $s \notin S$. Пусть в состоянии G выполнены условия применения внешнего правила op субъект-сессией, функционирующей от имени учётной записи пользователя u'. Определим предикат $can\_execute\_as(u, u', op, G)$, истинный тогда и только тогда, когда существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $vestige(G, op_1\langle s\rangle, \ldots, op_k\langle s\rangle) = (\widehat{op}_1, \ldots, \widehat{op}_m)$ и существует $i \in \{2, \ldots, m\}$, что $\widehat{op}_i = op$ и $\widehat{user}_{i-1}(s) = u'$.

Для теоретического анализа условий выполнения SQL-кода от имени заданной учётной записи введём следующее определение.

Определение 11. Пусть G — состояние системы $\Sigma(G^*, OP)$, $u, u' \in U$. Определим предикат $can\_act\_as(u, u', G)$, истинный тогда и только тогда, когда истинен предикат $can\_execute\_as(u, u', op, G)$ для любого внешнего правила $op \in OP_{\text{ext}}$, для которого в состоянии G выполнены условия его применения субъект-сессией, функционирующей от имени учётной записи пользователя u'.

В соответствии с замечанием 5 обладание учётной записью пользователя u правом доступа $impersonate_r$ к учётной записи пользователя u' позволяет применить правило $switch$ и в дальнейшем выполнить любое внешнее правило преобразования состояния системы от её имени. Для анализа возможности получения права $impersonate_r$ введём следующее определение.

Определение 12. Пусть G — состояние системы $\Sigma(G^*, OP)$, $u, u' \in U$, $s \notin S$. Определим предикат $can\_become(u, u', G)$, истинный тогда и только тогда, когда существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $G_k = G(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$ и $(u, u', impersonate_r) \in R_{e_k}$.

В случае истинности предиката $can\_become(u, u', G)$ субъект-сессия s, созданная пользователем u, без кооперации с другими субъект-сессиями может получить право доступа $impersonate_r$ к учётной записи пользователя u'.

4.1. Эквивалентность возможности выполнения произвольного SQL-кода от имени учётной записи пользователя и возможности получения к ней права олицетворения

Справедливо следующее утверждение о связи между возможностью выполнения произвольного SQL-кода от имени заданной учётной записи и возможностью получения к ней права доступа $impersonate_r$.

Утверждение 2. Пусть G — состояние системы $\Sigma(G^*, OP)$, $u, u' \in U$. Тогда предикат $can\_act\_as(u, u', G)$ истинен тогда и только тогда, когда истинен предикат $can\_become(u, u', G)$.

Доказательство. Пусть $s \notin S$.

Пусть истинен предикат $can\_act\_as(u, u', G)$. В соответствии со способом задания множества Gr_e владелец сущности может предоставить к ней любые права доступа. Отсюда, учитывая, что $owner(u') = u'$, следует, что $u' \in Gr(u', impersonate_r)$. Это означает, что в состоянии G выполнены условия применения $op = grant\_right(s, u, u', impersonate_r, \texttt{yes})$ в случае, если субъект-сессия s функционирует от имени учётной записи пользователя u'. В этом случае, по определению 11, истинен предикат $can\_execute\_as(u, u', op, G)$. Значит, существуют инициированные извне прави-

ла $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $vestige(G, op_1\langle s\rangle, \ldots, op_k\langle s\rangle) = (\widehat{op}_1, \ldots, \widehat{op}_m)$, и существует $i \in \{2, \ldots, m\}$, что $\widehat{op}_i = op$ и $\widehat{user}_{i-1}(s) = u'$. В соответствии с замечанием 15 последовательность $vestige(G, op_1\langle s\rangle \ldots, op_k\langle s\rangle)$ не включает правила, условия применения которых не выполняются. Значит, выполнены условия применения правила $op = grant\_right(s', u, u', impersonate_r, \texttt{yes})$, откуда $(u, u', impersonate_r) \in \widehat{R}_{e_i}$. Из монотонности правил преобразования состояний системы следует, что $(u, u', impersonate_r) \in R_{e_k}$, а значит, по определению истинен предикат $can\_become(u, u', G)$.

Пусть истинен предикат $can\_become(u, u', G)$. По определению 12 существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $G_k = G(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$ и $(u, u', impersonate_r) \in R_{e_k}$. Пусть для правила $op \in OP_{\text{ext}}$ в состоянии G выполнены условия его применения субъект-сессией, функционирующей от имени учётной записи пользователя u'. Легко заметить, что в результате применения внешних правил преобразований значение функции $user\_stack$ не изменяется. Таким образом, $user\_stack_k(s) = user\_stack_1(s) = (u)$, откуда $user_k(s) = u$. Так как $(u, u', impersonate_r) \in R_{e_k}$, в состоянии G_k выполнены условия применения правил $op_{k+1}\langle s\rangle = switch(s, u')$ и $user_{k+1}(s) = u'$.

Покажем, что в состоянии G_{k+1} выполнены условия применения правила op.

Во всех внешних правилах, за исключением $create\_session$, $create\_trigger$, $create\_procedure$ и $create\_container$, нет проверок на отсутствие элементов в состоянии системы. Отсюда, в силу монотонности правил преобразования состояний, в состоянии G_{k+1} выполнены условия применения op, а следовательно, по определению истинен предикат $can\_execute\_as(u, u', op, G)$.

Согласно предположению 5, применение правила $create\_session$ не может быть инициировано ранее созданной субъект-сессией, а следовательно, в рамках определения 10 правило op не может иметь вид $create\_session$.

Пусть $op \prec create\_container(\ldots, c, \ldots)$. Если $c \notin C_{k+1}$, то в силу монотонности правил в состоянии G_{k+1} выполнены условия применения правила op. Пусть $c \in C_{k+1}$. Тогда, согласно предположению 6, существует $i \in \{1, \ldots, k\}$, такое, что $op_i\langle s\rangle \prec create\_container(\ldots, c, \ldots)$. Заменим в правилах преобразования состояний $op_i\langle s\rangle, \ldots, op_{k+1}\langle s\rangle$ все случаи использования контейнера c на $c' \notin C_{i-1}$. Очевидно, что в этом случае $(u, u', impersonate_r) \in R_{e_{k+1}}$ и $c \notin C_{k+1}$, тогда в силу монотонности правил в состоянии G_{k+1} выполнены условия применения правила op, а следовательно, истинен предикат $can\_execute\_as(u, u', op, G)$.

Проводя аналогичные рассуждения для случаев $create\_procedure$ и $create\_trigger$, можно показать, что предикат $can\_execute\_as(u, u', op, G)$ также будет истинен.

Следовательно, предикат $can\_execute\_as(u, u', op, G)$ истинен для любого правила $op \in OP_{\text{ext}}$, для которого в состоянии G выполнены условия применения субъект-сессией, функционирующей от имени учётной записи пользователя u', а значит, по определению истинен предикат $can\_act\_as(u, u', G)$. ■

Следовательно, для определения условий истинности предиката $can\_act\_as(u, u', G)$ необходимо и достаточно обосновать условия истинности предиката $can\_become(u, u', G)$.

4.2. Достаточные условия выполнения *SQL*-кода от имени заданной учётной записи пользователя

Покажем, что если в начальном состоянии системы G_0 в результате применения извне правил субъект-сессией, созданной пользователем u, были выполнены правила

вида $grant\_right$ и add_member от имени учётной записи пользователя u', то предикат $can\_act\_as(u, u', G_0)$ истинен.

Утверждение 3. Пусть G_0 — начальное состояние системы $\Sigma(G^*, OP)$, $s \notin S$, $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, $op_1\langle s\rangle = create\_session(s, u)$, $vestige(G_0, op_1\langle s\rangle, \ldots, op_k\langle s\rangle) = (\widehat{op}_1, \ldots, \widehat{op}_m)$ и существует $i \in \{2, \ldots, m\}$, что $\widehat{op}_i \prec grant\_right$ или $\widehat{op}_i \prec add\_member$ и $\widehat{user}_{i-1}(s) = u'$. Тогда истинен предикат $can\_act\_as(u, u', G_0)$.

Доказательство. Возможны два случая: 1) $\widehat{[s]}_{i-1} = \varnothing$ и 2) $\widehat{[s]}_{i-1} = o$, где $o \in O_{p_{i-1}} \cup O_{t_{i-1}}$.

Рассмотрим первый случай. Если $\widehat{[s]}_{i-1} = \varnothing$, то применение правила $\widehat{op}_i$ было инициировано извне субъект-сессией s. Следовательно, существует $j \in \{1, \ldots, k\}$, такое, что $op_j\langle s\rangle = \widehat{op}_i$. Так как $\widehat{user}_{i-1}(s) = u'$, имеем $user_{j-1}(s) = u'$. Положим $op_j\langle s\rangle = grant\_right(s, u, u', impersonate_r, \texttt{no})$. Так как $(u', u', impersonate_r) \in R_{e_{j-1}}$, в состоянии G_{j-1} выполнены условия применения правила $op_j\langle s\rangle$ субъект-сессией, функционирующей от имени учётной записи пользователя u', откуда $(u, u', impersonate_r) \in R_{e_j}$. Тогда по определению истинен предикат $can\_become(u, u', G_0)$ и, в соответствии с утверждением 2, истинен предикат $can\_act\_as(u, u', G_0)$.

Рассмотрим второй случай. Возможны следующие два варианта: 2.1) $o \in O_{p_0} \cup O_{t_0}$ и 2.2) $o \notin O_{p_0} \cup O_{t_0}$.

Рассмотрим случай 2.1. В соответствии с предположением 7 в начальном состоянии $\Sigma(G^*, OP, G_0)$ для каждого $o \in O_{p_0} \cup O_{t_0}$ и каждого $op \in operations_0(o)$ выполняется $op \not\prec grant\_right$ и $op \not\prec add\_member$. Следовательно, существует $i' \in \{2, \ldots, i-1\}$, такое, что либо $\widehat{op}_{i'} \prec alter\_procedure$, либо $\widehat{op}_{i'} \prec alter\_trigger$. Выберем значение i' максимальным. В соответствии с предположением 6 применение правил $alter\_procedure$ и $alter\_trigger$ возможно только извне. Следовательно, существует $j \in \{1, \ldots, k\}$, такое, что $op_j\langle s\rangle = \widehat{op}_{i'}$. Возможны два случая: $o \in O_{p_0}$ и $o \in O_{t_0}$. В первом случае $op_j\langle s\rangle = alter\_procedure(\widetilde{op}_1, \ldots, \widetilde{op}_l)$ и существует $j' \in \{1, \ldots, l\}$, такое, что $\widetilde{op}_{j'} = \widehat{op}_i$. Заменим в правиле $op_j\langle s\rangle$ в последовательности $(\widetilde{op}_1, \ldots, \widetilde{op}_l)$ правило $\widetilde{op}_{j'}$ на $grant\_right(s, u, u', impersonate_r, \texttt{no})$. Из максимальности выбора i' следует, что в состоянии $\widehat{G}_{j-1}$ верно равенство $\widehat{operations}_{j-1}(o) = (\widetilde{op}_1, \ldots, \widetilde{op}_{j'-1}, grant\_right(s, u, u', impersonate_r, \texttt{no}), \widetilde{op}_{j'+1}, \ldots, \widetilde{op}_l)$, откуда $\widehat{op}_i \prec grant\_right(s, u, u', impersonate_r, \texttt{no})$ и $\widehat{user}_{i-1}(s) = u'$. Так как $(u', u', impersonate_r) \in \widehat{R}_{e_{i-1}}$, в состоянии $\widehat{G}_{i-1}$ выполнены условия применения правила $\widehat{op}_i$ и $(u, u', impersonate_r) \in \widehat{R}_{e_i}$, а значит, в силу монотонности правил преобразования состояний, $(u, u', impersonate_r) \in R_{e_k}$. Тогда по определению истинен предикат $can\_become(u, u', G_0)$ и, по утверждению 2, истинен предикат $can\_act\_as(u, u', G_0)$. Случай $o \in O_{t_0}$ обосновывается аналогично.

Рассмотрим случай 2.2. В соответствии с предположением 6 применение правил $create\_procedure$ и $create\_trigger$ возможно только извне. Так как $o \notin O_{p_0} \cup O_{t_0}$, существует $j \in \{1, \ldots, k\}$, такое, что $o \notin O_{p_{j-1}} \cup O_{t_{j-1}}$, $o \notin O_{p_j} \cup O_{t_j}$ и либо $op_j\langle s\rangle \prec create\_procedure$, либо $op_j\langle s\rangle \prec create\_trigger$. Возможны следующие два случая: 1) $\widehat{operations}_{i-1}(o) \neq operations_j(o)$; 2) $\widehat{operations}_{i-1}(o) = operations_j(o)$. Если $\widehat{operations}_{i-1}(o) \neq operations_j(o)$, то, проводя рассуждения, аналогичные случаю 2.1, можно показать, что истинен предикат $can\_act\_as(u, u', G_0)$. Рассмотрим второй случай: $\widehat{operations}_j(o) = (\widetilde{op}_1, \ldots, \widetilde{op}_l)$, тогда существует $j' \in \{1, \ldots, l\}$, такое, что $\widetilde{op}_{j'} = \widehat{op}_i$. Заменим в правиле $op_j\langle s\rangle$ в последовательности $(\widetilde{op}_1, \ldots, \widetilde{op}_l)$ правило $\widetilde{op}_{j'}$ на $grant\_right(s, u, u', impersonate_r, \texttt{no})$. Так как $\widehat{operations}_{i-1}(o) = operations_j(o)$, получим $\widehat{op}_i \prec grant\_right(s, u, u', impersonate_r, \texttt{no})$ и $\widehat{user}_{i-1}(s) = u'$. В соответствии

со способом задания $(u', u', impersonate_r) \in \widehat{R}_{e_{i-1}}$, и значит, в состоянии $\widehat{G}_{i-1}$ выполнены условия применения правила $\widehat{op}_i$ и $(u, u', impersonate_r) \in \widehat{R}_{e_i}$, откуда, в силу монотонности правил преобразования состояний, $(u, u', impersonate_r) \in R_{e_k}$. Тогда по определению истинен предикат $can\_become(u, u', G_0)$ и в соответствии с утверждением 2 истинен предикат $can\_act\_as(u, u', G_0)$. ■

5. Ролевые цепочки получения прав доступа

Пусть G — состояние системы $\Sigma(G^*, OP)$, $p \in P$, $r \in R$, $e \in E$, $\alpha_r \in R_r$. Будем использовать обозначение $(p, r, e, \alpha_r) \lessdot G$ в случае, если $(p, r, alter_r) \in R_e$ и $(r, e, \alpha_r) \in R_e$.

Пусть $u \in U$, тогда будем использовать обозначение $(u, e, \alpha_r) \lhd G$ в случае, если существуют роли $r_1, \ldots, r_k \in R$, где $k \geqslant 1$, такие, что $(u, r_1, r_2, alter_r) \lessdot G$, для каждого $i = 1, \ldots, k-2$ выполняется $(r_i, r_{i+1}, r_{i+2}, alter_r) \lessdot G$ и $(r_{k-1}, r_k, e, \alpha_r) \lessdot G$. В последнем случае будем использовать также обозначение $(u, r_k, e, \alpha_r) \lhd G$. В случае $(u, e, \alpha_r) \lhd G$ будем говорить, что учётная запись пользователя u обладает «ролевой цепочкой» получения права доступа α_r к сущности e. Если $(u, r, e, \alpha_r) \lhd G$, то будем говорить, что учётная запись пользователя u обладает «ролевой цепочкой» получения права доступа α_r к сущности e через роль r.

Будем использовать обозначение $(u, e, \alpha_r) \blacktriangleleft G$ в случае, если существуют роли $r_1, \ldots, r_k \in R$, где $k \geqslant 1$, такие, что $(u, r_1, r_2, alter_r) \lessdot G$, для каждого $i = 1, \ldots, k-2$ выполняется $(r_i, r_{i+1}, r_{i+2}, alter_r) \lessdot G$ и $(r_{k-1}, r_k, alter_r) \in R_e$ и $r_k \in Gr(e, \alpha_r)$. Если $(u, e, \alpha_r) \blacktriangleleft G$, то будем говорить, что учётная запись пользователя u обладает «ролевой цепочкой» предоставления права доступа α_r к сущности e. Очевидно, что если $(u, e, \alpha_r) \blacktriangleleft G$, то $(u, e, \alpha_r) \lhd G$.

В силу монотонности правил преобразования состояний справедливы следующие замечания.

Замечание 16. Если G — состояние системы $\Sigma(G^*, OP)$, $p \in P$, $r \in R$, $e \in E$, $\alpha_r \in R_r$, $(p, r, e, \alpha_r) \lessdot G$, $op_1, \ldots, op_k \in OP_{\text{ext}}$, где $k \geqslant 1$, $G_k = G(op_1, \ldots, op_k)$, то $(p, r, e, \alpha_r) \lessdot G_k$.

Замечание 17. Если G — состояние системы $\Sigma(G^*, OP)$, $u \in U$, $e \in E$, $\alpha_r \in R_r$, $(u, e, \alpha_r) \lhd G$, $op_1, \ldots, op_k \in OP_{\text{ext}}$, где $k \geqslant 1$, $G_k = G(op_1, \ldots, op_k)$, то $(u, e, \alpha_r) \lhd G_k$.

Замечание 18. Если G — состояние системы $\Sigma(G^*, OP)$, $u \in U$, $e \in E$, $\alpha_r \in R_r$, $(u, e, \alpha_r) \blacktriangleleft G$, $op_1, \ldots, op_k \in OP_{\text{ext}}$, где $k \geqslant 1$, $G_k = G(op_1, \ldots, op_k)$, то $(u, e, \alpha_r) \blacktriangleleft G_k$.

Утверждение 4. Если G — состояние системы $\Sigma(G^*, OP)$, $u \in U$, $r \in R$, $e \in E$, $\alpha_r \in R_r$, $(u, r, e, \alpha_r) \lessdot G$, $s \in S$, $user(s) = u$, $G_1 = G(add\_member(s, r, u))$, то $(u, e, \alpha_r) \in \in R_{e_1}$.

Доказательство. Следует из того, что в состоянии G выполнены условия применения правила $add\_member(s, r, u)$ от имени учётной записи u и при этом $(r, e, \alpha_r) \in R_e$. ■

Докажем, что если в результате применения извне правил субъект-сессией, созданной пользователем u, у принципала p появится право доступа $alter_r$ к роли r, то в начальном состоянии G_0 существует учётная запись пользователя u', обладающая ролевой цепочкой получения права доступа α_r к сущности e через роль r, и u имеет возможность выполнения от её имени произвольного SQL-кода.

Утверждение 5. Пусть G_0 — начальное состояние системы $\Sigma(G^*, OP)$, $u \in U$, $p \in P$, $r \in R$, $e \in E_0$, $\alpha_r \in R_r$, $(r, e, \alpha_r) \in R_{e_0}$, $s \notin S_0$, $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $op_1\langle s\rangle = create\_session(s, u)$, $G_k = G_0(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$, $(p, r, alter_r) \notin R_{e_0}$ и

$(p, r, alter_r) \in R_{e_k}$. Тогда существует $u' \in U$, такое, что предикат $can\_act\_as(u, u', G_0)$ истинен и $(u', r, e, \alpha_r) \lhd G_0$.

Доказательство. Пусть $vestige(G_0, op_1\langle s\rangle, \ldots, op_k\langle s\rangle) = (\widehat{op}_1, \ldots, \widehat{op}_m)$.

Так как $(p, r, alter_r) \notin R_{e_0}$ и $(p, r, alter_r) \in R_{e_k}$, существует $i \in \{1, \ldots, m\}$, такое, что $(p, r, alter_r) \notin \widehat{R}_{e_{i-1}}$ и $(p, r, alter_r) \in \widehat{R}_{e_i}$. Заметим, что в соответствии с замечанием 15 в состоянии $\widehat{G}_{i-1}$ выполнены условия применения правила $\widehat{op}_i$. Пусть l — количество правил вида $grant\_right$ и add_member в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_i)$. Доказательство проведём индукцией по l.

Пусть $l = 1$. Обозначим $\widehat{user}_{i-1}(s)$ как u'. В соответствии с утверждением 3 истинен предикат $can\_act\_as(u, u', G_0)$. Так как $l = 1$, $\widehat{op}_i$ — единственное правило вида $grant\_right$ и add_member, откуда $\widehat{R}_{e_{i-1}} = R_{e_0}$.

Если $\widehat{op}_i \prec grant\_right$, то $\widehat{op}_i \prec grant\_right(s, \ldots, e', alter_r, \ldots)$, где $e' \in \{r, c_r\}$ и $u' \in \widehat{Gr}_{i-1}(e', alter_r)$, а значит, $(u', e', alter_r) \in \widehat{R}_{e_{i-1}}$. Заметим, что $r \leqslant e'$, откуда $(u', r, alter_r) \in \widehat{R}_{e_{i-1}}$. Ранее было показано, что $\widehat{R}_{e_{i-1}} = R_{e_0}$, а значит, $(u', r, alter_r) \in R_{e_0}$. По условию $(r, e, \alpha_r) \in R_{e_0}$, следовательно, $(u', r, e, \alpha_r) \lhd G_0$.

Пусть $\widehat{op}_i \prec add\_member$, тогда $p \in U$, $\widehat{op}_i = add\_member(s, r', p)$, где $r' \in R$, $(u', r', alter_r) \in \widehat{R}_{e_{i-1}}$ и $(r', r, alter_r) \in \widehat{R}_{e_{i-1}}$. Так как $\widehat{R}_{e_{i-1}} = R_{e_0}$, имеем $(u', r', alter_r) \in$ $\in R_{e_0}$ и $(r', r, alter_r) \in R_{e_0}$. По условию $(r, e, \alpha_r) \in R_{e_0}$, а значит, $(u', r, e, \alpha_r) \lhd G_0$.

Пусть условия утверждения выполняются для всех $l < N$, где $N > 1$. Покажем, что оно верно и при $l = N$. Обозначим $\widehat{user}_{i-1}(s)$ как u''. В соответствии с утверждением 3 истинен предикат $can\_act\_as(u, u'', G_0)$.

Если $\widehat{op}_i \prec grant\_right$, то $\widehat{op}_i \prec grant\_right(s, \ldots, e', alter_r, \ldots)$, где $e' \in \{r, c_r\}$ и $u'' \in \widehat{Gr}_{i-1}(e', alter_r)$, а значит, $(u'', e', alter_r) \in \widehat{R}_{e_{i-1}}$. Заметим, что $r \leqslant e'$, откуда $(u'', r, alter_r) \in \widehat{R}_{e_{i-1}}$. Возможны случаи: 1) $(u'', r, alter_r) \in R_{e_0}$; 2) $(u'', r, alter_r) \notin R_{e_0}$.

С л у ч а й 1. Положим $u' = u''$. По условию $(r, e, \alpha_r) \in R_{e_0}$, откуда $(u', r, e, \alpha_r) \lhd G_0$.

С л у ч а й 2. Так как $(u'', r, alter_r) \notin R_{e_0}$ и $(u'', e', alter_r) \in \widehat{R}_{e_{i-1}}$, существует $j \in \{1, \ldots, i-1\}$, что $(u'', r, alter_r) \notin \widehat{R}_{e_{j-1}}$ и $(u'', r, alter_r) \in \widehat{R}_{e_j}$. Так как $\widehat{op}_i \prec grant\_right$ и $j < i$, для u'', r и e выполнено предположение индукции с количеством правил вида $grant\_right$ и add_member в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_j)$, меньшим чем N. Тогда по предположению индукции существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и $(u', r, e, \alpha_r) \lhd G_0$.

Пусть $\widehat{op}_i \prec add\_member$. Тогда $p \in U$, $\widehat{op}_i = add\_member(s, r', p)$, где $r' \in R$, $(u'', r', alter_r) \in \widehat{R}_{e_{i-1}}$ и $(r', r, alter_r) \in \widehat{R}_{e_{i-1}}$. Возможны случаи: 1) $(u'', r', alter_r) \in R_{e_0}$ и $(r', r, alter_r) \in R_{e_0}$; 2) $(u'', r', alter_r) \notin R_{e_0}$ и $(r', r, alter_r) \in R_{e_0}$; 3) $(r', r, alter_r) \notin R_{e_0}$.

С л у ч а й 1. Положим $u' = u''$. По условию $(r, e, \alpha_r) \in R_{e_0}$, следовательно, $(u', r, e, \alpha_r) \lhd G_0$.

С л у ч а й 2. Так как $(u'', r', alter_r) \notin R_{e_0}$ и $(u'', r', alter_r) \in \widehat{R}_{e_{i-1}}$, существует $j \in \{1, \ldots, i-1\}$, такое, что $(u'', r', alter_r) \notin \widehat{R}_{e_{j-1}}$ и $(u'', r', alter_r) \in \widehat{R}_{e_j}$. Так как $\widehat{op}_i \prec add\_member$ и $(r', r, alter_r) \in R_{e_0}$, для u'', r' и r выполнено предположение индукции с количеством правил вида $grant\_right$ и add_member в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_j)$, меньшим чем N. Тогда по предположению индукции существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и $(u', r', r, alter_r) \lhd G_0$. По условию $(r, e, \alpha_r) \in R_{e_0}$, откуда $(u', r, e, \alpha_r) \lhd G_0$.

С л у ч а й 3. Так как $(r', r, alter_r) \notin R_{e_0}$ и $(r', r, alter_r) \in \widehat{R}_{e_{i-1}}$, существует $j \in \{1, \ldots, i-1\}$, такое, что $(r', r, alter_r) \notin \widehat{R}_{e_{j-1}}$ и $(r', r, alter_r) \in \widehat{R}_{e_j}$. Так как $\widehat{op}_i \prec add\_member$, для r', r и e выполнено предположение индукции с количеством

правил вида $grant\_right$ и add_member в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_j)$, меньшим чем N. Тогда по предположению индукции существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и $(u', r, e, \alpha_r) \lhd G_0$. ■

6. Необходимые и достаточные условия предоставления прав доступа

Введём предикат $can\_grant\_right(u, e, \alpha_r, G)$, истинный тогда, когда в результате применения правил субъект-сессией, созданной пользователем u, учётная запись пользователя u получает возможность предоставлять право доступа α_r к сущности e.

Определение 13. Пусть G — состояние системы $\Sigma(G^*, OP)$, $u \in U$, $e \in E$, $\alpha_r \in \in R_r$, $s \notin S$. Определим предикат $can\_grant\_right(u, e, \alpha_r, G)$, истинный тогда и только тогда, когда существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $G_k = G(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$ и $u \in Gr_k(e, \alpha_r)$.

Покажем, что если в результате применения извне правил субъект-сессией, созданной пользователем u, у принципала p появляется право на предоставление права доступа α_r к сущности e, то в начальном состоянии G_0 существует учётная запись пользователя u', имеющая право на предоставление этого права к e, и u имеет возможность выполнения от её имени произвольного SQL-кода.

Утверждение 6. Пусть G_0 — начальное состояние системы $\Sigma(G^*, OP)$, $u \in U$, $e \in E_0$, $\alpha_r \in R_r$, $s \notin S_0$, $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $op_1\langle s\rangle = create\_session(s, u)$, $G_k = G_0(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$, $p \in P$, $p \notin Gr_0(e, \alpha_r)$ и $p \in Gr_k(e, \alpha_r)$. Тогда существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и либо $u' \in Gr_0(e, \alpha_r)$, либо $(u', e, \alpha_r) \blacktriangleleft G_0$.

Доказательство. Пусть $vestige(G_0, op_1\langle s\rangle, \ldots, op_k\langle s\rangle) = (\widehat{op}_1, \ldots, \widehat{op}_m)$.

Так как $p \notin Gr_0(e, \alpha_r)$ и $p \in Gr_k(e, \alpha_r)$, существует $i \in \{1, \ldots, m\}$, такое, что $p \notin \widehat{Gr}_{i-1}(e, \alpha_r)$ и $p \in \widehat{Gr}_i(e, \alpha_r)$. Пусть l — количество правил вида $grant\_right$ и add_member в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_i)$. Доказательство проведём индукцией по l.

Пусть $l = 1$. Обозначим $\widehat{user}_{i-1}(s)$ как u'. В соответствии с утверждением 3 истинен предикат $can\_act\_as(u, u', G_0)$. Заметим, что в соответствии с замечанием 15 в состоянии $\widehat{G}_{i-1}$ выполнены условия применения правила $\widehat{op}_i$. Так как $l = 1$, $\widehat{op}_i$ — единственное правило вида $grant\_right$ и add_member, откуда $\widehat{R}_{e_{i-1}} = R_{e_0}$ и $\widehat{Gr}_{e_{i-1}} = Gr_{e_0}$.

Пусть $\widehat{op}_i \prec grant\_right$, тогда $\widehat{op}_i = grant\_right(s, p, e, \alpha_r, \texttt{yes})$ и $u' \in \widehat{Gr}_{i-1}(e, \alpha_r)$. Так как $\widehat{Gr}_{e_{i-1}} = Gr_{e_0}$, получаем $u' \in Gr_0(e, \alpha_r)$, а значит, выполнено условие утверждения.

Пусть $\widehat{op}_i \prec add\_member$, тогда $p \in U$, $\widehat{op}_i = add\_member(s, r, p)$, где $r \in R$, $(u', r, alter_r) \in \widehat{R}_{e_{i-1}}$ и $r \in \widehat{Gr}_{i-1}(e, \alpha_r)$. Так как $\widehat{R}_{e_{i-1}} = R_{e_0}$ и $\widehat{Gr}_{e_{i-1}} = Gr_{e_0}$, имеем $(u', r, alter_r) \in R_{e_0}$ и $r \in Gr_0(e, \alpha_r)$. Следовательно, $(u', e, \alpha_r) \blacktriangleleft G_0$, а значит, выполнено условие утверждения.

Пусть условия утверждения выполняются для всех $l < N$, где $N > 1$. Покажем, что оно верно и при $l = N$. Обозначим $\widehat{user}_{i-1}(s)$ как u''. В соответствии с утверждением 3 истинен предикат $can\_act\_as(u, u'', G_0)$.

Пусть $\widehat{op}_i \prec grant\_right$, тогда $\widehat{op}_i \prec grant\_right(s, p, e, \alpha_r, \texttt{yes})$ и $u'' \in \widehat{Gr}_{i-1}(e, \alpha_r)$. Возможны два случая: 1) $u'' \in Gr_0(e, \alpha_r)$; 2) $u'' \notin Gr_0(e, \alpha_r)$.

С л у ч а й 1. Если $u'' \in Gr_0(e, \alpha_r)$, то, положив $u' = u''$, видим, что выполнены условия утверждения.

С л у ч а й 2. Так как $u'' \notin Gr_0(e,\alpha_r)$ и $u'' \in \widehat{Gr}_{i-1}(e,\alpha_r)$, существует $j \in \{1, \ldots, i-1\}$, такое, что $u'' \notin \widehat{Gr}_{j-1}(e,\alpha_r)$ и $u'' \in \widehat{Gr}_j(e,\alpha_r)$. Так как $\widehat{op}_i \prec grant\_right$ и $j < i$, в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_j)$ количество правил вида $grant\_right$ и add_member меньше N. Значит, по предположению индукции существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и либо $u' \in Gr_0(e,\alpha_r)$, либо $(u', e, \alpha_r) \blacktriangleleft G_0$.

Пусть $\widehat{op}_i \prec add\_member$, тогда $p \in U$ и $\widehat{op}_i \prec add\_member(s, r, p)$, где $r \in R$, $(u'', r, alter_r) \in \widehat{R}_{e_{i-1}}$ и $r \in \widehat{Gr}_{i-1}(e,\alpha_r)$. Возможны три случая: 1) $(u'', r, alter_r) \in R_{e_0}$ и $r \in Gr_0(e,\alpha_r)$; 2) $r \notin Gr_0(e,\alpha_r)$; 3) $(u'', r, alter_r) \notin R_{e_0}$ и $r \in Gr_0(e,\alpha_r)$.

С л у ч а й 1. Легко заметить, что $(u'', e, \alpha_r) \blacktriangleleft G_0$. Положив $u' = u''$, видим, что выполнены условия утверждения.

С л у ч а й 2. Так как $r \notin Gr_0(e,\alpha_r)$ и $r \in \widehat{Gr}_{i-1}(e,\alpha_r)$, существует $j \in \{1, \ldots, i-1\}$, что $r \notin \widehat{Gr}_{j-1}(e,\alpha_r)$ и $r \in \widehat{Gr}_j(e,\alpha_r)$. Так как $\widehat{op}_i \prec add\_member$ и $j < i$, в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_j)$ количество правил вида $grant\_right$ и add_member меньше N. Значит, по предположению индукции существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и либо $u' \in Gr_0(e,\alpha_r)$, либо $(u', e, \alpha_r) \blacktriangleleft G_0$.

С л у ч а й 3. Так как $r \in Gr_0(e,\alpha_r)$, верно $(r, e, \alpha_r) \in R_{e_0}$. Кроме того, так как $(u'', r, alter_r) \in \widehat{R}_{e_{i-1}}$, в силу монотонности правил преобразования состояний, $(u'', r, alter_r) \in R_{e_k}$. Заметим, что так как $(u'', r, alter_r) \notin R_{e_0}$, $(u'', r, alter_r) \in R_{e_k}$ и $(r, e, \alpha_r) \in R_{e_0}$, выполнены условия утверждения 5. Следовательно, существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и $(u', r, e, \alpha_r) \lhd G_0$. Легко заметить, что ввиду $(u', r, e, \alpha_r) \lhd G_0$ и $r \in Gr_0(e,\alpha_r)$ выполнено $(u', e, \alpha_r) \blacktriangleleft G_0$. ■

Покажем, что справедливы следующие необходимые и достаточные условия истинности предиката can_grant_right.

Теорема 1. Пусть G_0 — начальное состояние системы $\Sigma(G^*, OP)$, $u \in U$, $e \in E_0$, $\alpha_r \in R_r$. Предикат $can\_grant\_right(u, e, \alpha_r, G_0)$ истинен тогда и только тогда, когда существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и выполнено одно из условий:

Условие 1. $u' \in Gr_0(e,\alpha_r)$.

Условие 2. $(u', e, \alpha_r) \blacktriangleleft G_0$.

Доказательство. Обоснуем достаточность условий утверждения.

Пусть выполнено условие 1. Положим $op = grant\_right(s, u, e, \alpha_r, \texttt{yes})$. Так как $u' \in Gr_0(e,\alpha_r)$, то в G_0 выполнены условия применения правила op субъект-сессией, функционирующей от имени учётной записи пользователя u'. Тогда в соответствии с определением 11 из истинности предиката $can\_act\_as(u, u', G_0)$ следует, что истинен предикат $can\_execute\_as(u, u', op, G_0)$. Отсюда по определению 10 существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$ и $s \notin S_0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $vestige(G_0, op_1\langle s\rangle, \ldots, op_k\langle s\rangle) = (\widehat{op}_1, \ldots, \widehat{op}_m)$, и существует $i \in \{2, \ldots, m\}$ такое, что $\widehat{op}_i = op$ и $\widehat{user}_{i-1}(s) = u'$. Заметим, что в соответствии с замечанием 15 последовательность $\widehat{op}_1, \ldots, \widehat{op}_m$ не включает правила, условия применения которых не выполняются, поэтому выполнены условия применения правила $\widehat{op}_i$ в состоянии $\widehat{G}_{i-1}$, а значит, $u \in \widehat{Gr}_i(e,\alpha_r)$. Пусть $G_k = G_0(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$. В силу монотонности правил преобразования состояний $u \in Gr_k(e,\alpha_r)$. Значит, по определению 13 истинен предикат $can\_grant\_right(u, e, \alpha_r, G_0)$.

Пусть выполнено условие 2. Согласно утверждению 2, из истинности предиката $can\_act\_as(u, u', G_0)$ следует истинность предиката $can\_become(u, u', G_0)$. Отсюда, по определению 12, существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$,

где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s,u)$, $G_k = G_0(op_1\langle s\rangle,\ldots,op_k\langle s\rangle)$ и $(u,u',impersonate_r) \in R_{e_k}$. Положим $op_{k+1}\langle s\rangle = switch(s,u')$. Заметим, что в состоянии G_k выполнены условия применения правила $op_{k+1}\langle s\rangle$ и $user_{k+1}(s) = u'$. В соответствии с замечанием 18 $(u',e,\alpha_r) \blacktriangleleft G_{k+1}$. Значит, существуют роли $r_1,\ldots,r_n \in R$, где $n \geqslant 1$, такие, что $(u',r_1,r_2,alter_r) \lessdot G_{k+1}$, для каждого $i = 1,\ldots,n-2$ выполняется $(r_i,r_{i+1},r_{i+2},alter_r) \lessdot G_{k+1}$, и $(r_{n-1},r_n,alter_r) \in R_{e_{k+1}}$, и $r_n \in Gr_{k+1}(e,\alpha_r)$. Положим $op_{k+2}\langle s\rangle = add\_member(s,r_1,u'),\ldots,\ op_{k+n}\langle s\rangle = add\_member(s,r_{n-1},u')$. Пусть $G_{k+n} = G_{k+1}(op_{k+2}\langle s\rangle,\ \ldots,op_{k+n}\langle s\rangle)$. Так как $user_{k+1}(s) = u'$, то, учитывая, что $(u',r_1,r_2,alter_r) \lessdot G_{k+1}$, в соответствии с утверждением 4 $(u',r_2,alter_r) \in R_{e_{k+2}}$. В силу замечания 16, $(r_1,r_2,r_3,\ alter_r) \lessdot G_{k+2}$, откуда $(r_2,r_3,alter_r) \in R_{e_{k+2}}$, а значит, $(u',r_2,r_3,alter_r) \lessdot G_{k+2}$. Проводя аналогичные рассуждения, легко показать, что $(u',r_n,alter_r) \in R_{e_{k+n}}$. Положим $op_{k+n+1}\langle s\rangle = add\_member(s,r_n,u)$. Пусть $G_{k+n+1} = G_{k+n}(op_{k+n+1}\langle s\rangle)$. Заметим, что в состоянии G_{k+n} выполнены условия применения правила, а значит, $r_n \in UA_{k+n+1}(u)$. Так как $r_n \in Gr_{k+1}(e,\alpha_r)$, то в силу монотонности правил преобразования состояний $r_n \in Gr_{k+n+1}(e,\alpha_r)$. Тогда $u \in Gr_{k+n+1}(e,\alpha_r)$, а значит, по определению 13 истинен предикат $can\_grant\_right(u,e,\alpha_r,G_0)$.

Обоснуем необходимость выполнения условий утверждения. Пусть истинен предикат $can\_grant\_right(u,e,\alpha_r,G_0)$, тогда существуют инициированные извне правила $op_1\langle s\rangle,\ldots,op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s,u)$, $G_k = G(op_1\langle s\rangle,\ldots,op_k\langle s\rangle)$ и $u \in Gr_k(e,\alpha_r)$. Заметим, что так как $(u,u,impersonate_r) \in R_{e_0}$, истинен предикат $can\_become(u,u,G_0)$, а значит, согласно утверждению 2, истинен предикат $can\_act\_as(u,u,G_0)$. Если $u \in Gr_0(e,\alpha_r)$, то так как истинен предикат $can\_become(u,u,G_0)$, выполнено условие 1 утверждения. Если $u \notin Gr_0(e,\alpha_r)$, то выполнены условия утверждения 6 и существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u,u',G_0)$ и либо $u' \in Gr_0(e,\alpha_r)$, либо $(u',e,\alpha_r) \blacktriangleleft G_0$. ■

Из утверждения 6 и теоремы 1 очевидна справедливость следующего следствия.

Следствие 1. Пусть G_0 — начальное состояние системы $\Sigma(G^*,OP)$, $u \in U$, $e \in E_0$, $\alpha_r \in R_r$, $s \notin S_0$, $op_1\langle s\rangle,\ldots,op_k\langle s\rangle \in OP_{\text{ext}}$, где $op_1\langle s\rangle = create\_session(s,u)$, $G_k = G_0(op_1\langle s\rangle,\ldots,op_k\langle s\rangle)$, $p \in P$, $p \notin Gr_0(e,\alpha_r)$ и $p \in Gr_k(e,\alpha_r)$. Тогда истинен предикат $can\_grant\_right(u,e,\alpha_r,G_0)$.

7. Необходимые и достаточные условия получения прав доступа

Введём предикат $can\_get\_right(u,e,\alpha_r,G)$, который будет истинен, когда в результате применения правил субъект-сессией, созданной пользователем u, учётная запись пользователя u может получить право доступа α_r к сущности e.

Определение 14. Пусть G — состояние системы $\Sigma(G^*,OP)$, $u \in U$, $e \in E$, $\alpha_r \in R_r$, $s \notin S$ и $(u,e,\alpha_r) \notin R_e$. Определим предикат $can\_get\_right(u,\ e,\alpha_r,G)$, который будет истинным тогда и только тогда, когда существуют инициированные извне правила $op_1\langle s\rangle,\ldots,op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s,u)$, $G_k = G(op_1\langle s\rangle,\ldots,op_k\langle s\rangle)$ и $(u,e,\alpha_r) \in R_{e_k}$.

Обоснуем необходимые и достаточные условия получения учётной записью пользователя прав доступа к сущности в начальном состоянии системы G_0.

Теорема 2. Пусть G_0 — начальное состояние системы $\Sigma(G^*,OP)$, $u \in U$, $e \in E_0$, $\alpha_r \in R_r$ и $(u,e,\alpha_r) \notin R_{e_0}$. Предикат $can\_get\_right(u,e,\alpha_r,G_0)$ истинен тогда и только тогда, когда существуют $u' \in U$, $e' \in E_0$, такие, что предикат $can\_act\_as(u,u',G_0)$ истинен, $e \leqslant e'$ и выполнено одно из следующих условий.

У с л о в и е 1. $u' \in Gr_0(e',\alpha_r)$.

У с л о в и е 2. $(u', e', \alpha_r) \lhd G_0$.

Доказательство. Достаточность.

Пусть выполнено условие 1. Положим $op = grant\_right(s, u, e', \alpha_r, \mathtt{no})$. Так как $u' \in Gr_0(e', \alpha_r)$, в G_0 выполнены условия применения правила op субъект-сессией, функционирующей от имени учётной записи пользователя u'. Тогда в соответствии с определением 11 из истинности предиката $can\_act\_as(u, u', G_0)$ следует, что истинен предикат $can\_execute\_as(u, u', op, G_0)$. Значит, существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $s \notin S_0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $vestige(G_0, op_1\langle s\rangle, \ldots, op_k\langle s\rangle) = (\widehat{op}_1, \ldots, \widehat{op}_m)$ и существует $i \in \{2, \ldots, m\}$, что $\widehat{op}_i = op$ и $\widehat{user}_{i-1}(s) = u'$. Заметим, что в соответствии с замечанием 15 последовательность $\widehat{op}_1, \ldots, \widehat{op}_m$ не включает правила, условия применения которых не выполняются, поэтому выполнены условия применения правила $\widehat{op}_i$ в состоянии $\widehat{G}_{i-1}$, а значит, $(u, e', \alpha_r) \in \widehat{R}_{e_i}$. Пусть $G_k = G_0(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$. В силу монотонности правил преобразования состояний, $(u, e', \alpha_r) \in R_{e_k}$. Отсюда, так как $e \leqslant e'$, в соответствии с замечанием 3 $(u, e, \alpha_r) \in R_{e_k}$. Значит, по определению 14 истинен предикат $can\_get\_right(u, e, \alpha_r, G)$.

Пусть выполнено условие 2. Согласно утверждению 2, из истинности предиката $can\_act\_as(u, u', G_0)$ следует истинность предиката $can\_become(u, u', G_0)$. Значит, по определению 12 существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $k > 0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $G_k = G(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$ и $(u, u', impersonate_r) \in R_{e_k}$. Положим $op_{k+1}\langle s\rangle = switch(s, u')$. Заметим, что в состоянии G_k выполнены условия применения правила $op_{k+1}\langle s\rangle$ и, следовательно, $user_{k+1}(s) = u'$. В соответствии с замечанием 17 $(u', e', \alpha_r) \lhd G_{k+1}$. Значит, существуют роли $r_1, \ldots, r_n \in R$, где $n \geqslant 1$, такие, что $(u', r_1, r_2, alter_r) \lessdot G_{k+1}$, для каждого $i = 1, \ldots, n-2$ выполняется $(r_i, r_{i+1}, r_{i+2}, alter_r) \lessdot G_{k+1}$ и $(r_{n-1}, r_n, e', \alpha_r) \lessdot G_{k+1}$. Положим $op_{k+2}\langle s\rangle = add\_member(s, r_1, u'), \ldots, op_{k+n}\langle s\rangle = add\_member(s, r_{n-1}, u')$. Пусть $G_{k+n} = G_k(op_{k+2}\langle s\rangle, \ldots, op_{k+n}\langle s\rangle)$. Так как $user_{k+1}(s) = u'$, учитывая $(u', r_1, r_2, alter_r) \lessdot G_{k+1}$, в соответствии с утверждением 4 получим $(u', r_2, alter_r) \in R_{e_{k+2}}$. В силу замечания 16 $(r_1, r_2, r_3, alter_r) \lessdot G_{k+2}$, откуда $(r_2, r_3, alter_r) \in R_{e_{k+2}}$, а значит, $(u', r_2, r_3, alter_r) \lessdot G_{k+2}$. Проводя аналогичные рассуждения, легко показать, что $(u', r_n, alter_r) \in R_{e_{k+n}}$. Положим $op_{k+n+1}\langle s\rangle = add\_member(s, r_n, u)$. Пусть $G_{k+n+1} = G_{k+n}(op_{k+n+1}\langle s\rangle)$. Заметим, что в состоянии G_{k+n} выполнены условия применения правила, а значит, $r_n \in UA_{k+n+1}(u)$. Так как $(r_{n-1}, r_n, e', \alpha_r) \lessdot G_{k+1}$, в силу замечания 16 имеем $(r_{n-1}, r_n, e', \alpha_r) \lessdot G_{k+n+1}$, а значит, $(r_n, e', \alpha_r) \in R_{e_{k+n+1}}$, и тогда $(u, e', \alpha_r) \in R_{e_{k+n+1}}$. Отсюда, так как $e \leqslant e'$, в соответствии с замечанием 3 $(u, e, \alpha_r) \in R_{e_{k+n+1}}$. Значит, по определению 14 истинен предикат $can\_get\_right(u, e, \alpha_r, G_0)$.

Необходимость.

Пусть истинен предикат $can\_get\_right(u, e, \alpha_r, G_0)$. Тогда, согласно определению 14, существуют инициированные извне правила $op_1\langle s\rangle, \ldots, op_k\langle s\rangle \in OP_{\text{ext}}$, где $k \geqslant 0$ и $s \notin S_0$, такие, что $op_1\langle s\rangle = create\_session(s, u)$, $G_k = G_0(op_1\langle s\rangle, \ldots, op_k\langle s\rangle)$ и $(u, e, \alpha_r) \in R_{e_k}$. Пусть $vestige(G_0, op_1\langle s\rangle, \ldots, op_k\langle s\rangle) = (\widehat{op}_1, \ldots, \widehat{op}_m)$.

Так как $(u, e, \alpha_r) \notin R_{e_0}$ и $(u, e, \alpha_r) \notin R_{e_k}$, существует $i \in \{1, \ldots, m\}$, что $(u, e, \alpha_r) \notin \widehat{R}_{e_{i-1}}$ и $(u, e, \alpha_r) \notin \widehat{R}_{e_i}$. Пусть l — количество правил вида $grant\_right$ и add_member в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_i)$. Доказательство проведём индукцией по l.

Пусть $l = 1$. Обозначим $\widehat{user}_{i-1}(s)$ как u'. В соответствии с утверждением 3 истинен предикат $can\_act\_as(u, u', G_0)$. Заметим, что в соответствии с замечанием 15 в состо-

янии $\widehat{G}_{i-1}$ выполнены условия применения правила $\widehat{op}_i$. Так как $l = 1$, $\widehat{op}_i$ — единственное правило вида $grant\_right$ и add_member, откуда $\widehat{R}_{e_{i-1}} = R_{e_0}$ и $\widehat{Gr}_{e_{i-1}} = Gr_{e_0}$.

Пусть $\widehat{op}_i \prec grant\_right$, тогда $\widehat{op}_i = grant\_right(s, \ldots, e', \alpha_r, \ldots)$, $e' \in \widehat{E}_{i-1}$, $e \leqslant e'$ и $u' \in \widehat{Gr}_{i-1}(e', \alpha_r)$. В системе отсутствуют правила, приводящие к изменению родительских контейнеров сущностей, поэтому $e' \in E_0$. Так как $\widehat{Gr}_{e_{i-1}} = Gr_{e_0}$, выполняется $u' \in Gr_0(e', \alpha_r)$, а значит, выполнено условие 1 утверждения.

Если $\widehat{op}_i \prec add\_member$, то $\widehat{op}_i = add\_member(s, r, u)$, $r \in R$, $(u', r, alter_r) \in \widehat{R}_{e_{i-1}}$ и $(r, e', \alpha_r) \in \widehat{R}_{e_{i-1}}$, где $e' \in \widehat{E}_{i-1}$ и $e \leqslant e'$. В системе отсутствуют правила, приводящие к изменению родительских контейнеров сущностей, поэтому $e' \in E_0$. Так как $\widehat{R}_{e_{i-1}} = R_{e_0}$, имеем $(u', r, alter_r) \in R_{e_0}$ и $(r, e', \alpha_r) \in R_{e_0}$. Следовательно, $(u', e', \alpha_r) \lhd G_0$, а значит, выполнено условие 2 утверждения.

Пусть условия утверждения выполняются для всех $l < N$, где $N > 1$. Покажем, что они верны и при $l = N$. Обозначим $\widehat{user}_{i-1}(s)$ как u''. В соответствии с утверждением 3 истинен предикат $can\_act\_as(u, u'', G_0)$.

Пусть $\widehat{op}_i \prec grant\_right$, тогда $\widehat{op}_i \prec grant\_right(s, \ldots, e', \alpha_r, \ldots)$, $e' \in \widehat{E}_{i-1}$, $e \leqslant e'$ и $u'' \in \widehat{Gr}_{i-1}(e', \alpha_r)$. В системе отсутствуют правила, приводящие к изменению родительских контейнеров сущностей, поэтому $e' \in E_0$. Возможны следующие два случая: 1) $u'' \in Gr_0(e', \alpha_r)$; 2) $u'' \notin Gr_0(e', \alpha_r)$.

С л у ч а й 1. Если $u'' \in Gr_0(e', \alpha_r)$, то, положив $u' = u''$, видим, что выполнено условие 1 утверждения.

С л у ч а й 2. Так как $u'' \in \widehat{Gr}_{i-1}(e', \alpha_r)$, в силу монотонности правил преобразования состояний $u'' \in Gr_k(e', \alpha_r)$. Так как $u'' \notin Gr_0(e', \alpha_r)$ и $u'' \in Gr_k(e', \alpha_r)$, в соответствии с утверждением 6 существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и либо $u' \in Gr_0(e', \alpha_r)$, либо $(u', e', \alpha_r) \blacktriangleleft G_0$. Учитывая, что в случае $(u', e', \alpha_r) \blacktriangleleft G_0$ также выполняется $(u', e', \alpha_r) \lhd G_0$, видим, что выполнены условия 1 и 2 утверждения.

Пусть $\widehat{op}_i \prec add\_member$, тогда $\widehat{op}_i \prec add\_member(s, r, u)$, где $r \in R$, $(u'', r, alter_r) \in \widehat{R}_{e_{i-1}}$ и $(r, e', \alpha_r) \in \widehat{R}_{e_{i-1}}$, где $e' \in \widehat{E}_{i-1}$ и $e \leqslant e'$. В системе отсутствуют правила, приводящие к изменению родительских контейнеров сущностей, поэтому $e' \in E_0$. Возможны три случая: 1) $(u'', r, alter_r) \in R_{e_0}$ и $(r, e', \alpha_r) \in R_{e_0}$; 2) $(r, e', \alpha_r) \notin R_{e_0}$; 3) $(u'', r, alter_r) \notin R_{e_0}$ и $(r, e', \alpha_r) \in R_{e_0}$.

С л у ч а й 1. Легко заметить, что $(u'', e', \alpha_r) \lhd G_0$. Положив $u' = u''$, видим, что выполнено условие 2 утверждения.

С л у ч а й 2. Так как $(r, e', \alpha_r) \notin R_{e_0}$ и $(r, e', \alpha_r) \in \widehat{R}_{e_{i-1}}$, существует $j \in \{1, \ldots, i-1\}$, такое, что $(r, e', \alpha_r) \notin \widehat{R}_{e_{j-1}}$ и $(r, e', \alpha_r) \in \widehat{R}_{e_j}$. Так как $\widehat{op}_i \prec add\_member$ и $j < i$, в последовательности $(\widehat{op}_1, \ldots, \widehat{op}_j)$ количество правил вида $grant\_right$ и add_member меньше N. Значит, по предположению индукции существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и либо $u' \in Gr_0(e', \alpha_r)$, либо $(u', e', \alpha_r) \lhd G_0$.

С л у ч а й 3. Так как $(u'', r, alter_r) \in \widehat{R}_{e_{i-1}}$, в силу монотонности правил преобразования состояний получим $(u'', r, alter_r) \in R_{e_k}$. Заметим, что ввиду $(r, e', \alpha_r) \in R_{e_0}$, $(u'', r, alter_r) \notin R_{e_0}$ и $(u'', r, alter_r) \in R_{e_k}$, в соответствии с утверждением 5 существует $u' \in U$, такое, что истинен предикат $can\_act\_as(u, u', G_0)$ и $(u', r, e', \alpha_r) \lhd G_0$. Отсюда следует, что $(u', e', \alpha_r) \lhd G_0$, а значит, выполнено условие 2 утверждения. ■

Заключение

Таким образом, для обеспечения возможности теоретического анализа безопасности СУБД построена формальная модель, в которую добавлены новые элементы, поз-

воляющие более полно учесть особенности управления доступом в MS SQL Server. В рамках модели определены необходимые и достаточные условия предоставления и получения прав доступа. Обоснована эквивалентность возможности выполнения SQL-кода от имени заданной учётной записи и получения к ней права олицетворения. В дальнейшем планируется определение необходимых и достаточных условий выполнения SQL-кода от имени заданной учётной записи, а также проведение анализа условий, при которых пользователь может активизировать процедуру или триггер от имени учётной записи заданного пользователя.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Ролевая ДП-модель управления доступом и информационными потоками в операционных системах семейства Linux // Прикладная дискретная математика. 2012. № 1(15). С. 69–90.
2. *Колегов Д. Н.* Дискреционная модель безопасности управления доступом и информационными потоками в компьютерных системах с функционально или параметрически ассоциированными сущностями: дис. ... канд. техн. наук. Томск, 2009.
3. *Смольянинов В. Ю.* Правила преобразования состояний СУБД ДП-модели // Прикладная дискретная математика. 2013. № 1(18). С. 50–68.
4. *Bruchez R.* Microsoft SQL Server 2012 Security Cookbook. Pact Publishing, 2012. 307 с.

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

DOI 10.17223/20710410/24/6

УДК 519.172.1+517.553

АНАЛИТИЧЕСКАЯ СЛОЖНОСТЬ КЛАСТЕРНЫХ ДЕРЕВЬЕВ[1]

А. И. Нормов, Т. М. Садыков

Российский экономический университет им. Г. В. Плеханова, г. Москва, Россия

E-mail: SadykovTM@rsute.ru

Вводится понятие аналитической сложности двоичного дерева — его неотрицательной целочисленной характеристики, отражающей комбинаторную структуру дерева и свойства его «наиболее экономичного» аналитического представления. Изучаются свойства аналитической сложности дерева и предлагается алгоритм её вычисления. Развитая в работе методика применяется для сравнения кластерных деревьев.

Ключевые слова: *кластерный анализ, двоичное дерево, аналитическая сложность.*

Введение

Анализ данных большого объёма требует, как правило, их систематизации и классификации. Формируемые классы (кластеры) объектов многочисленной генеральной совокупности данных, подлежащей изучению, могут сравниваться друг с другом путём их отождествления с элементами некоторой комбинаторной структуры (графа, дерева, матрицы и пр.). Эта группа методов анализа данных включает в себя, в числе прочих, дискриминантный анализ, кластеризацию и факторный анализ данных [1]. При этом возникает задача сравнения свойств и характеристик комбинаторных структур, сопоставляемых большим совокупностям исследуемых данных при помощи перечисленных методов. В частности, актуальной является задача сравнения структур кластерных деревьев, которые строятся при помощи кластерного анализа [1, 2].

В настоящей работе вводится понятие аналитической сложности двоичного дерева. Аналитическая сложность функции $f(x, y)$ двух переменных — это её числовая характеристика со значениями в $\mathbb{N} \cup \{\infty\}$, содержащая информацию о возможных представлениях $f(x, y)$ в виде суперпозиций функций одной переменной и фиксированной функции $s(x, y)$ двух переменных, в качестве которой всюду в дальнейшем используется сумма аргументов: $s(x, y) = x + y$ [3]. В работе изучаются свойства аналитической сложности двоичных деревьев и предлагается алгоритм её вычисления. Под аналитической сложностью двоичного дерева понимается аналитическая сложность функции двух переменных, сопоставленной этому дереву некоторым каноническим образом. Полученные результаты применяются для анализа и сравнения структур кластерных деревьев.

[1]Второй автор поддержан грантами РФФИ №13-01-12417-офи-м2 и 14-01-00544-а и грантом Правительства РФ для проведения исследований под руководством ведущих ученых в Сибирском федеральном университете (договор №14.Y26.31.0006).

1. Предварительные сведения

Следуя принятой в теории графов терминологии [4], под *степенью* вершины графа будем понимать количество рёбер, инцидентных этой вершине. *Степенью графа* (в частности, дерева) называется наибольшая из степеней его вершин. Дерево называется *двоичным*, если это упорядоченное дерево, каждый узел которого имеет не более двух потомков. Под числом *уровней* в дереве с заданным корнем будем понимать количество рёбер, связывающих корень с максимально удалённой от него вершиной.

Типичной задачей анализа данных (вычисляемых или полученных в результате измерений) является задача построения так называемого кластерного дерева по набору данных об изучаемой совокупности объектов и по заданной метрике (расстоянию) в пространстве их параметров [1]. Алгоритмы кластеризации широко известны и реализованы в многочисленных компьютерных системах статистического анализа данных (например, в системе *Statistica 12*). Если измерения выполнены с достаточно высокой точностью, а используемая метрика позволяет различать все параметры рассматриваемых объектов, то построенное по этим данным кластерное дерево будет двоичным. Действительно, повышение точности измерений соответствует малому возмущению набора данных, содержащего информацию об изучаемой совокупности объектов, а значит, и расстояний между ними в смысле используемой метрики. По теореме Сарда [5, § 2] множество конфигураций, три или более элемента которых лежат на одинаковом расстоянии друг от друга (в смысле произвольной фиксированной достаточно гладкой функции расстояния), имеет нулевую меру. Это и означает, что соответствующее данной конфигурации кластерное дерево будет двоичным. Преобразование кластерного дерева, соответствующее повышению точности измерений и отображающее вершину высокой степени в набор вершин степени 3, иллюстрируется рис. 1.

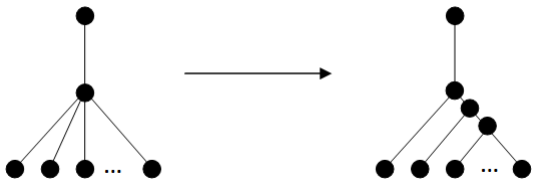

Рис. 1. Преобразование дерева степени $n > 3$ с двумя уровнями в двоичное дерево с $n - 1$ уровнями

Результат кластеризации существенным образом зависит от метрики (т. е. расстояния в пространстве параметров изучаемых объектов), с использованием которой сравниваются объекты кластеризуемого множества. При этом от выбора метрики зависит как геометрическая, так и комбинаторная структура кластерного дерева. Характер этой зависимости иллюстрируется следующим примером.

Пример 1. Простейшим случаем, в котором при использовании разных метрик получаются двоичные деревья с разной комбинаторной структурой, является случай, когда кластеризуемое множество состоит из четырёх элементов, каждый из которых характеризуется двумя числовыми параметрами. Рассмотрим, например, множество точек плоскости $M = \{(0, 0), (4, 0), (6, 3), (8, 3)\}$ (рис. 2).

Кластеризация множества M с использованием стандартного евклидова расстояния и метрики Манхэттена (в которой расстояние между точками плоскости с евклидовыми координатами (x_1, x_2) и (y_1, y_2) равно $|x_1 - y_1| + |x_2 - y_2|$) даёт кластерные деревья, изображённые на рис. 3.

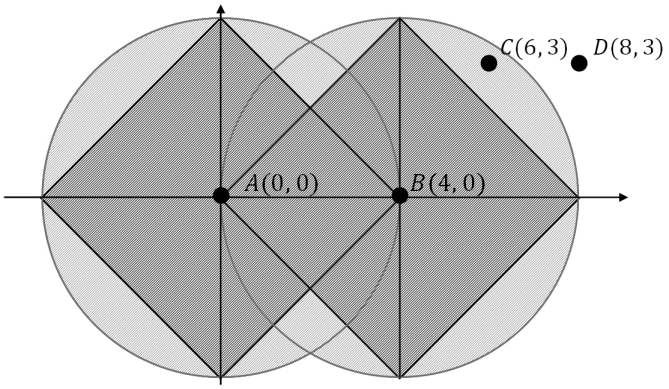

Рис. 2. Конфигурация точек плоскости, дающая кластерные деревья с различной комбинаторной структурой в зависимости от выбора метрики

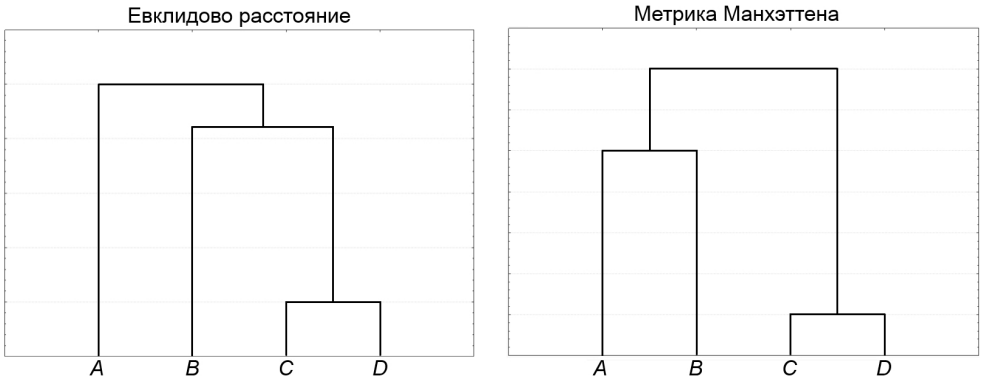

Рис. 3. Кластерные деревья, построенные с использованием разных метрик по одному и тому же набору исходных данных

Причина зависимости комбинаторной структуры кластерного дерева от выбора метрики в пространстве параметров иллюстрируется рис. 2. В то время как в метрике Манхэттена точки множества M группируются в пары похожих элементов, в евклидовой метрике каждая последующая точка лежит на всё большем расстоянии от кластера, образованного всеми предыдущими точками.

Всюду в дальнейшем предполагается, что сравнение объектов кластеризуемого множества производится с использованием произвольной, но фиксированной метрики в пространстве их параметров.

2. Аналитическая сложность функции двух переменных

Аналитическая сложность голоморфной функции $f(x, y)$ двух комплексных переменных — это её числовая характеристика со значениями в $\mathbb{N} \cup \{\infty\}$, содержащая информацию о «наиболее экономичном» представлении $f(x, y)$ в виде суперпозиций функций одной комплексной переменной и фиксированной функции $s(x, y)$ двух переменных. В качестве последней всюду в дальнейшем используется сумма аргументов: $s(x, y) = x + y$. В работе [3] предложена иерархия классов аналитической сложности $Cl_0 \subset Cl_1 \subset \ldots \subset Cl_n \subset \ldots \subset Cl_\infty$ для функций двух комплексных переменных. Согласно этой иерархии, тривиальный класс функций нулевой аналитической сложности образуют функции, которые зависят лишь от одной из переменных или являются постоянными. В качестве «материала» для построения функций более высокой (т. е. ненулевой) аналитической сложности используются функции нулевого класса и функ-

ция $s(x,y)$, всевозможные конечные композиции которых и образуют множество функций конечной аналитической сложности. Стратификация этого множества осуществляется путём учёта числа применений функции $s(x,y)$, необходимого для образования нужной композиции. Будем исходить из следующего индуктивного определения.

Определение 1 [3]. Нулевой класс аналитической сложности Cl_0 состоит из функций, зависящих не более чем от одной из своих переменных. Функция $f(x,y)$ лежит в множестве Cl_n функций аналитической сложности $n>0$, если для некоторой точки $(x_0,y_0)\in\mathbb{C}^2$ и некоторого ростка $\mathfrak{f}(x,y)\in\mathcal{O}(U(x_0,y_0))$ этой функции, голоморфного в окрестности этой точки, найдутся ростки голоморфных функций $a,b\in Cl_{n-1}$ и $c\in Cl_0$, такие, что $\mathfrak{f}(x,y)=c(a(x,y)+b(x,y))$. Если такое представление невозможно ни для какого конечного n, то функция f имеет бесконечную аналитическую сложность.

Данное определение является корректным в том смысле, что аналитическая сложность голоморфной функции двух переменных не зависит от ростка функции, для которого ищется наиболее экономичное представление из определения 1. Это следует из того, что для каждого класса аналитической сложности существует дифференциальный критерий принадлежности функции этому классу, отождествляющий множество функций заданной аналитической сложности с множеством всех решений системы дифференциальных многочленов с целыми коэффициентами. В силу принципа консерватизма аналитическое продолжение решения такой системы уравнений вдоль любого пути также должно быть её решением и, следовательно, иметь ту же аналитическую сложность. Аналитическую функцию $f(x,y)$, $x,y\in\mathbb{C}$, назовём функцией аналитической сложности n, если $f(x,y)\in Cl_n$, но $f(x,y)\notin Cl_{n-1}$. Всюду в дальнейшем под сложностью функции двух переменных будем понимать её аналитическую сложность.

Для Cl_1 известен дифференциальный критерий принадлежности функции этому классу [3]: аналитическая функция $f(x,y)$ лежит в Cl_1 в том и только в том случае, когда

$$f'_x(f'_y)^2f'''_{xxy}-(f'_x)^2f'_yf'''_{xxy}+f''_{xy}(f'_x)^2f''_{yy}-f''_{xy}(f'_y)^2f''_{xx}=0. \tag{1}$$

Получить это уравнение можно следующим образом: продифференцируем функцию $f(x,y)=c\left(a\left(x\right)+b\left(y\right)\right)$ по переменным x и y:

$$f'_x=a'\left(x\right)c'\left(a\left(x\right)+b\left(y\right)\right),\quad f'_y=b'\left(y\right)c'\left(a\left(x\right)+b\left(y\right)\right).$$

Отсюда видно, что функция $\ln\left(f'_x/f'_y\right)$ представима в виде суммы функции переменной x и функции переменной y, а значит, имеет место равенство $\left(\ln\left(f'_x/f'_y\right)\right)''_{xy}=0$. Вычисляя производные в левой части, заключаем, что числитель полученного выражения равен левой части уравнения (1). Подобное уравнение существует для любого Cl_n, но уже для Cl_2 его вычисление является нетривиальной (и на данный момент нерешённой) задачей [6].

Пример 2. Приведём важные для дальнейшего примеры функций двух переменных конечной аналитической сложности.

а) Функция x^y лежит в первом классе аналитической сложности. Это следует из представления $x^y=\exp\left(\exp\left(\ln\ln x+\ln y\right)\right)$ и того обстоятельства, что данная функция существенно зависит от обеих переменных. Другой способ убедиться в том, что $x^y\in Cl_1$, состоит в применении дифференциального критерия (1): подстановка x^y вместо f даёт тождественный нуль.

б) Согласно «а» и определению 1, для произвольной функции одной переменной $h(t)$ композиция $h(x^y)$ также лежит в Cl_1.

в) Аналогичные рассуждения показывают, что аналитическая сложность функции $x^{(y^x)}$ равна 2. Действительно, применение дифференциального критерия (1) даёт $x^{(y^x)} \notin Cl_1$. Представление, аналогичное приведённому в примере «а», позволяет убедиться в том, что аналитическая сложность функции $x^{(y^x)}$ не превосходит двух.

3. Аналитическая сложность двоичного дерева и алгоритм её вычисления

Опишем процедуру сопоставления двоичному дереву неотрицательного целого числа — его аналитической сложности. Эта величина является классификационным параметром кластерного дерева (ср. с [7]) и играет в дальнейшем важную роль. С этой целью заданному двоичному дереву T некоторым каноническим способом сопоставляется аналитическая функция двух переменных $E_T(x, y)$. Аналитической сложностью двоичного дерева T по определению называется аналитическая сложность функции $E_T(x, y)$. Как мы увидим, эта величина корректно определена (и при этом конечна), т. е. зависит лишь от самого двоичного дерева T. Тем самым любому двоичному дереву сопоставлена целочисленная неотрицательная характеристика, которая используется, в частности, для классификации кластерных деревьев.

Будем рассматривать аналитические функции с двумя независимыми переменными x, y. Отметим, что если функция E допускает представление в виде $E = \tilde{C}(A_{m-1}(x, y) + B_{n-1}(x, y))$, где A_{m-1} — функция сложности $m - 1$, B_{n-1} — функция сложности $n - 1$, а $\tilde{C}$ — произвольная непостоянная функция одной переменной, то сложность E равна $\max(m, n)$.

Алгоритм 1. *Построение аналитической функции конечной сложности по двоичному дереву*

Зафиксируем вершину двоичного дерева степени не выше 2, которую будем считать корнем. Для сопоставления дереву аналитической функции двух переменных присвоим каждой вершине значение в зависимости от её степени в соответствии со следующими правилами:

1) вершинам степени 1, за исключением корня дерева, сопоставляем последовательно значение x или y, чередуя эти переменные;
2) вершинам степени 2, за исключением корня, сопоставляем произвольную функцию h одной переменной;
3) вершинам степени 3 сопоставляем функцию возведения в степень «^»;
4) корню дерева степени 1 сопоставляем произвольную функцию h, а корню степени 2 — функцию «^».

После сопоставления значений вершинам необходимо выполнить «сборку» функции. Она начинается с корня дерева и выполняется следующим образом:

1) если корню сопоставлено значение «^», то ему соответствует функция $E_1^{E_2}$, где E_1, E_2 — функции левого и правого потомков соответственно;
2) если корню присвоено значение h, то ему соответствует функция $h(E)$, где E — функция потомка;
3) далее по аналогии применяем последовательно данную операцию ко всем поддеревьям двоичного дерева; листьям соответствуют функции одной переменной x и y.

На рис. 4 приведён пример построения функции по дереву.

Отметим, что алгоритм 1 обращает команду `TreeForm[E[T]]` в среде компьютерной алгебры *Mathematica 9.0* [8] и позволяет задать комбинаторную структуру двоичного дерева при помощи формулы от двух переменных.

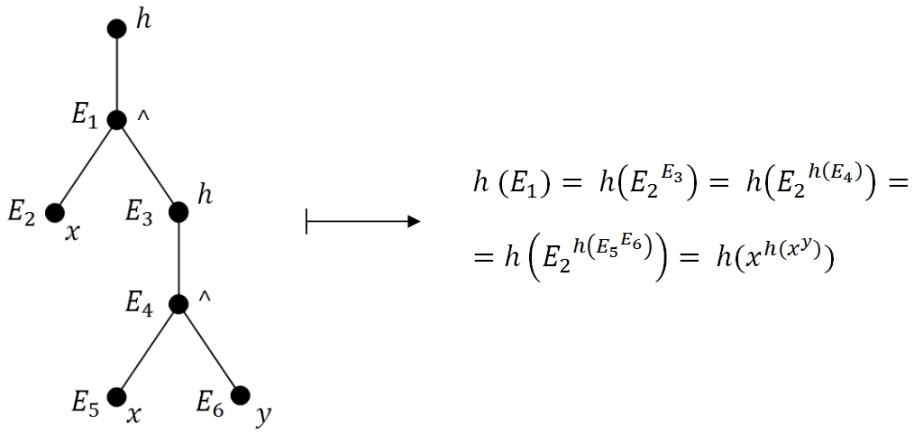

Рис. 4. Построение функции $E(T)$ конечной аналитической сложности по двоичному дереву T

Определение 2. *Аналитической сложностью $C(T)$ двоичного дерева T* называется аналитическая сложность функции, которая кодирует это дерево в смысле алгоритма 1.

Пример 3. Для изображённых на рис. 5 двоичных деревьев соответствующие им функции двух переменных и их аналитическая сложность имеют следующий вид: $E(T_1) = h(h(x^y))$, $C(T_1) = 1$; $E(T_2) = x^{(y^x)}$, $C(T_2) = 2$; $E(T_3) = h(h(h(x)))$, $C(T_3) = 0$; $E(T_4) = h(x^{h(x^y)})$, $C(T_4) = 2$; $E(T_5) = h\left((x^y)^{((x^y)^x)}\right)$, $C(T_5) = 3$.

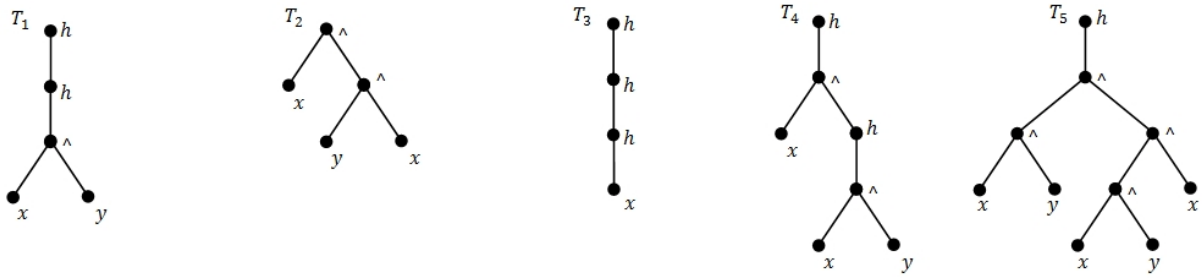

Рис. 5. Примеры двоичных деревьев различной аналитической сложности

Определение 3. Под *редукцией $R(T)$ двоичного дерева T* будем понимать исключение из дерева отличных от корня вершин степени 2 и корня дерева в случае, если он имеет степень 1.

Например, функции, сопоставляемые деревьям на рис. 6 при помощи алгоритма 1, имеют следующий вид: $E(T) = h\left(x^{h(y^x)}\right)$, $E(R(T)) = x^{(y^x)}$. Из рассуждений, приведённых в примере 2, следует, что деревья T и $R(T)$ имеют одинаковую аналитическую сложность (равную двум).

Основной результат работы состоит в следующем.

Теорема 1. Аналитическая сложность двоичного дерева T равна числу уровней в дереве $R(T)$.

Доказательство. Дерево $R(T)$ может быть тривиальным (т. е. содержать единственную вершину) лишь в случае, когда исходное дерево T является линейным (т. е. не содержит узлов степени выше двух). Линейному дереву алгоритм 1 сопоставляет

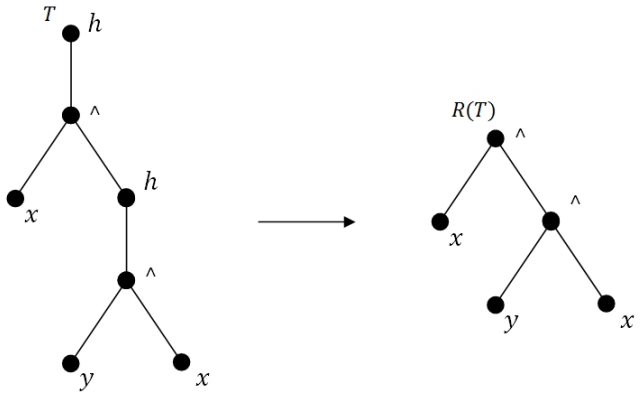

Рис. 6. Редукция $R(T)$ двоичного дерева T

функцию одной переменной $h(h(\ldots(x)\ldots))$, которая в соответствии с определением 1 имеет нулевую аналитическую сложность. Таким образом, утверждение теоремы имеет место в случае, когда редукция даёт тривиальное дерево.

Пусть редукция $R(T)$ нетривиальна. Если исходное дерево T содержит один уровень, то T есть двоичное дерево с тремя вершинами и двумя рёбрами. Такому дереву алгоритм 1 сопоставляет функцию двух переменных x^y, которая имеет аналитическую сложность равную 1.

Если редукция дерева T нетривиальна, то в соответствии с алгоритмом 1 степень корня редуцированного дерева $R(T)$ равна 2. Линейная же часть дерева T от корня до первого узла степени 2 не влияет ни на результат редукции, ни на аналитическую сложность T. Таким образом, можем без ограничения общности считать, что корень дерева T имеет степень 2.

Предположим, что утверждение теоремы имеет место для всех двоичных деревьев, таких, что соответствующие им редуцированные деревья содержат не более k уровней. Обозначим через T_1 и T_2 поддеревья исходного дерева T, смежные с его корнем. По предположению, аналитическая сложность каждого из деревьев T_i равна числу уровней в $R(T_i)$. Обозначим через $L(T)$ число уровней в дереве T, а через E_i — функцию, сопоставляемую поддереву T_i при помощи алгоритма 1. Из определения редуцированного дерева следует равенство

$$L(R(T)) = \max(L(R(T_1)), L(R(T_2))) + 1. \tag{2}$$

С другой стороны, в соответствии с определением 1 аналитической сложности функции двух переменных

$$C(T) = C(E_1^{E_2}) = (\exp(\exp(\ln\ln E_1 + \ln E_2))) = \max(C(E_1), C(E_2)) + 1. \tag{3}$$

Согласно индуктивной гипотезе, $C(E_i) = L(R(T_i))$, $i = 1, 2$. Это означает, что выражения (2) и (3) совпадают. ■

Отметим некоторые следствия доказанной теоремы.

Следствие 1. Аналитическая сложность дерева с n вершинами не превышает числа $\left\lfloor \dfrac{n-1}{2} \right\rfloor$, где $\lfloor x \rfloor$ — наибольшее целое число, не превосходящее x.

Доказательство. На рис. 7 изображено одно из двоичных деревьев максимальной аналитической сложности при заданном количестве n вершин.

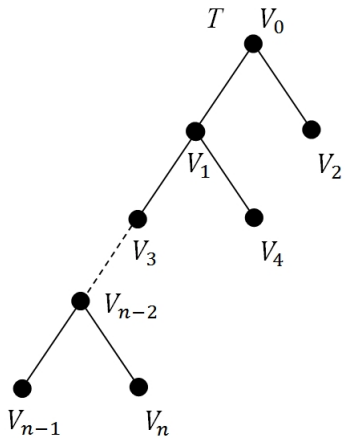

Рис. 7. Двоичное дерево максимальной аналитической сложности при заданном числе вершин

Поскольку редукция дерева на рис. 7 совпадает с ним самим, по теореме 1 его аналитическая сложность равна числу уровней в нём. Следовательно, для дерева на рис. 7 имеет место неравенство $2C(T) + 1 \leqslant n$. Если исходное дерево допускает нетривиальную редукцию или содержит меньшее количество уровней при фиксированном числе вершин, его аналитическая сложность уменьшается. ■

Следствие 2. Аналитическая сложность дерева T, полученного путём отождествления корня двоичного дерева T_1 с вершиной степени 1 двоичного дерева T_2, не превосходит суммы их аналитических сложностей $C(T_1) + C(T_2)$.

Доказательство. Действительно, операция редукции может быть применена к каждому из деревьев T_1 и T_2 по отдельности, а значит, согласно теореме 1,

$$C(T) = L(R(T)) \leqslant L(R(T_1)) + L(R(T_2)) = C(T_1) + C(T_2).$$

Равенство здесь имеет место в том и только в том случае, когда корень дерева T_1 отождествляется с вершиной степени 1 дерева T_2, находящейся на уровне с максимальным номером. ■

В заключение отметим нижнюю границу скорости роста аналитической сложности двоичного дерева как функции количества его вершин.

Следствие 3. Пусть T — максимально ветвящееся двоичное дерево с k уровнями, т. е. такое, что каждая его вершина, за исключением корня, имеет степень либо 1, либо 3 и все вершины, за исключением уровня $k - 1$, имеют одинаковую степень. Аналитическая сложность дерева T равна $\lfloor \log_2(n + 1) - 1 \rfloor$.

Заключение

Результаты работы дают эффективную методику сравнения кластерных деревьев, полученных на основе реальных данных, и использованных для этого параметров объектов кластеризуемого множества. Малая аналитическая сложность дерева даёт возможность сделать вывод о высокой регулярности его комбинаторной структуры. Близкая аналитическая сложность двух кластерных деревьев, построенных по одному и тому же набору данных для разных метрик, может указывать на сопоставимую эффективность этих метрик для описания исследуемой совокупности объектов (при дополнительном исследовании решающих правил, основанных на этих метриках).

Наконец, близость аналитической сложности кластерного дерева к одной из оценок, данных в следствиях 1 и 3, может указывать либо на недостаточность (для оценки из следствия 1), либо на избыточность (для оценки из следствия 3) множества параметров, учитываемых в метрике, используемой для кластеризации.

Авторы признательны В. К. Белошапке за плодотворные обсуждения вопросов аналитической сложности.

ЛИТЕРАТУРА

1. *Дюран Б., Оделл П.* Кластерный анализ. М.: Статистика, 1977. 128 с.
2. *Миркин Б. Г.* Методы кластерного анализа для поддержки принятия решений. М.: Изд. дом Национального исследовательского университета «Высшая школа экономики», 2011. 84 с.
3. *Белошапка В. К.* Об аналитической сложности функций двух переменных // Российский журнал математической физики. 2007. Т. 14. № 3. С. 243–249.
4. *Зыков А. П.* Основы теории графов. М.: Наука, 1987. 383 с.
5. *Арнольд В. И., Варченко А. Н., Гусейн-Заде С. М.* Особенности дифференцируемых отображений. Т. 1. М.: Наука, 1984. 293 с.
6. *Красиков В. А., Садыков Т.М.* Об аналитической сложности дискриминантов // Труды Математического института им. В. А. Стеклова. 2012. Т. 279. С. 86–101.
7. *Чикалов И. В.* Алгоритм построения деревьев решений с минимальным суммарным весом вершин // Вестник ННГУ. Математическое моделирование и оптимальное управление. 2000. Т. 11. № 2. С. 200–204.
8. *Морозов А. А., Таранчук В. Б.* Программирование задач численного анализа в системе Mathematica. Минск: БГПУ, 2005. 145 с.

DOI 10.17223/20710410/24/7
УДК 519.6

ОЦЕНКА ЭКСПОНЕНТА НЕКОТОРЫХ ГРАФОВ С ПОМОЩЬЮ ЧИСЕЛ ФРОБЕНИУСА ДЛЯ ТРЁХ АРГУМЕНТОВ

В. М. Фомичев

Финансовый университет при Правительстве Российской Федерации, Национальный исследовательский ядерный университет «МИФИ», г. Москва, Россия

E-mail: fomichev@nm.ru

Выведена формула чисел Фробениуса для трёх аргументов и с их помощью получена оценка экспонента сильносвязного n-вершинного орграфа при $n > 2$, в котором имеются три дуги вида (i, r), (r, j), (i, j), где $i, r, j \in \{1, \ldots, n\}$. Показано, что во многих случаях данная оценка экспонента существенно лучше уже известных оценок.

Ключевые слова: *число Фробениуса, порождённая множеством чисел аддитивная полугруппа, экспонент графа.*

Введение

Основные обозначения:

— $\mathbb{N}$ — множество натуральных чисел, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$;
— $a \mid b$ и $a \nmid b$ — «a делит b» и «a не делит b» соответственно, $a, b \in \mathbb{N}$;
— $d = (\theta, \lambda)$ — наибольший общий делитель натуральных чисел θ, λ;
— $\langle \theta, \lambda, \ldots \rangle$ — аддитивная полугруппа, порождённая числами $\theta, \lambda, \ldots$;
— если $A = \{a_1, \ldots, a_k\} \subseteq \mathbb{N}$, $b \in \mathbb{N}_0$, то $bA = \{ba_1, \ldots, ba_k\}$, $b \pm A = \{b \pm a_1, \ldots, b \pm a_k\}$;
— $C(\theta, \lambda, \ldots) = d\mathbb{N} \setminus \langle \theta, \lambda, \ldots \rangle$, $z = \max C(\theta, \lambda)$;
— $\overline{\langle \theta, \lambda \rangle}$ — подмножество полугруппы $\langle \theta, \lambda \rangle$, состоящее из чисел, меньших z;
— $g(\theta, \lambda, \ldots)$ — число Фробениуса для аргументов $\theta, \lambda, \ldots$

Пусть $1 < \theta < \lambda < l$ и числа θ, λ, l взаимно простые.

Функция Фробениуса $g(\theta, \lambda, l)$ для трёх аргументов определена как $\max C(\theta, \lambda, l)$, то есть как наибольшее натуральное число, не представимое в виде линейной комбинации чисел θ, λ, l с неотрицательными целыми коэффициентами:

$$g(\theta, \lambda, l) = \max\{t \in \mathbb{N} : t \neq c_1\theta + c_2\lambda + c_3 l\}.$$

Задача определения $g(\theta, \lambda, l)$ известна как диофантова проблема Фробениуса для трёх аргументов, задача определения множества $C(\theta, \lambda, l)$ — как расширенная проблема Фробениуса для трёх аргументов. Обзор основных результатов по проблеме Фробениуса для различного числа аргументов можно найти в [1]. Здесь отметим лишь те факты, которые наиболее важны для изложения результатов данной работы.

Для двух взаимно простых чисел θ, λ получение формулы числа Фробениуса, равного $\max C(\theta, \lambda)$, и описание множества $C(\theta, \lambda)$ относится ещё к 1884 г. [2]:

$$g(\theta, \lambda) = \theta\lambda - \theta - \lambda, \tag{1}$$

$$C(\theta, \lambda) = \{g(\theta, \lambda) - \overline{\langle \theta, \lambda \rangle}\}. \tag{2}$$

Из (2) следует, что

$$|\overline{\langle\theta,\lambda\rangle}| = |C(\theta,\lambda)| = (\theta-1)(\lambda-1)/2 = (g(\theta,\lambda)+1)/2. \tag{3}$$

Если $d>1$, то из (1) следует, что число $z=\max C(\theta,\lambda)$ определено формулой

$$z = \theta\lambda/d - \theta - \lambda. \tag{4}$$

Формула числа Фробениуса от трёх и более аргументов известна лишь для частных случаев. В [3] доказано, что не существует конечного числа полиномов, позволяющих выразить через них в общем случае число Фробениуса $g(\theta,\lambda,l)$ от трёх аргументов с помощью разбиения области определения.

Более продуктивным стал алгоритмический подход к определению числа Фробениуса $g(a_1,\ldots,a_k)$ от k аргументов. Построены алгоритмы при $k=3$ (O. Rodseth, J. Davison и др. [1]); при $k>3$ [4, 5] оценка сложности алгоритма определения $g(a_1,\ldots,a_k)$ снижена до величины порядка $\mathrm{O}(ka_1)$ операций. Эта оценка улучшена [6] до величины $\mathrm{O}(l+ha_1)$, где l и h — характеристики множества $\{a_1,\ldots,a_k\}$, принимающие для различных множеств аргументов значения от 2 до $\min(a_1,k)$.

В данной работе выведена формула числа $g(\theta,\lambda,l)$, использующая арифметические и теоретико-множественные операции во множестве натуральных чисел. Приведены примеры вычисления $g(\theta,\lambda,l)$ по полученной формуле. Число Фробениуса $g(\theta,\lambda,l)$ использовано для оценки экспонентов весьма широкого класса орграфов. Полученная оценка существенно улучшает известные оценки для рассмотренного класса орграфов.

1. Формула числа Фробениуса для трёх аргументов

Обозначим $K(\theta,\lambda,l)$ множество натуральных чисел k, таких, что $kdl\in C(\theta,\lambda)$, $S(l)$ — подмножество множества $C(\theta,\lambda)$:

$$S(l) = \bigcup_{k\in K(\theta,\lambda,l)} \{(kdl+b)\in C(\theta,\lambda) : b\in\overline{\langle\theta,\lambda\rangle}\}.$$

Утверждение 1.

а) Если $a\in\mathbb{N}$ и $d\mid a$, то $l\mid a$, если и только если $dl\mid a$;

б) $S(l) = C(\theta,\lambda)\cap\langle\theta,\lambda,l\rangle$;

в) $C(\theta,\lambda)\backslash S(l)\neq\varnothing$;

г) $S(l)=\varnothing$, если и только если $K(\theta,\lambda,l)=\varnothing$.

Доказательство.

а) По условию $d=(\theta,\lambda)$ и числа θ,λ,l взаимно простые, тогда $(l,d)=1$, отсюда следует требуемое утверждение.

б) Пусть $a\in C(\theta,\lambda)\cap\langle\theta,\lambda,l\rangle$. Тогда $a=nl+b$, где $b\in\langle\theta,\lambda\rangle$, при этом $n\in\mathbb{N}$, иначе при $n=0$ имеем противоречие: $a\in\langle\theta,\lambda\rangle$.

Ввиду $a\in C(\theta,\lambda)$ имеем $a-b=nl\in C(\theta,\lambda)$, иначе приходим к противоречию: $a\in\langle\theta,\lambda\rangle$.

Заметим, что $d\mid nl$ и $l\mid nl$, тогда по утверждению 1a верно $dl\mid nl$. Отсюда $nl=kdl$ при натуральном k, где $k\in K(\theta,\lambda,l)$, так как $nl\in C(\theta,\lambda)$. Тогда $a=kdl+b\in S(l)$. Следовательно, $C(\theta,\lambda)\cap\langle\theta,\lambda,l\rangle\subseteq S(l)$.

Обратно, пусть $a\in S(l)$, тогда $a=kdl+b$ при некотором $k\in K(\theta,\lambda,l)$, где $ndl\in$ $\in C(\theta,\lambda)$ и $b\in\overline{\langle\theta,\lambda\rangle}$. По определению $S(l)\subseteq C(\theta,\lambda)$, поэтому получаем $a\in C(\theta,\lambda)\cap$ $\cap\langle\theta,\lambda,l\rangle$. Значит, выполнено и обратное включение $S(l)\subseteq C(\theta,\lambda)\cap\langle\theta,\lambda,l\rangle$.

в) Для любого $b \in \{1, \ldots, \theta - 1\}$ выполнено $b \notin \langle \theta, \lambda \rangle$ и $b \notin \langle \theta, \lambda, l \rangle$, тогда $b \in C(\theta, \lambda)$ и $b \notin S(l)$. Следовательно, $C(\theta, \lambda) \backslash S(l) \neq \varnothing$.

г) Следует из определения множества $S(l)$. ■

Уточним строение множества $S(l)$. Определим на множестве $\mathbb{N}_0^2$ всех пар целых неотрицательных чисел отношение частичного порядка: $(n_1, m_1) \leqslant (n_2, m_2)$, если и только если $n_1 \leqslant n_2$ и $m_1 \leqslant m_2$, при этом если $n_1 < n_2$ или $m_1 < m_2$, то пишется $(n_1, m_1) < (n_2, m_2)$. Множество $\mathbb{N}_0^2$ — решётка относительно частичного порядка $\leqslant$. Нулём решетки $\mathbb{N}_0^2$ является пара $(0, 0)$. Через $I(n, m)$ обозначим идеал решетки $\mathbb{N}_0^2$:

$$I(n, m) = \{(n', m') \in N_0^2 : (n', m') \leqslant (n, m)\}.$$

На решётке $\mathbb{N}_0^2$ определена операция сложения $(n, m) + (n', m') = (n + n', m + m')$, относительно которой множество $\mathbb{N}_0^2$ образует полугруппу. (Заметим, что эта операция отличается от операции сложения в решётке, определяемой как $a + b = \sup\{a, b\}$.)

Лемма 1. Любое число $b \in \overline{\langle \theta, \lambda \rangle}$ однозначно представляется в виде линейной комбинации чисел θ и λ с целыми неотрицательными коэффициентами.

Доказательство. Пусть имеются два различных представления

$$b = n\theta + m\lambda = n'\theta + m'\lambda.$$

Все коэффициенты положительные, поэтому пары (n, m) и (n', m') несравнимы. Пусть $0 \leqslant n < n'$ и $0 \leqslant m' < m$. Тогда $(n' - n)\theta = (m - m')\lambda$. Так как числа θ/d и λ/d взаимно простые, последнее равенство выполнено только если $n' - n = k\lambda/d$ и $m - m' = k\theta/d$ при некотором натуральном k. Отсюда $m \geqslant k\theta/d$ и $b \geqslant n\theta + k\theta\lambda/d \geqslant \theta\lambda/d$, то есть в соответствии с (4) $b > z$, что противоречит условию $b \in \overline{\langle \theta, \lambda \rangle}$. ■

Лемма 1 устанавливает биекцию f между множеством $\overline{\langle \theta, \lambda \rangle}$ и подмножеством решетки $\mathbb{N}_0^2$: если $b = n\theta + m\lambda$, то $f(b) = (n, m)$. В этом случае определим идеал $I(b) = I(n, m)$.

Заметим, что полугрупповая операция на множестве $\overline{\langle \theta, \lambda \rangle}$ согласована с операцией сложения в $\mathbb{N}_0^2$, то есть если $a, b, (a + b) \in \overline{\langle \theta, \lambda \rangle}$, то $f(a + b) = f(a) + f(b)$.

Лемма 2. Если $a, b \in \overline{\langle \theta, \lambda \rangle}$, то $a - b \in \overline{\langle \theta, \lambda \rangle}$, если и только если $f(b) \leqslant f(a)$.

Доказательство. Пусть $f(a) = (n, m)$, $f(b) = (n', m')$. Если $(n', m') \leqslant (n, m)$, то $f(a - b) = (n - n', m - m')$, то есть $a - b \in \overline{\langle \theta, \lambda \rangle}$.

Обратно, если $a - b \in \overline{\langle \theta, \lambda \rangle}$, то $f(a - b) = (n'', m'')$, где $n'', m'' \geqslant 0$. Тогда $a = b + (a - b)$, следовательно, $(n', m') \leqslant (n, m)$, то есть $f(b) \leqslant f(a)$. ■

Утверждение 2. $S(l) = \bigcup\limits_{k \in K(\theta, \lambda, l)} (kdl + I(g(\theta, \lambda) - kdl))$.

Доказательство. Обозначим

$$b_k = g(\theta, \lambda) - kdl, \tag{5}$$

где $k \in K(\theta, \lambda, l)$. В силу равенств (2), (3) при $k \in K(\theta, \lambda, l)$ выполнено $b_k \in \overline{\langle \theta, \lambda \rangle}$, тогда $kdl + b_k = g(\theta, \lambda) \in S(l)$. Возьмём $b \in \overline{\langle \theta, \lambda \rangle}$; если $f(b) \leqslant f(b_k)$ и $kdl + b \in C(\theta, \lambda)$, то $kdl + b \in S(l)$. Тогда $(kdl + I(b_k)) \subseteq S(l)$. В силу произвольности $k \in K(\theta, \lambda, l)$ получаем $\bigcup\limits_{k \in K(\theta, \lambda, l)} (kdl + I(b_k)) \subseteq S(l)$.

Обратно, если $a \in S(l)$, то по определению $a = kdl + b$, где $a, kdl \in C(\theta, \lambda)$ и $b \in \overline{\langle \theta, \lambda \rangle}$. Тогда $b = a - kdl$. Вычитая это равенство из (5), получаем $b_k - b = g(\theta, \lambda) - a$. В силу

равенств (2), (3) имеем $(g(\theta,\lambda)-a) \in \overline{\langle\theta,\lambda\rangle}$, следовательно, $b_k - b \in \overline{\langle\theta,\lambda\rangle}$. Отсюда $f(b) \leqslant f(b_k)$ в соответствии с леммой 2, то есть $b \in I(b_k)$ и $a \in (kdl + I(b_k))$. Таким образом, верно и обратное включение $S(l) \subseteq \bigcup\limits_{k\in K(\theta,\lambda,l)} (kdl + I(b_k))$. ■

Теорема 1. $g(\theta,\lambda,l) = \max C(\theta,\lambda)\backslash S(l) + (d-1)l$.

Доказательство. Пусть $d = 1$. По определению $C(\theta,\lambda,l) = \mathbb{N}\backslash\langle\theta,\lambda,l\rangle$. При $d = 1$ имеет место включение $C(\theta,\lambda,l) \subseteq C(\theta,\lambda)$, поэтому

$$C(\theta,\lambda,l) = C(\theta,\lambda)\backslash\langle\theta,\lambda,l\rangle = C(\theta,\lambda)\backslash(C(\theta,\lambda)\cap\langle\theta,\lambda,l\rangle).$$

Отсюда в соответствии с утверждением 1*б* получим $C(\theta,\lambda,l) = C(\theta,\lambda)\backslash S(l)$. Следовательно, $g(\theta,\lambda,l) = \max C(\theta,\lambda,l) = \max C(\theta,\lambda)\backslash S(l)$, то есть при $d = 1$ теорема верна.

Пусть $d > 1$. Из разбиения $\mathbb{N}_0 = \bigcup\limits_{r=0}^{d-1}(r + d\mathbb{N}_0)$, где $r + d\mathbb{N}_0$ — множество всех чисел из $\mathbb{N}_0$, сравнимых с r по модулю d, $r = 0,\ldots,d-1$, следует разбиение

$$C(\theta,\lambda,l) = \bigcup_{r=0}^{d-1}(r + dN_0)\cap C(\theta,\lambda,l).$$

Отсюда получаем

$$\max C(\theta,\lambda,l) = \max\{z_0,\ldots,z_{d-1}\}, \tag{6}$$

где $z_r = \max((r + d\mathbb{N}_0)\cap C(\theta,\lambda,l))$, $r = 0,\ldots,d-1$.

Используем очевидное разбиение $d\mathbb{N}_0 = \langle\theta,\lambda\rangle \cup S(l) \cup (C(\theta,\lambda)\backslash S(l))$.

Любое число a из $C(\theta,\lambda,l)$ имеет вид $a = nl + b$, где n — натуральное, b кратно d и $b \notin \langle\theta,\lambda,l\rangle$. Тогда $b \notin \langle\theta,\lambda\rangle \cup S(l)$, иначе имеем противоречие: $a \in \langle\theta,\lambda,l\rangle$. Следовательно, $b \in C(\theta,\lambda)\backslash S(l)$ и при $r = 0,\ldots,d-1$

$$z_r = \max((r + d\mathbb{N}_0)\cap(C(\theta,\lambda)\backslash S(l))).$$

Число $a \in (r + dN_0)\cap(C(\theta,\lambda)\backslash S(l))$, если и только если $nl \equiv r \pmod d$, $r = 0,\ldots,d-1$. Следовательно, $z_r = nl + \beta$, где $nl \equiv r \pmod d$ и $\beta = \max C(\theta,\lambda)\backslash S(l)$. Покажем, что n — наименьшее натуральное, при котором $nl \equiv r \pmod d$. Заметим, что в силу взаимной простоты чисел d и l при любом $r = 0,\ldots,d-1$ такое наименьшее $n \in \in \{0,\ldots,d-1\}$. Действительно, при $n \geqslant d$ разделим n на d с остатком: $n = dq + \delta$, где $q > 0$, $\delta < d$. Тогда

$$z_r = (qdl + \beta) + \delta l,$$

где число $(dql+\beta)$ больше β и кратно d. Значит, $(dql+\beta) \in \langle\theta,\lambda\rangle\cup S(l)$, следовательно, $(dql + \beta) \in \langle\theta,\lambda,l\rangle$. Отсюда получаем противоречие: $z_r \in \langle\theta,\lambda,l\rangle$. Тогда в соответствии с (6)

$$\max C(\theta,\lambda,l) = \max C(\theta,\lambda)\backslash S(l) + \max\{0,l,\ldots,(d-1)l\} = \max C(\theta,\lambda)\backslash S(l) + (d-1)l.$$

Теорема доказана. ■

Замечание. Из теоремы 1 и равенства (4) следует оценка числа $g(\theta,\lambda,l)$ (частный случай оценки Брауэра [7]):

$$g(\theta,\lambda,l) \leqslant \theta\lambda/d - \theta - \lambda + (d-1)l. \tag{7}$$

Пример 1. Определим $g(7,11,12)$. Вычисляем: $d=1$, $z=7\cdot 11-7-11=59$, тогда $|\overline{\langle 7,11\rangle}|=|C(7,11)|=6\cdot 10/2=30$.

Выпишем в табл. 1 множество чисел $a\in\overline{\langle 7,11\rangle}$ с соответствующими парами $f(a)\in\mathbb{N}_0^2$ и множество чисел $(z-a)\in C(7,11)$. Получаем, что множество чисел из $C(7,11)$, кратных 12, есть $\{12,24,48\}$. Тогда $K(7,11,12)=\{1,2,4\}$. В соответствии с утверждением 2 надо определить идеалы $I(59-12)=I(47)$, $I(59-24)=I(35)$, $I(59-48)=I(11)$.

Таблица 1

$a/f(a)$	$z-a$	$a/f(a)$	$z-a$	$a/f(a)$	$z-a$	$a/f(a)$	$z-a$	$a/f(a)$	$z-a$	$a/f(a)$	$z-a$
56/(8,0)	**3**										
49/(7,0)	**10**										
42/(6,0)	**17**	53/(6,1)	**6**								
35/(5,0)	**24**	46/(5,1)	**13**	57/(5,2)	**2**						
28/(4,0)	**31**	39/(4,1)	**20**	50/(4,2)	**9**						
21/(3,0)	**38**	32/(3,1)	**27**	43/(3,2)	**16**	54/(3,3)	**5**				
14/(2,0)	**45**	25/(2,1)	**34**	36/(2,2)	**23**	47/(2,3)	**12**	58/(2,4)	**1**		
7/(1,0)	**52**	18/(1,1)	**41**	29/(1,2)	**30**	40/(1,3)	**19**	51/(1,4)	**8**		
0/(0,0)	**59**	11/(0,1)	**48**	22/(0,2)	**37**	33/(0,3)	**26**	44/(0,4)	**15**	55/(0,5)	**4**

Из табл. 1 имеем: $f(47)=(2,3)$, $f(35)=(5,0)$, $f(11)=(0,1)$. Зададим идеалы в табл. 2 (три части табл. 1).

Таблица 2

Идеал $I(47)$				Идеал $I(35)$	Идеал $I(11)$
				35/(5,0)	
				28/(4,0)	
				21/(3,0)	
14/(2,0)	25/(2,1)	36/(2,2)	47/(2,3)	14/(2,0)	
7/(1,0)	18/(1,1)	29/(1,2)	40/(1,3)	7/(1,0)	11/(0,1)
0/(0,0)	11/(0,1)	22/(0,2)	33/(0,3)	0/(0,0)	0/(0,0)

Из табл. 2 в соответствии с утверждением 2 получаем

$$S(12)=(12+I(47))\cup(24+I(47))\cup(48+I(11))=$$
$$=\{26,19,12,37,30,23,48,41,34,59,52,45,38,31,24\}.$$

Отсюда и из табл. 1 получаем окончательно

$$g(7,11,12)=\max(C(7,11)\backslash S(12))=\max\{17,10,27,20,13,6,16,9,2,5,15,8,1,4\}=27.$$

Пример 2. Определим $g(8,38,41)$. Вычисляем: $d=2$, $z=2(4\cdot 19-4-19)=106$, тогда $|\overline{\langle 8,38\rangle}|=|C(8,38)|=3\cdot 18/2=27$.

Выпишем в табл. 3 множество чисел $a\in\overline{\langle 8,38\rangle}$ с соответствующими парами $f(a)\in$ $\in\mathbb{N}_0^2$ и множество чисел $(z-a)\in C(8,38)$. Получаем, что во множестве $C(8,38)$ лишь число 82 кратно 82, тогда $K(8,38,41)=\{1\}$. В соответствии с утверждением 2 определяем идеал $I(106-82)=I(24)$.

Из табл. 3 также имеем: $f(24)=(3,0)$; $I(24)=\{24,16,8,0\}$; $S(41)=82+I(24)=\{82,90,98,106\}$. Отсюда и из табл. 3 получаем окончательно

$$g(8,38,41)=\max(C(8,38)\backslash S(41))+41=74+41=115.$$

Таблица 3

$a/f(a)$	$z-a$	$a/f(a)$	$z-a$	$a/f(a)$	$z-a$
104/(13,0)	**2**				
96/(12,0)	**10**				
88/(11,0)	**18**				
80/(10,0)	**26**				
72/(9,0)	**34**				
64/(8,0)	**42**	102/(8,1)	**4**		
56/(7,0)	**50**	94/(7,1)	**12**		
48/(6,0)	**58**	86/(6,1)	**20**		
40/(5,0)	**66**	78/(5,1)	**28**		
32/(4,0)	**74**	70/(4,1)	**36**		
24/(3,0)	**82**	62/(3,1)	**44**	100/(3,2)	**6**
16/(2,0)	**90**	54/(2,1)	**52**	92/(2,2)	**14**
8/(1,0)	**98**	46/(1,1)	**60**	84/(1,2)	**22**
0/(0,0)	**106**	38/(0,1)	**68**	76/(0,2)	**30**

Заметим, что описание формулы числа Фробениуса для четырёх аргументов несколько сложнее, чем для трёх. Один из факторов усложнения — отсутствие жёсткой связи между множествами $\langle\theta,\lambda,l\rangle$ и $C(\theta,\lambda,l)$, подобной той, что задана для множеств $\langle\theta,\lambda\rangle$ и $C(\theta,\lambda)$ равенством (2).

2. Оценка экспонентов ориентированных графов

Используем формулу числа $g(\theta,\lambda,l)$ для оценки экспонентов некоторых графов.

Граф Γ с матрицей смежности вершин M называется примитивным, если $M^\gamma{>}0$ при некотором натуральном γ. Наименьшее такое число γ называют экспонентом (показателем примитивности) графа Γ (обозначается $\exp\Gamma$).

Универсальный критерий примитивности графа Γ [8, с. 226] следующий: если $C_1,\ldots,C_k$ суть все простые циклы графа Γ длин $l_1,\ldots,l_k$ соответственно, то сильносвязный орграф (связный граф) Γ примитивный, если и только если $(l_1,\ldots,l_k)=1$.

Универсальная и частные оценки экспонентов n-вершинных графов систематизированы и даны в [9]. Рассмотрим новые классы примитивных орграфов и графов и получим оценку их экспонентов.

В n-вершинном орграфе (графе) Γ при $n>1$ всякие три дуги (ребра) вида (i,r), (r,j) и (i,j), $i,r,j\in\{1,\ldots,n\}$, назовём транзитивной тройкой дуг (рёбер). Заметим, что всякая петля в орграфе (графе) является такой тройкой. Длину пути w в Γ, измеряемую числом дуг (рёбер), составляющих путь, обозначим $l(w)$. В частности, петля в Γ есть путь длины 1. Пустой путь (путь длины 0, который не содержит дуг (рёбер) и может относиться к любой вершине орграфа) обозначим $w_\varnothing$.

Для последовательности $w_1,\ldots,w_m$ из $m>1$ непустых путей определён путь w с помощью операции конкатенации, если конечная вершина предыдущего пути совпадает с начальной вершиной следующего пути (операция конкатенации обозначена символом $\cdot$), записывается $w=w_1{\cdot}\ldots{\cdot}w_m$. Если $w=w_1{\cdot}\ldots{\cdot}w_m$, то $l(w)=l(w_1){+}\ldots{+}l(w_m)$.

Теорема 2. Если при $n>2$ в сильносвязном n-вершинном орграфе Γ имеются транзитивная тройка дуг $(i,r),(r,j),(i,j)$ и простой путь w из j в i длины $t-1$, где $t>1$, то орграф Γ примитивный и выполнены оценки:

а) если $i=j$, или $r=i$, или $r=j$, то $\exp\Gamma\leqslant 2n-2$;

б) если i,j,r — три различные вершины и путь w не проходит через вершину r, то $\exp\Gamma\leqslant(t-1)^2+2n-2$;

в) если i, j, r — три различные вершины и путь w проходит через вершину r, которая делит путь w на пути длины $\tau - 1$ и $t - \tau$, где $0 < t - \tau < \tau < t$, то

$$\exp\Gamma \leqslant g(t - \tau + 1, \tau, t) + 2(n - t + \tau) - 1,$$

где g — число Фробениуса от трёх аргументов.

Доказательство.

а) Если $i = j$, или $r = i$, или $r = j$, то в Γ имеется петля, и орграф Γ примитивный. Тогда в соответствии с [8, с. 397] $\exp\Gamma \leqslant 2n - 2$.

б) Если i, j, r суть три различные вершины, $n > 2$ и путь w не проходит через вершину r, то $t \leqslant n - 1$ и в Γ имеются циклы z_1 и z_2, где $z_1 = w \cdot (i, r) \cdot (r, j)$, $z_2 = w \cdot (i, j)$. Отсюда $l(z_1) = t + 1$, $l(z_2) = t$, следовательно, $(l(z_1), l(z_2)) = 1$ и в соответствии с универсальным критерием орграф Γ является примитивным. Цикл z_2 — часть цикла z_1, отсюда в соответствии с оценкой [10, теорема 1*б*] $\exp\Gamma \leqslant (t - 1)^2 + 2n - 2$.

в) Если i, j, r суть три различные вершины, $n > 2$ и путь w проходит через вершину r, то $t \leqslant n$ и выполнено равенство $w = w_1 \cdot w_2$, где w_1 есть простой путь из j в r и w_2 — простой путь из r в i. Тогда в Γ имеются циклы z_0, z_1 и z_2, где $z_0 = w \cdot (i, j)$, $z_1 = w_1 \cdot (r, j)$, $z_2 = (i, r) \cdot w_2$. В соответствии с условиями $l(z_0) = t$, $l(z_1) = \tau$, $l(z_2) = t - \tau + 1$.

Пусть $(l(z_0), l(z_1), l(z_2)) = \delta$. Тогда δ делит $l(z_1) + l(z_2)$, то есть δ делит $t + 1$. Вместе с тем δ делит $l(z_0)$, где $l(z_0) = t$. Следовательно, $\delta = 1$ и граф Γ примитивный.

Для получения оценки $\exp\Gamma$ воспользуемся утверждением 3*б* из [10].

При обходе какого-либо цикла z' выделим его вершину ε как начальную, цикл в этом случае обозначим $z'(\varepsilon)$. Для целого неотрицательного q через $qz'(\varepsilon)$ обозначим цикл, составленный из q-кратно пройденного цикла $z'(\varepsilon)$, где $0z'(\varepsilon) = w_\varnothing$.

Пусть множество W состоит из всех вершин пути w и V — из остальных вершин орграфа Γ, в частности, не исключено $V = \varnothing$. Пусть α, β — любые вершины орграфа Γ и $u(\alpha, \beta)$ — путь из α в β вида

$$u(\alpha, \beta) = u(\alpha, r) \cdot z(r) \cdot u(r, \beta), \tag{8}$$

где $u(\alpha, r)$ — кратчайший путь из α в r; $u(r, \beta)$ — кратчайший путь из r в β; $z(r)$ — цикл, проходящий только через вершины множества W. Так как вершина r является общей для циклов z_0, z_1 и z_2, то цикл $z(r)$ может быть представлен в виде

$$z(r) = q_0 z_0(r) \cdot q_1 z_1(r) \cdot q_2 z_2(r), \tag{9}$$

где q_0, q_1, q_2 — целые неотрицательные числа.

Оценим длину пути $u(\alpha, \beta)$. Используем представления

$$u(\alpha, \beta) = u(\alpha, \varepsilon) \cdot u(\varepsilon, r), u(r, \beta) = u(r, \xi) \cdot u(\xi, \beta), \tag{10}$$

где $u(\alpha, \varepsilon)$ — кратчайший путь из вершины α до ближайшей вершины ε из множества W; $u(\xi, \beta)$ — кратчайший путь в β из ближайшей вершины ξ множества W; $u(\varepsilon, r)$ и $u(r, \xi)$ — пути наименьшей длины соответственно из ε в r и из r в ξ, проходящие только через вершины множества W. Заметим, что при $\alpha \in W$ путь $u(\alpha, \varepsilon)$ пустой, при $\beta \in W$ путь $u(\xi, \beta)$ пустой.

Учитывая, что $|W| = t$, где по условию $t \leqslant n$, из определения данных путей получаем $l(u(\alpha, \varepsilon)) \leqslant n - t$, $l(u(\xi, \beta)) \leqslant n - t$, $l(u(\varepsilon, r)) < \max\{l(z_1), l(z_2)\} = \tau$, $l(u(r, \xi)) < \max\{l(z_1), l(z_2)\} = \tau$. Отсюда и из (10) следует

$$l(u(\alpha, r)) \leqslant n - t + \tau - 1, \quad l(u(r, \beta)) \leqslant n - t + \tau - 1. \tag{11}$$

В соответствии с определением числа Фробениуса и равенством (9) коэффициенты q_0, q_1, q_2 можно выбрать так, чтобы длина пути $z(r)$ равнялась любому числу, превосходящему число Фробениуса $g(t-\tau+1, \tau, t)$. Следовательно, из (8) и (11) получаем, что подбором коэффициентов q_0, q_1, q_2 можно добиться, чтобы для любых вершин α, β длина пути $u(\alpha, \beta)$ принимала любое значение, большее или равное $g(t-\tau+1, \tau, t) + 2(n-t+\tau) - 1$. По следствию 1 теоремы 2 [11] это равносильно положительности матрицы A^s при любом $s \geqslant g(t-\tau+1, \tau, t) + 2(n-t+\tau) - 1$, где A — матрица смежности вершин графа Г. Требуемая оценка экспонента выполнена. ■

Следствие. Если выполнены условия теоремы 2*в* и $(t+1, \tau) = d$, то

$$\exp \Gamma \leqslant (t-\tau+1)\tau/d + (d-4)t + 2n + 2\tau - 2.$$

Доказательство. Заметим, что $(t+1, \tau) = (t-\tau+1, \tau) = d$. С учётом этого требуемая оценка получается из теоремы 2*в* с использованием оценки (7). ■

Пример 3. Пусть для примитивного орграфа Г со 100 вершинами выполнены условия теоремы 2*в*, где $t = 62$, $\tau = 36$. Тогда $t - \tau + 1 = 27$, требуется определить $g(27, 36, 62)$.

Вычисляем: $d = (27, 36) = 9$, $z = 9 \cdot 5 = 45$, тогда $|\overline{\langle 27, 36 \rangle}| = |C(27, 36)| = 6/2 = 3$.

Поскольку $\overline{\langle 27, 36 \rangle} = \{0, 27, 36\}$, $C(27, 36) = \{9, 18, 45\}$, получаем $S(62) = \varnothing$ и $\max C(27, 36) = 45$. В соответствии с теоремой 1 $g(27, 36, 62) = 45 + 8 \cdot 62 = 541$. Отсюда по теореме 2*в* $\exp \Gamma \leqslant 541 + 2 \cdot 74 - 1 = 688$.

Пример 4. Пусть для примитивного орграфа Г со 100 вершинами выполнены условия теоремы 2*в*, где $t = 59$, $\tau = 52$. Тогда $t - \tau + 1 = 8$, требуется определить $g(8, 52, 59)$.

Вычисляем: $d = (8, 52) = 4$, $z = 4 \cdot 11 = 44$, тогда $|\overline{\langle 8, 52 \rangle}| = |C(8, 52)| = 12/2 = 6$.

Имеем $\overline{\langle 8, 52 \rangle} = \{0, 8, 16, 24, 32, 40\}$, $C(8, 52) = \{4, 12, 20, 28, 36, 44\}$. Множество чисел из $C(8, 52)$, кратных 236, пусто, значит, пусто множество $S(236)$. Тогда по теореме 1

$$g(8, 52, 59) = \max C(8, 52) + 3 \cdot 59 = 44 + 177 = 221.$$

Отсюда по теореме 2*в* $\exp \Gamma \leqslant 221 + 2 \cdot 93 - 1 = 406$.

Точное сравнение полученной оценки с известными оценками затруднено наличием в ней слагаемого в виде числа Фробениуса, которое во многих случаях вносит основной вклад в величину оценки. Рассмотрим числовые примеры.

Для примитивного 100-вершинного орграфа из примера 3 оценки следующие:

— универсальная оценка Виландта [12] — 9802;
— оценка, данная в [8, с. 227], — 2746 (длина кратчайшего цикла 27);
— оценка теоремы 1*б* [10] — 1758 (длины циклов 27 и 62);
— оценка примера 3 — 688.

Для примитивного 100-вершинного орграфа из примера 4 оценки следующие:

— универсальная оценка Виландта [12] — 9802;
— оценка, данная в [8, с. 227], — 884 (длина кратчайшего цикла 8);
— оценка теоремы 1*б* [10] — 597 (длины циклов 8 и 59);
— оценка примера 4 — 406.

Анализ числовых примеров позволяет считать, что полученная оценка, как правило, значительно точнее других известных оценок.

ЛИТЕРАТУРА

1. *Alfonsin J. R.* The Diophantine Frobenius Problem. Oxford University Press, 2005.
2. *Sylvester J. J.* Problem 7382 // Mathematical Questions from the Educational Times. 1884. V. 37. P. 26.
3. *Curtis F.* On formulas for the Frobenius number of a numerical semigroup // Math. Scand. 1990. No. 67. P. 190–192.
4. *Heap B. R. and Lynn M. S.* On a linear Diophantine problem of Frobenius: an improved algorithm // Numer. Math. 1965. No 7. P. 226–231.
5. *Bocker S. and Liptak Z.* The "money changing problem" revisited: computing the Frobenius number in time $O(ka_1)$. Technical Report No. 2004-2, Univ. of Bielefeld, Technical Faculty, 2004.
6. *Фомичев В. М.* Эквивалентные по Фробениусу примитивные множества чисел // Прикладная дискретная математика. 2014. № 1(23). С. 20–26.
7. *Brauer A.* On a problem of partitions // Am. J. Math. 1942. No. 64. P. 299–312.
8. *Сачков В. Н., Тараканов В. Е.* Комбинаторика неотрицательных матриц. М.: ТВП, 2000. 448 с.
9. *Когос К. Г., Фомичев В. М.* Положительные свойства неотрицательных матриц // Прикладная дискретная математика. 2012. № 4(18). С. 116–121.
10. *Фомичев В. М.* Оценки экспонентов примитивных графов // Прикладная дискретная математика. 2011. № 2(12). С. 101–112.
11. *Берж К.* Теория графов и её применение. М.: ИЛ, 1962. 320 с.
12. *Wielandt H.* Unzerlegbare nicht negative Matrizen // Math. Zeitschr. 1950. N. 52. S. 642–648.

2014 Прикладная теория графов №2(24)

DOI 10.17223/20710410/24/8

УДК 519.248:62-192+519.176

ВЫВОД АСИМПТОТИЧЕСКИХ КОНСТАНТ ДЛЯ ВЕРОЯТНОСТИ НЕСВЯЗНОСТИ ПЛАНАРНОГО ВЗВЕШЕННОГО ГРАФА[1]

Г. Ш. Цициашвили, М. А. Осипова, А. С. Лосев

Институт прикладной математики ДВО РАН, ДВФУ, г. Владивосток, Россия
Дальневосточный федеральный университет, г. Владивосток, Россия

E-mail: guram@iam.dvo.ru, mao1975@list.ru, alexax@bk.ru

Приведено доказательство формул для вычисления асимптотических констант вероятности несвязности планарного взвешенного графа с высоконадёжными рёбрами.

Ключевые слова: *вес, грань, цикл, вероятность несвязности.*

Введение

В [1] построен алгоритм вычисления вероятности несвязности для планарного взвешенного графа с высоконадёжными рёбрами. Алгоритм имеет кубическую по числу рёбер в двойственном графе сложность. Он основан на доказательстве асимптотического соотношения и на получении формул для вычисления его параметров. Речь идёт о минимальном объёме разреза и о некотором весовом коэффициенте. В настоящей работе приводится вывод формул для вычисления весового коэффициента.

1. Основной результат

Рассмотрим неориентированный связный граф G без петель и кратных рёбер с конечным множеством вершин U и рёбер W. Пусть каждому ребру графа $w \in W$ соответствует вес b_w. Обозначим $\mathcal{L}$ множество разрезов графа, $d(L)$ — число рёбер (объём) разреза L, D — минимальный объём разрезов. Предположим, что рёбра графа G отказывают независимо с вероятностями $\overline{p}(w)$, $w \in W$. Для вероятности несвязности $\overline{P}$ графа G (отсутствия хотя бы между двумя вершинами графа работающего пути) в [2] приведена следующая теорема.

Теорема 1. Если $\overline{p}(w) \sim b_w h$, $h \to 0$, $w \in W$, где $b_w > 0$, $w \in W$, то

$$\overline{P} \sim h^D \mathcal{B}_D, \quad \mathcal{B}_D = \sum_{L \in \mathcal{L}: d(L) = D} \prod_{w \in L} b_w, \quad h \to 0.$$

Замечание 1. Условие $\overline{p}(w) \sim b_w h$, $h \to 0$, $w \in W$, в отличие от условия $\overline{p}(w) \sim h$, $h \to 0$, $w \in W$, значительно расширяет область рассматриваемых моделей.

Рассмотрим планарный граф G, каждое ребро которого принадлежит какому-либо простому циклу. Рёбра планарного графа G разбивают плоскость на грани; обозначим n число граней (включая внешнюю), m — число рёбер графа. Графу G сопоставим двойственный граф G^*: грани z графа G соответствует вершина z графа G^*, ребру w графа G, принадлежащему граням z_1, z_2, — ребро w, соединяющее вершины z_1, z_2 графа G^*. Пусть элементы a_{ij}, $i, j = 1, \ldots, n$, матрицы A определяют число рёбер, содержащихся в пересечении граней $z_i \cap z_j$, $i \neq j$, $a_{ii} = 0$. Известно [3],

[1]Работа поддержана грантом РФФИ № 14-01-00873 А.

что $D = \min(k : 2 \leqslant k \leqslant 5, c_k > 0)$, где c_k — число простых циклов длины k в G^*, определяемое по формулам, которые приведены в [4].

Обозначим $\mathcal{K}^*$ множество циклов K^* графа G^*, $d(K^*)$ — длину цикла K^*, D^* — минимальную длину цикла. Известно [3], что циклам минимальной длины графа G^* соответствуют разрезы минимального объёма графа G, причём $D^* = D$. Тогда

$$\mathcal{B}_D = \sum_{K^* \in \mathcal{K}^* : d(K^*) = D} \prod_{w \in K^*} b_w. \tag{1}$$

Поэтому для $\mathcal{B}_D$ предлагается вывести аналоги формул [4], полученные для констант c_k. Пусть константы $b_{ij}(k) = b_{ji}(k) = b_{w_k}$, $k = 1, \ldots, a_{ij}$, определяют веса рёбер w_k, содержащихся в пересечении граней $z_i \cap z_j$, $1 \leqslant i \neq j \leqslant n$, графа G, при этом $b_{ii}(k) = 0$. В частности, в случае $D > 2$ имеет место $a_{ij} = 1$, $1 \leqslant i \neq j \leqslant n$.

Теорема 2. Для планарного графа G, каждое ребро которого принадлежит какому-либо циклу, имеют место соотношения

$$\mathcal{B}_2 = \frac{1}{4} \sum_{1 \leqslant i,j \leqslant n} \left(\left(\sum_{1 \leqslant k \leqslant a_{ij}} b_{ij}(k) \right)^2 - \sum_{1 \leqslant k \leqslant a_{ij}} b_{ij}^2(k) \right); \quad \mathcal{B}_3 = \frac{1}{6} \operatorname{tr} B^3; \tag{2}$$

$$\mathcal{B}_4 = \frac{1}{8} \left(\operatorname{tr} B^4 - 2 \sum_{1 \leqslant i,j,k \leqslant n} b_{ij}^2(1) b_{jk}^2(1) + \sum_{1 \leqslant i,j \leqslant n} b_{ij}^4(1) \right); \tag{3}$$

$$\mathcal{B}_5 = \frac{1}{10} \left(\operatorname{tr} B^5 - 5 \sum_{1 \leqslant i,j \leqslant n} b_{ij}^2(1) b_{jj}^{(3)}(1) + 5 \sum_{1 \leqslant i,j \leqslant n} b_{ij}^3(1) b_{ji}^{(2)}(1) \right), \tag{4}$$

где $B = ||b_{ij}(1)||_{i,j=1}^n$; $b_{ij}^{(l)}(1)$, $1 \leqslant i, j \leqslant n$, — элементы матрицы B^l, $l > 1$.

Доказательство. Используем рисунки замкнутых путей с четырьмя и пятью рёбрами, приведённые в [5]. Остановимся сначала на доказательстве формулы для $\mathcal{B}_2$:

$$\mathcal{B}_2 = \sum_{K^* \in \mathcal{K}^* : d(K^*) = 2} \prod_{w \in K^*} b_w = \frac{1}{4} \sum_{1 \leqslant i,j \leqslant n} \sum_{1 \leqslant t \neq s \leqslant a_{ij}} b_{ij}(t) b_{ij}(s) =$$

$$= \frac{1}{4} \sum_{1 \leqslant i,j \leqslant n} \left(\sum_{1 \leqslant t,s \leqslant a_{ij}} b_{ij}(t) b_{ij}(s) - \sum_{1 \leqslant t \leqslant a_{ij}} b_{ij}^2(t) \right) = \frac{1}{4} \sum_{1 \leqslant i,j \leqslant n} \left[\left(\sum_{1 \leqslant t \leqslant a_{ij}} b_{ij}(t) \right)^2 - \sum_{1 \leqslant t \leqslant a_{ij}} b_{ij}^2(t) \right].$$

Формула для $\mathcal{B}_3$ очевидна:

$$\mathcal{B}_3 = \sum_{K^* \in \mathcal{K}^* : d(K^*) = 3} \prod_{w \in K^*} b_w = \frac{1}{6} \sum_{1 \leqslant i,j,k \leqslant n} b_{ij}(1) b_{jk}(1) b_{ki}(1) = \frac{1}{6} \sum_{1 \leqslant i \leqslant n} b_{ii}^{(3)}(1) = \frac{1}{6} \operatorname{tr} B^3.$$

Доказательство формулы для $\mathcal{B}_4$ основано на рис. 1 из [5], в котором приведены всевозможные замкнутые пути, состоящие из четырёх рёбер:

$$\mathcal{B}_4 = \sum_{K^* \in \mathcal{K}^* : d(K^*) = 4} \prod_{w \in K^*} b_w = \frac{1}{8} \sum_{\substack{1 \leqslant i,j,k,m \leqslant n: \\ i \neq k, j \neq m}} b_{ij}(1) b_{jk}(1) b_{km}(1) b_{mi}(1) =$$

$$= \frac{1}{8} \sum_{1 \leqslant i,j,k,m \leqslant n} b_{ij}(1) b_{jk}(1) b_{km}(1) b_{mi}(1) - \frac{1}{8} \sum_{1 \leqslant i,j,k \leqslant n : i \neq k} b_{ij}(1) b_{jk}(1) b_{kj}(1) b_{ji}(1) -$$

$$- \frac{1}{8} \sum_{1 \leqslant i,j,m \leqslant n : j \neq m} b_{ij}(1) b_{ji}(1) b_{im}(1) b_{mi}(1) - \frac{1}{8} \sum_{1 \leqslant i,j \leqslant n} b_{ij}(1) b_{ji}(1) b_{ij}(1) b_{ji}(1).$$

Достаточно несложные выкладки приводят далее к равенству (3).

Для вычисления $\mathcal{B}_5$ используем рис. 3 и 4 из [5], в которых представлены всевозможные замкнутые пути, состоящие из пяти рёбер:

$$\mathcal{B}_5 = \sum_{K^*\in\mathcal{K}^*:d(K^*)=5}\ \prod_{w\in K^*} b_w = \frac{1}{10}\sum_{\substack{1\leqslant i,j,k,m,s\leqslant n:\\ i\neq k,i\neq m,j\neq m,j\neq s,k\neq s}} b_{ij}(1)b_{jk}(1)b_{km}(1)b_{ms}(1)b_{si}(1) =$$

$$= \frac{1}{10}\sum_{1\leqslant i,j,k,m,s\leqslant n} b_{ij}(1)b_{jk}(1)b_{km}(1)b_{ms}(1)b_{si}(1) - \frac{1}{10}\sum_{\substack{1\leqslant i,j,m,s\leqslant n:\\ j\neq m,j\neq s}} b_{ij}(1)b_{ji}(1)b_{im}(1)b_{ms}(1)b_{si}(1) -$$

$$-\frac{1}{10}\sum_{\substack{1\leqslant i,j,k,s\leqslant n:\\ j\neq s,k\neq s}} b_{ij}(1)b_{jk}(1)b_{ki}(1)b_{is}(1)b_{si}(1) - \frac{1}{10}\sum_{\substack{1\leqslant i,j,k,s\leqslant n:\\ i\neq k,k\neq s}} b_{ij}(1)b_{jk}(1)b_{kj}(1)b_{js}(1)b_{si}(1) -$$

$$-\frac{1}{10}\sum_{\substack{1\leqslant i,j,k,m\leqslant n:\\ i\neq k,i\neq m}} b_{ij}(1)b_{jk}(1)b_{km}(1)b_{mj}(1)b_{ji}(1) - \frac{1}{10}\sum_{\substack{1\leqslant i,j,k,m\leqslant n:\\ i\neq m,j\neq m}} b_{ij}(1)b_{jk}(1)b_{km}(1)b_{mk}(1)b_{ki}(1) -$$

$$-\frac{1}{10}\sum_{1\leqslant i,j,s\leqslant n} b_{ij}(1)b_{ji}(1)b_{ij}(1)b_{js}(1)b_{si}(1) - \frac{1}{10}\sum_{1\leqslant i,j,k\leqslant n} b_{ij}(1)b_{jk}(1)b_{ki}(1)b_{ik}(1)b_{ki}(1) -$$

$$-\frac{1}{10}\sum_{1\leqslant i,j,k\leqslant n} b_{ij}(1)b_{jk}(1)b_{kj}(1)b_{jk}(1)b_{ki}(1) - \frac{1}{10}\sum_{1\leqslant i,j,k\leqslant n} b_{ij}(1)b_{jk}(1)b_{ki}(1)b_{ij}(1)b_{ji}(1) -$$

$$-\frac{1}{10}\sum_{1\leqslant i,j,m\leqslant n} b_{ij}(1)b_{ji}(1)b_{im}(1)b_{mj}(1)b_{ji}(1).$$

Сравнительно простые выкладки приводят далее к равенству (4). ■

Замечание 2. Число арифметических операций, необходимых для реализации алгоритма вычисления констант D, $\mathcal{B}_D$ с помощью описанного алгоритма, равно $\mathrm{O}(n^{\min(3,D)})$, где n — число вершин в графе G^*.

Замечание 3. Прямое использование формулы (1) может привести к увеличению необходимого числа арифметических операций до $\mathrm{O}(n^{\max(3,D)})$, поскольку для определения путей $K^*\in\mathcal{K}^* : d(K^*) = D$ при реализации формулы (1) необходимо перебирать все замкнутые пути длины D в графе G^*. Однако в широко распространённом случае $D = 2$ у алгоритма, основанного на формуле (1), появляются преимущества в точности вычисления в системе с плавающей запятой из-за отсутствия операций вычитания.

2. Вычислительный эксперимент

Сравним результаты вычисления вероятности несвязности $\overline{P}$ по асимптотической формуле и методом Монте-Карло $\overline{P}^*$ с числом реализаций 10^6. Положим $h = 0{,}02$, $b(1,2) = b(1,4) = b(1,3) = b(5,3) = b(5,4) = b(5,2) = 1$, $b(2,3) = b(2,4) = b(3,4) = 1{,}2$, тогда

$$\overline{P}_G \approx 0{,}000071424, \quad \overline{P}^*_G \approx 0{,}000068, \quad \left|\frac{\overline{P}^*_{G_1}}{\overline{P}_{G_1}} - 1\right| \approx 0{,}05029.$$

В результате проведённых вычислений время счёта методом Монте-Карло составило 20 мин, а по асимптотическому соотношению — не более 1 мин, что подтверждает полученную теоретическую оценку сложности вычислений.

Авторы благодарят А. Н. Воропаева за помощь в проверке равенств (3), (4) и рецензента за полезные замечания и рекомендации.

ЛИТЕРАТУРА

1. *Tsitsiashvili G. Sh., Osipova M. A., and Losev A. S.* Disconnection probability of planar weighted graph // Appl. Math. Sci. 2014. No. 8(10). P. 469–472.
2. *Tsitsiashvili G. Sh.* Complete calculation of disconnection probability in planar graphs // Reliability: The. Appl. 2012. No. 7(1). P. 154–159.
3. *Прасолов В. В.* Элементы комбинаторной и дифференциальной топологии. М.: МЦНМО, 2004.
4. *Цициашвили Г. Ш., Лосев А. С.* Связность планарного графа с высоконадёжными ребрами // Прикладная дискретная математика. 2012. № 3(17). С. 102–106.
5. *Harary F. and Manvel B.* On the number of cycles in a graph // Matematickycasopis. 1971. No. 21(1). P. 55–63.

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ДИСКРЕТНЫХ АВТОМАТОВ

DOI 10.17223/20710410/24/9

УДК 004.312, 530.145

БЫСТРЫЙ АЛГОРИТМ СИНТЕЗА ОБРАТИМЫХ СХЕМ НА ОСНОВЕ ТЕОРИИ ГРУПП ПОДСТАНОВОК

Д. В. Закаблуков

Московский государственный технический университет им. Н. Э. Баумана, г. Москва, Россия

E-mail: dmitriy.zakablukov@gmail.com

Рассматриваются различные алгоритмы синтеза обратимых схем. Приведены результаты сравнения этих алгоритмов по основным характеристикам. Предложен новый быстрый алгоритм синтеза обратимых схем, основанный на теории групп подстановок и позволяющий получить схему с вентильной сложностью $\mathrm{O}(n2^m)$ за время $\mathrm{O}(n2^m)$ без использования дополнительных входов, где n — количество входов схемы, m — верхняя граница логарифма количества подвижных точек заданного преобразования.

Ключевые слова: *обратимые схемы, алгоритм синтеза, группы подстановок.*

Введение

Обратимость вычислений может потребоваться в совершенно различных областях науки и техники, таких, как квантовые вычисления и нанотехнологии. Данное требование зачастую обусловлено необходимостью максимально снизить величину тепловых потерь. И только обратимость вычислений гарантирует теоретический нулевой уровень тепловых потерь [1]. Как следствие, схемы из обратимых вентилей могут найти широкое применение в устройствах, работающих в условиях ограниченных вычислительных ресурсов, в том числе и в устройствах защиты информации.

Если алгоритм защиты можно описать обратимым преобразованием и его можно реализовать в обратимой схеме, то в таком случае в одной и той же схеме за счёт обратимости реализуются прямой алгоритм и обратный к нему, поэтому можно говорить об оценке сверху для вентильной сложности реализации этих алгоритмов.

Введём базовые понятия. *Логический вентиль* $n \times m$ — устройство с n входами и m выходами, дающее на выходах результат булевого преобразования $f\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ над входами. *Обратимый вентиль* $n \times n$ (далее просто обратимый вентиль) — логический вентиль $n \times n$, для которого реализуемое им булево преобразование является биекцией. Далее по тексту n обозначает количество входов и выходов обратимого вентиля, если не оговорено иначе. В данной работе среди всех обратимых вентилей рассматриваются только вентили NOT и k-CNOT:

1) N_j — вентиль NOT, инвертирующий свой j-й вход:

$$f_j\left(\langle x_1, \ldots, x_j, \ldots, x_n\rangle\right) = \langle x_1, \ldots, x_j \oplus 1, \ldots, x_n\rangle;$$

2) $C_{i_1,\ldots,i_k;j}$ — вентиль k-CNOT (инвертор с k контролирующими входами), инвертирующий свой j-й вход тогда и только тогда, когда сигнал на всех входах $i_1, \ldots, i_k$ равен 1:

$$f_{i_1,\ldots,i_k;j}\left(\langle x_1, \ldots, x_j, \ldots, x_n\rangle\right) = \langle x_1, \ldots, x_j \oplus x_{i_1} \wedge \ldots \wedge x_{i_k}, \ldots, x_n\rangle.$$

Правильно сформированная *обратимая схема* — ациклическая комбинационная логическая схема, в которой все вентили обратимы и соединены друг с другом последовательно без ветвлений. *Вентильная сложность* схемы — количество вентилей в ней.

В работе рассматриваются только такие обратимые схемы, в которых все вентили имеют одинаковое количество входов, при этом выходы одного вентиля напрямую соединяются со входами следующего за ним вентиля. В этом случае входами обратимой схемы являются входы первого вентиля, выходами — выходы последнего вентиля в композиции. Соединение вентилей (операцию композиции вентилей) будем обозначать $*$. Пример обратимой схемы при $n \geqslant 4$: $C_{4;1} * C_{2,3;4} * N_4$.

Любая обратимая схема задает биективное булево преобразование $f\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^n$. Рассмотрев все возможные векторы входных значений схемы и соответствующие им векторы выходных значений, можно говорить о задаваемой этой схемой подстановке h из симметрической группы $S(\mathbb{Z}_2^n)$ (для краткости будем обозначать симметрическую группу $S(\mathbb{Z}_2^n)$ как S_{2^n}). Следовательно, любая обратимая схема с n входами задаёт подстановку из S_{2^n}, и наоборот, любая подстановка из S_{2^n} задаёт семейство обратимых схем с n входами. Подстановка является *чётной*, если она представима в виде композиции чётного числа транспозиций, и *нечётной* в противном случае. Операцию композиции подстановок будем обозначать $\circ$.

Фундаментальный вопрос для теории обратимой логики заключается в следующем: какие обратимые функции могут быть реализованы при помощи заданной библиотеки обратимых вентилей. В данной работе рассмотрены основные переборные и неперборные алгоритмы синтеза обратимых схем из вентилей NOT и k-CNOT и их основные характеристики, приведены результаты сравнения алгоритмов синтеза по следующим характеристикам: временная сложность алгоритма; требуемое количество памяти для синтеза схемы; вентильная сложность синтезированной схемы. Представлен также новый, основанный на теории групп подстановок алгоритм синтеза обратимой схемы, состоящей из вентилей NOT, 1-CNOT и 2-CNOT и реализующей заданную чётную подстановку $h \in A_{2^n}$ при $n > 3$ с вентильной сложностью $\mathrm{O}(n2^m)$, где $m = \lceil \log_2 |M| \rceil$; $M = \{x : x \in \mathbb{Z}_{2^n}, h(x) \neq x\}$ — множество подвижных точек подстановки h.

Все алгоритмы синтеза схем можно классифицировать как переборные и неперборные. Нумерация алгоритмов в данной работе устроена следующим образом: переборные алгоритмы имеют префикс A1, неперборные — A2, предлагаемые новые алгоритмы — A3. Далее l означает вентильную сложность синтезированной схемы, если не оговорено иначе.

1. Обзор существующих алгоритмов синтеза

В работе [2] представлен переборный алгоритм A1.1 синтеза обратимой схемы, дающей на одном из выходов значение заданной булевой функции от входов. Основной задачей алгоритма является минимизация количества дополнительных входов синтезированной схемы. Он основан на «весовых функциях» из теории информации (терминология авторов) и предсказании наилучшего решения на один шаг вперед. Рабочей единицей алгоритма является каскад из N обратимых вентилей (N является

параметром алгоритма и задаётся вручную). Одновременно рассматриваются два последовательных каскада, и выбирается та пара каскадов, использование которой даёт наилучшее приближение к заданной функции. Авторами алгоритма не оговаривается, как выбирать параметр N, как время синтеза зависит от N и всегда ли алгоритм способен синтезировать схему для любой заданной функции. Остаётся неясным, какова вентильная сложность синтезированной схемы. Предположительно, алгоритму A1.1 для синтеза схемы требуется объём памяти, намного превосходящий N^2 (хранение всех возможных пар каскадов), а временная сложность намного превосходит $(N^2)^l$.

В работе [3] предложен переборный алгоритм A1.2 синтеза *оптимальных* по сложности обратимых схем из вентилей NOT, 1-CNOT и 2-CNOT, реализующих чётные подстановки. Данный алгоритм основан на технике *поиска в глубину с итеративным углублением*. Для его работы необходимо построить библиотеку оптимальных схем сложности k и реализуемых ими преобразований. Синтез схемы осуществляется полным перебором всевозможных соединений схем из библиотеки друг с другом. Согласно утверждению авторов, для некоторых заданных функций алгоритм A1.2 не способен синтезировать схему при помощи построенной библиотеки оптимальных схем. Временная сложность алгоритма A1.2 намного превышает величину N^l, где N — количество схем в данной библиотеке, а требуемый для синтеза объём памяти составляет порядка $\mathrm{O}(kN)$ (хранение библиотеки оптимальных схем).

В [4] представлен переборный алгоритм A1.3 синтеза обратимых схем с оптимальной вентильной сложностью, вентили которых входят в заранее сформированную библиотеку вентилей. Данный алгоритм использует математическое программное обеспечение `GAP` (Groups, Algorithms, Programming), представляющее собой систему для вычислительной дискретной алгебры. Авторами показывается, что проблему синтеза обратимой логики можно свести к проблеме теории групп подстановок, которую и решает `GAP`. Однако не даётся никаких оценок на временную сложность алгоритма A1.3 и требуемый им объём памяти для синтеза схемы.

В [5] представлен непереборный алгоритм A2.1 синтеза обратимых схем с близкой к оптимальной вентильной сложностью, использующий спектральный метод Радемахера — Уолша. Для заданного преобразования $f\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ строится система выходных функций $f_i(x_1, \ldots, x_n)$ и их спектров R_i. Для каждого R_i запускается алгоритм поиска следующего обратимого вентиля, применение которого даёт наилучшее приближение к заданной функции. По словам авторов, при рассмотрении всех вентилей k-CNOT время синтеза схемы растет экспоненциально, поэтому временную сложность алгоритма A2.1 можно оценить как $\mathrm{O}(2^{nl})$; требуемый для синтеза объём памяти составляет порядка $\mathrm{O}(n2^n)$ (хранение спектров n функций).

В [6] предложен непереборный алгоритм A2.2 синтеза обратимых схем из вентилей NOT и k-CNOT, входом которого является таблица заданного обратимого преобразования $f\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^n$. Суть алгоритма заключается в последовательном упорядочивании строк таблицы перестановками, соответствующими обратимым вентилям. Временная сложность алгоритма A2.2, по словам авторов, составляет порядка $\mathrm{O}(n2^n)$, результат синтеза гарантирован; вентильная сложность синтезированной схемы не превосходит $(n-1)2^n+1$. Требуемый для синтеза объём памяти составляет $\mathrm{O}(2^n)$ (хранение таблицы преобразования).

Похожий непереборный алгоритм A2.3 синтеза обратимых схем представлен в работе [7]. Алгоритм принимает на вход таблицу заданного обратимого преобразования $f\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^n$. Синтез схемы происходит путем упорядочивания в этой таблице строк и столбцов. Временная сложность алгоритма A2.3, по заверениям авторов, равна $\mathrm{O}(l2^n)$,

вентильная сложность синтезированной схемы близка к оптимальной. Требуемый для синтеза объём памяти составляет $\mathrm{O}(2^n)$ (хранение таблицы).

В [8] предложен неперебoрный алгоритм A2.4 синтеза обратимой схемы, реализующей заданную чётную подстановку $h \in A_{2^n}$ при $n > 3$. Эта подстановка представляется в виде произведения циклов длины 3 специального вида, каждый из которых реализуется вентилями NOT и 2-CNOT. По словам авторов, временная сложность алгоритма A2.4 в худшем случае равна $\frac{10}{3}n^2 2^n$, вентильная сложность синтезированной схемы не превосходит $4n^2 2^n$; требуемый для синтеза объём памяти зависит от подстановки h, но не превышает по порядку $\mathrm{O}(2^n)$ (хранение всех элементов подстановки).

2. Сравнение алгоритмов синтеза

Для заданного обратимого преобразования $f\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ рассмотрим соответствующую ему подстановку $h_f \in S_{2^n}$ и множество подвижных точек этой подстановки $M = \{x : x \in \mathbb{Z}_{2^n}, h_f(x) \neq x\}$. Обозначим $m = \lceil \log_2 |M| \rceil$. Подстановку h_f можно представить в виде композиции не более чем 2^m транспозиций. Для алгоритма синтеза A2.4 множитель 2^n в оценках его основных характеристик соответствует случаю $m = n$, поэтому ниже этот множитель заменён на 2^m.

В таблице приведено сравнение описанных выше алгоритмов синтеза обратимых схем из вентилей NOT и k-CNOT по основным характеристикам. Обозначения: $T(A)$ — временная сложность алгоритма; $M(A)$ — требуемый для синтеза объём памяти; l — вентильная сложность синтезированной схемы.

Сравнение алгоритмов синтеза обратимых схем

Алгоритм	Результат	$T(A)$	$M(A)$	l	Примечание
A1.1	Не гарантирован	$\gg (N^2)^l$	N^2	Не известно	Переборный, не универсальный; N — параметр алгоритма
A1.2	Не гарантирован	$\gg N^l$	$\mathrm{O}(kN)$	Оптимальная	Переборный; N — размер библиотеки оптимальных схем сложности $< k$
A1.3	Гарантирован	Не известно	Не известно	Оптимальная	Переборный; использует ПО *GAP*
A2.1	Не гарантирован	$\mathrm{O}(2^{nl})$	$\mathrm{O}(n2^n)$	Близкая к оптимальной	Неперебoрный; использует спектральный метод Радемахера — Уолша
A2.2	Гарантирован	$\mathrm{O}(n2^n)$	$\mathrm{O}(2^n)$	$\mathrm{O}(2^n)$	Неперебoрный; использует таблицу преобразования
A2.3	Гарантирован	$\mathrm{O}(l2^n)$	$\mathrm{O}(2^n)$	Близкая к оптимальной	Неперебoрный; использует таблицу преобразования
A2.4	Гарантирован	$\mathrm{O}(n^2 2^m)$	$\mathrm{O}(2^m)$	$\mathrm{O}(n^2 2^m)$	Неперебoрный; использует теорию групп подстановок

Из таблицы видно, что только алгоритмы A2.2 – A2.4 гарантированно дают результат синтеза схемы за приемлемое время. При этом лишь алгоритм A2.3 позволяет получить схему с близкой к оптимальной вентильной сложностью. Тем не менее в случае $m = \mathrm{o}(n)$ ни один из алгоритмов синтеза, кроме алгоритма A2.4, не позволяет получить результат синтеза за время порядка $\mathrm{O}(2^n)$ (аналогично для требуемого объёма памяти). Однако существенным недостатком алгоритма A2.4 является избыточная вентильная сложность синтезированной схемы по сравнению с другими алгоритмами.

3. Новый быстрый алгоритм синтеза

Рассмотрим произвольную чётную подстановку $h \in A_{2^n}$, $n > 3$, и множество подвижных точек этой подстановки $M = \{x : x \in \mathbb{Z}_{2^n}, h(x) \neq x\}$. Обозначим $m = \lceil \log_2 |M| \rceil$. В случае, когда подстановка h представляет собой один длинный цикл, её можно разложить в композицию не более чем $2^m - 1$ транспозиций:

$$(i_1, i_2, i_3, \ldots, i_{2^m}) = (i_1, i_2) \circ (i_1, i_3) \circ \ldots \circ (i_1, i_{2^m}). \quad (1)$$

В случае, когда h представляет собой композицию нескольких циклов, каждый цикл можно разложить в композицию транспозиций по формуле (1), что при фиксированном значении m даёт максимальное количество транспозиций не более $2^m - 1$.

Любой элемент $x \in \mathbb{Z}_{2^n}$ может быть представлен в виде $x = \sum\limits_{i=1}^{n} x_i \cdot 2^{i-1}$; $x_i \in \mathbb{Z}_2$ будем называть *разрядом* элемента x.

Самый простой способ А3.1 синтеза обратимой схемы, задающей подстановку h, можно описать следующим образом:

1) Разложить заданную подстановку h в композицию транспозиций.
2) Каждую транспозицию $t = (x, y)$ действием сопряжения привести к виду $t' = (x', y')$, где $x, y, x', y' \in \mathbb{Z}_{2^n}$ и существует такое значение j, что $x'_j = y'_j \oplus 1$, при этом для всех остальных разрядов выполняется условие $x'_i = y'_i = 1$, $i \neq j$.
3) Найти композиции обратимых вентилей NOT и k-CNOT, задающих транспозицию t' и действие сопряжением на транспозицию t.

Действие сопряжением подстановкой g на подстановку h задаётся как

$$h^g = g^{-1} \circ h \circ g.$$

Для любого обратимого вентиля E (NOT или k-CNOT) задаваемая им подстановка g является обратной к самой себе. Это следует из определения вентилей. Поэтому действие сопряжением в данном случае выражается как $h^g = g \circ h \circ g$. Таким образом, для действия сопряжением подстановкой, задаваемой обратимым вентилем E (NOT или k-CNOT), требуется ровно два вентиля E. Действие сопряжением не меняет цикловой структуры подстановки, поэтому транспозиция t в результате действия сопряжением всегда остаётся одной транспозицией.

Для транспозиции $t = (x, y)$ введём четыре множества: $B_{00} = \{i : x_i = y_i = 0\}$, $B_{01} = \{i : x_i = 0, y_i = 1\}$, $B_{10} = \{i : x_i = 1, y_i = 0\}$, $B_{11} = \{i : x_i = y_i = 1\}$. Мощности этих множеств обозначим b_{00}, b_{01}, b_{10}, b_{11}. Очевидно, что $b_{00} + b_{01} + b_{10} + b_{11} = n$; если $x \neq y$, то $b_{01} \neq 0$ или $b_{10} \neq 0$. Рассмотрим два случая.

1) $b_{01} \neq 0$, $b_{10} \neq 0$.

Пусть $j \in B_{10}$, $k \in B_{01}$. Для каждого $i \in B_{10}$, $i \neq j$, будем действовать сопряжением на t подстановкой, задаваемой вентилем $C_{k;i}$. Затем для каждого $i \in B_{01}$ будем действовать сопряжением на полученную транспозицию подстановкой, задаваемой вентилем $C_{j;i}$. На последнем шаге для каждого $i \in B_{00}$ будем действовать сопряжением на полученную транспозицию подстановкой, задаваемой вентилем N_i. В результате получим искомую транспозицию $t' = (x', y')$.

Для сопряжения требуется $2(b_{10} - 1)$ вентилей $C_{k;i}$, $2b_{01}$ вентилей $C_{j;i}$ и $2b_{00}$ вентилей N_i; всего $2(b_{10} + b_{01} + b_{00} - 1)$ вентилей NOT и 1-CNOT. В худшем случае $b_{10} + b_{01} + b_{00} = n$ (при $b_{11} = 0$). Следовательно, для получения транспозиции t' при $b_{01} \neq 0$, $b_{10} \neq 0$ требуется $2(n - 1)$ вентилей NOT и 1-CNOT.

2) $b_{01} = 0$ или $b_{10} = 0$.

Без ограничения общности рассмотрим только случай $b_{01} = 0$, $b_{10} \neq 0$. Пусть $j \in B_{10}$. Сначала действуем сопряжением на t подстановкой, задаваемой вентилем N_j. Затем для каждого $i \in B_{10}$, $i \neq j$, будем действовать сопряжением на полученную транспозицию подстановкой, задаваемой вентилем $C_{j;i}$. После этого вновь будем действовать сопряжением на полученную транспозицию подстановкой, задаваемой вентилем N_j. На последнем шаге для каждого $i \in B_{00}$ будем действовать сопряжением на полученную транспозицию подстановкой, задаваемой вентилем N_i. В результате получим искомую транспозицию $t' = (x', y')$.

Для сопряжения требуется $2(b_{10} - 1)$ вентилей $C_{j;i}$, четыре вентиля N_j и $2b_{00}$ вентилей N_i; всего $2(b_{10} + b_{00} + 1)$ вентилей NOT и 1-CNOT. В худшем случае $b_{10} + b_{00} = n$ (при $b_{11} = 0$). Следовательно, для получения транспозиции t' при $b_{01} = 0$ или $b_{10} = 0$ требуется $2(n + 1)$ вентилей NOT и 1-CNOT.

Транспозиция t' задается вентилем $C_{I;j}$ с $n - 1$ управляющими входами из множества $I = \{1, 2 \ldots, n\} \backslash \{j\}$. Следовательно, для второго и третьего шагов алгоритма A3.1 требуется не более $2(n + 1) + 1$ вентилей NOT и k-CNOT.

Умножая максимально возможное количество транспозиций на сложность реализации одной транспозиции, получаем, что вентильная сложность схемы, синтезированной алгоритмом A3.1, не превосходит $(2^m - 1)(2(n + 1) + 1) = \mathrm{O}(n2^m)$. Временная сложность алгоритма составляет порядка $\mathrm{O}(n2^m)$, требуемый для синтеза объём памяти равен $\mathrm{O}(2^m)$ (хранение всех элементов подстановки). Недостатком данного алгоритма является использование вентилей $(n - 1)$-CNOT, что в некоторых случаях является недопустимым, так как такой вентиль нельзя заменить на композицию вентилей 2-CNOT без использования дополнительных входов схемы [3].

Усовершенствованный итоговый алгоритм A3.2 синтеза обратимых схем из вентилей NOT, 1-CNOT и 2-CNOT, предлагаемый в данной работе, основан на доказательстве теоремы из [9], согласно которой множество подстановок, задаваемых вентилями NOT, 1-CNOT и 2-CNOT с n входами, генерирует знакопеременную группу A_{2^n} при $n > 3$.

Композицию двух независимых циклов можно выразить следующим образом:

$$(i_1, i_2, \ldots, i_{k_1}) \circ (j_1, j_2, \ldots, j_{k_2}) = (i_1, i_2) \circ (j_1, j_2) \circ (i_1, i_3, \ldots, i_{k_1}) \circ (j_1, j_3, \ldots, j_{k_2}). \quad (2)$$

Цикл длины $k \geqslant 5$ можно выразить как

$$(i_1, i_2, \ldots, i_k) = (i_1, i_2) \circ (i_3, i_4) \circ (i_1, i_3, i_5, i_6, \ldots, i_k). \quad (3)$$

Следовательно, имея исходное разложение чётной подстановки в композицию независимых циклов и используя формулы (2) и (3), эту подстановку можно выразить в виде композиции пар транспозиций, из которых только одна будет парой зависимых транспозиций, остальные — независимых. Максимально возможное количество транспозиций в представлении подстановки h не превосходит $2^m - 1$, следовательно, количество пар независимых транспозиций не превосходит 2^{m-1}.

Рассмотрим пару независимых транспозиций $p = (x, y) \circ (z, w)$. Действие сопряжением не меняет цикловой структуры подстановки, поэтому p в результате действия сопряжением всегда будет оставаться парой независимых транспозиций. Применяя такие же рассуждения, как и для транспозиции $t = (x, y)$, приведём пару p действием сопряжением к виду $p' = (2^n - 1, 2^n - 1 - 2^{i_1}) \circ (z', w')$, где i_1 — индекс разряда, в котором различаются элементы первой транспозиции (во всех остальных разрядах 1),

а (z', w') — новая транспозиция, получившаяся в результате действия сопряжением. Для этого шага потребуется не более $2(n+1)$ обратимых вентилей. Затем, применяя такой же подход для элементов (2^n-1) и z' из пары p', получаем в результате действия сопряжением новую пару $p'' = (2^n-1, 2^n-1-2^{i_1}) \circ (2^n-1-2^{i_2}, w'')$, где i_2 — индекс разряда, в котором различаются первые элементы первой и второй транспозиций (во всех остальных разрядах 1), а w'' — новый элемент второй транспозиции, получившийся в результате действия сопряжением. Для этого шага также потребуется не более $2(n+1)$ обратимых вентилей.

Покажем, как можно действием сопряжением привести пару p'' к виду $q = (2^n-1, 2^n-1-2^{i_1}) \circ (2^n-1-2^{i_2}, 2^n-1-2^{i_1}-2^{i_2})$. Рассмотрим два случая.

1) $w''_{i_1} = w''_{i_2} = 0$.

В этом случае сначала действуем сопряжением на p'' подстановками, задаваемыми вентилями N_{i_1} и N_{i_2}. Затем для каждого i, такого, что $w''_i \neq 1$, $i \neq i_1, i_2$, действуем сопряжением на полученную пару транспозиций подстановкой, задаваемой вентилем $C_{i_1,i_2;i}$. После этого вновь действуем сопряжением на полученную пару транспозиций подстановками, задаваемыми вентилями N_{i_1} и N_{i_2}.

Для сопряжения требуется не более $2(n-2)$ вентилей $C_{i_1,i_2;i}$ и по четыре вентиля N_{i_1} и N_{i_2}. Следовательно, для получения пары транспозиций q при $w''_{i_1} = w''_{i_2} = 0$ требуется не более $2(n+2)$ вентилей NOT и 2-CNOT.

2) $w''_{i_1} = 1$ или $w''_{i_2} = 1$ (в том числе и одновременно).

Поскольку w'' не равно ни одному из остальных элементов пары транспозиций p'', существует такой индекс i_3, что $w''_{i_3} = 0$. Действуем сопряжением на p'' подстановкой, задаваемой вентилем N_{i_3}. Затем действуем сопряжением на полученную пару транспозиций подстановкой, задаваемой вентилем $C_{i_3;i_1}$, если $w''_{i_1} = 1$, и подстановкой, задаваемой вентилем $C_{i_3;i_2}$, если $w''_{i_2} = 1$. После этого вновь действуем сопряжением на полученную пару транспозиций подстановкой, задаваемой вентилем N_{i_3}, и приходим к случаю 1.

Для сопряжения требуется не более четырёх вентилей 2-CNOT ($C_{i_3;i_1}$ и $C_{i_3;i_2}$), четыре вентиля N_{i_3} и не более $2(n+2)$ вентилей NOT и 2-CNOT (при переходе к случаю 1). Следовательно, для получения пары транспозиций q при $w''_{i_1} = 1$ или $w''_{i_2} = 1$ требуется не более $2(n+6)$ вентилей NOT и 2-CNOT.

Пара независимых транспозиций q задаётся обратимым вентилем $C_{I;j}$, где $I = \{1, 2, \ldots, n\} \setminus \{i_1, i_2\}$. Поскольку $|I| = n-2$, вентиль $C_{I;j}$ можно заменить на композицию не более чем $8(n-5)$ вентилей 2-CNOT без использования дополнительных входов схемы [3]. Таким образом, суммарную вентильную сложность реализации пары независимых транспозиций можно оценить как

$$L(p_{\text{indep}}) \leqslant 4(n+1) + 2(n+6) + 8(n-5) = 14n + \mathrm{O}(1).$$

Рассмотрим пару зависимых транспозиций $p = (x, y) \circ (x, z)$. Такую пару можно выразить через композицию пар независимых транспозиций:

$$(x, y) \circ (x, z) = ((x, y) \circ (a, b)) \circ ((a, b) \circ (x, z)).$$

Следовательно, суммарную вентильную сложность реализации пары зависимых транспозиций можно оценить как

$$L(p_{\text{dep}}) \leqslant 2L(p_{\text{indep}}) \leqslant 28n + \mathrm{O}(1).$$

Таким образом, умножая максимально возможное количество пар транспозиций одного типа (зависимых или независимых) в представлении подстановки h на вентильную сложность реализации этого типа пары транспозиций, получаем, что вентильная сложность обратимой схемы $\mathfrak{S}$, синтезированной алгоритмом A3.3, не превосходит следующей величины:

$$L(\mathfrak{S}) \leqslant 2^{m-1}L(p_{\text{indep}}) + L(p_{\text{dep}}) \leqslant 2^{m-1}(14n + \mathrm{O}(1)) + 28n + \mathrm{O}(1) \lesssim 7n2^m,$$

где знак $\lesssim$ означает следующее: $f \lesssim g$, если $\lim\limits_{n\to\infty} \sup f(n)/g(n) \leqslant 1$.

Временная сложность алгоритма A3.2 составляет $\mathrm{O}(n2^m)$, а требуемый для синтеза объём памяти равен $\mathrm{O}(2^m)$ (хранение всех элементов подстановки).

Предложенные алгоритмы синтеза A3.1 и A3.2 обладают одним существенным недостатком: вентильная сложность синтезированной схемы зависит только от m, но никак не зависит от вида конкретной подстановки h. Например, рассмотрим преобразование $f\left(\langle x_1, x_2, \ldots, x_n\rangle\right) = \langle x_1, x_2 \oplus x_1, x_3, \ldots, x_n\rangle$. Соответствующую подстановку h_f можно задать одним вентилем $C_{1;2}$. При этом количество подвижных точек подстановки h_f равно половине от всех элементов $\mathbb{Z}_{2^n}$ (для которых $x_1 = 1$), т. е. $m = n - 1$. Отсюда следует, что для всех преобразований, схожих с f, алгоритмы A3.1 и A3.2 будут синтезировать схему с вентильной сложностью порядка $\mathrm{O}(n2^{n-1})$.

Заключение

В работе рассмотрены различные алгоритмы синтеза обратимых схем и их основные характеристики. Для частного случая, когда заданное преобразование можно описать чётной подстановкой $h \in A_{2^n}$, $n > 3$, для которой $m = \lceil \log_2 |M| \rceil = \mathrm{o}(n)$, где M — множество подвижных точек, показано, что лишь алгоритмы синтеза, основанные на теории групп подстановок, могут гарантированно синтезировать схему за время $\mathrm{O}(2^n)$.

Предложенный быстрый алгоритм A3.2 синтеза обратимых схем на основе теории групп подстановок имеет на порядок меньшие временную сложность ($\mathrm{O}(n2^m)$ против $\mathrm{O}(n^2 2^m)$) и вентильную сложность синтезированной схемы ($\mathrm{O}(n2^m)$ против $\mathrm{O}(n^2 2^m)$) по сравнению с существующим алгоритмом A2.4, сохраняя при этом такой же объём памяти, требуемый для синтеза — $\mathrm{O}(2^m)$. Вместе с тем алгоритм A3.2 может синтезировать схему из вентилей NOT, 1-CNOT и 2-CNOT без использования дополнительных входов схемы, в отличие от алгоритма A3.1.

К сожалению, вентильная сложность схемы, синтезированной алгоритмами A3.1 и A3.2, зависит только от m, но никак не зависит от вида подстановки h. Направлением дальнейших исследований является изучение возможности минимизации вентильной сложности синтезированной схемы для некоторого класса чётных подстановок.

ЛИТЕРАТУРА

1. *Bennett C. H.* Logical reversibility of computation // IBM J. Res. Dev. 1973. V. 17. P. 525–532.
2. *Khlopotine A. B., Perkowski M. A., and Kerntopf P.* Reversible logic synthesis by iterative compositions // Int. Workshop Logic Synthesis, New Orleans, Louisiana, June 4–7, 2002. P. 261–266.
3. *Shende V. V., Prasad A. K., Markov I. L., and Hayes J. P.* Synthesis of reversible logic circuits // IEEE Trans. CAD. 2003. V. 22. No. 6. P. 710–722.
4. *Yang G., Song X., Hung W. N., and Perkowski M. A.* Fast synthesis of exact minimal reversible circuits using group theory // Proc. ASP DAC'05, Shanghai, China, January 18–21, 2005. P. 1002–1005.

5. *Miller D. M. and Dueck G. W.* Spectral techniques for reversible logic synthesis // 6th Int. Symp. Representations and Methodology of Future Comput. Technol., Trier, Germany, 2003. P. 56–62.
6. *Miller D. M., Maslov D., and Dueck G. W.* A transformation based algorithm for reversible logic synthesis // Design Automation Conference (DAC), Anaheim, CA, 2003. P. 318–323.
7. *Saeedi M., Sedighi M., and Zamani M. S.* A novel synthesis algorithm for reversible circuits // Int. Conf. on Computer-Aided Design (ICCAD), USA, 2007. P. 65–68
8. *Yang G., Song X., Hung W. N., et al.* Group theory based synthesis of binary reversible circuits // 3rd Annual Conf. Theory Appl. of Models of Comput. (TAMC), Beijing, China, 2006. P. 365–374
9. *Закаблуков Д. В., Жуков А. Е.* Исследование схем из обратимых логических элементов // Информатика и системы управления в XXI веке: сб. трудов № 9 молодых ученых, аспирантов и студентов. М.: МГТУ им. Н. Э. Баумана, 2012. С. 148–157.

DOI 10.17223/20710410/24/10

УДК 621.374.3; 004.4:004.9

О ЗАЩИТЕ ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ В ПРОЦЕССЕ ПРОЕКТИРОВАНИЯ УСТРОЙСТВ НА ОСНОВЕ FPGA XILINX

Д. И. Черемисинов

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

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

Рассматривается влияние задачи охраны интеллектуальной собственности на процесс проектирования устройств на основе FPGA Xilinx. Лицензионное соглашение САПР FPGA Xilinx запрещает использование инструментов этой САПР для перепроектирования, поэтому обсуждаются программно-технические средства обеспечения контроля соблюдения таких соглашений пользователями САПР. Показано, что существует класс устройств на основе FPGA, конфигурационный файл которых принципиально не поддаётся декомпиляции. Этот класс составляют, в частности, устройства с частичной динамической реконфигурацией посредством внутреннего контроллера самореконфигурации.

Ключевые слова: *перепроектирование, FPGA, защита интеллектуальной собственности.*

Введение

Результаты интеллектуальной деятельности имеют стоимостные оценки, как и прочие продукты человеческого труда, так как они могут быть включены в товарооборот на коммерческих условиях и давать полезный эффект (экономический, социальный и т. п.). Защита современных цифровых электронных устройств необходима для предотвращения экономических потерь от кражи интеллектуальной собственности, воплощённой в функциональных возможностях устройства. Возможность кражи обеспечивается при копировании устройства (cloning) или путём его перепроектирования (reverse engineering). В то время как при копировании целью является создание возможно более точной копии оригинального устройства, при перепроектировании нарушитель извлекает алгоритм, реализованный в устройстве, и затем улучшает, изменяет и маскирует его при сохранении функционирования.

Программируемое устройство — это электронный компонент, позволяющий строить реконфигурируемые цифровые схемы. В отличие от элементов с фиксированной функцией, функция реконфигурируемого элемента не определена во время изготовления, и прежде чем использоваться в схеме, элемент должен быть запрограммирован, то есть должна быть задана его конфигурация в виде последовательности конфигурационных битов (`bitstream`).

Программируемые устройства делятся на энергонезависимые и энергозависимые. Энергозависимые устройства не сохраняют битов конфигурации, и конфигурационный файл нужно загружать из внешней памяти при каждом включении питания. В FPGA конфигурационный файл обрабатывается логикой программирования, которая сама не программируема. Реконфигурируемую часть FPGA составляет пользовательская логика (user logic) — она реализует заданное пользователем поведение устройства. Большинство современных FPGA являются энергозависимыми.

Продуктом интеллектуальной деятельности при проектировании FPGA является информация о способах достижения требуемого поведения разработанного устройства (пользовательская логика). Сама информация имеет нематериальный характер, и интеллектуальным продуктом она становится, будучи воплощённой в объективированную форму — техническую документацию, тесты, описания на формальном языке, которые служат исходными данными для САПР и т. д. Опасность — возможность возникновения ситуации, при которой можно понести ущерб. Информационная безопасность обеспечивается защитой информации. Защита информации должна обеспечивать предотвращение ущерба в результате утери (хищения, утраты, искажения, подделки) информации в любом её виде. Угроза — это потенциальная возможность определённым образом нарушить защиту информации. Попытка реализации угрозы называется атакой, а тот, кто предпринимает такую попытку, — злоумышленником (malicious). В то время как информационная безопасность — это состояние защищённости информационной среды, защита информации представляет собой деятельность по предотвращению опасностей, то есть процесс, направленный на достижение этого состояния и состоящий в помещении защищаемой собственности в более безопасное место. Основным принципом противодействия угрозам безопасности информации является превентивность принимаемых мер защиты, так как устранение последствий проявления угроз требует значительных финансовых, временных и материальных затрат. Информационная безопасность — социальная проблема, потому что безопасность бессмысленна, пока человек не определит то, что собой представляет опасность. По характеру обеспечения информационной безопасности выделяют: законодательную деятельность; административную (приказы и другие действия руководства организаций, связанных с защищаемыми информационными системами); процедурную (меры безопасности, ориентированные на людей); программно-техническую. Программно-технические методы защиты интеллектуальной собственности являются существенной частью процедуры проектирования на FPGA.

В FPGA объектом защиты является интеллектуальная собственность, воплощённая в конфигурационном файле. Эта собственность создается производителем БИС FPGA и САПР для её программирования, производителями компонентов (IP cores) и разработчиком устройства.

1. Владельцы интеллектуальной собственности

На рынке энергозависимых FPGA доминируют три продавца БИС FPGA и одновременно производители САПР для программирования своих БИС: Altera, Lattice и Xilinx, каждый из которых через 12–18 месяцев представляет на рынок новое семейство FPGA. Все эти продавцы FPGA не являются владельцами производственных мощностей — они только проектируют FPGA, изготовляют БИС другие компании — «кремневые фабрики». Продавцы FPGA имеют две основные проблемы защиты своей интеллектуальной собственности. Во-первых, они должны защитить топологию и технологию самой БИС FPGA от перепроектирования, копирования или изменений. Во-вторых, они должны обеспечить клиентам возможность защитить их интеллектуальную собственность в ходе проектирования и при эксплуатации у конечного пользователя. Продавцы FPGA заинтересованы в интеграции и всемерном расширении использования компонентов от независимых источников, так как это стимулирует рынок и ведёт к увеличению продаж FPGA и САПР. В то же время продавцы FPGA и компонентов рассматривают разработчиков устройства как угрозу своей интеллектуальной

собственности и принимают меры по её защите. Противником всех троих владельцев интеллектуальной собственности является конечный пользователь устройства.

Разработка компонентов, представляющих собой готовые функциональные описания, является значительным рынком в области производства электронных устройств. Покупая и объединяя их в собственный проект, разработчики устройства экономят в стоимости и времени проектирования. Имеются компоненты таких размеров, что могут занять FPGA целиком. Компоненты распространяются как модули на языке описания аппаратуры или в синтезированном виде как сеть логических элементов. Прибыль от продажи компонентов получается из платежей продавцов FPGA, которые закладывают затраты на разработку собственных САПР и компонентов сторонних разработчиков в цену БИС.

Проблемой производителя компонентов является защита против не имеющего лицензию разработчика устройства. Компоненты лицензируются, распространяются и защищаются теми же способами, как и программное обеспечение. Некоторые поставщики зашифровывают свои компоненты и поставляют специальные инструменты разработки, которые их могут обработать. Недостаток такой защиты — сложность интеграции в существующие САПР и ненадёжность: защита держится на программных инструментах, которые могут быть взломаны.

Другой подход состоит в том, чтобы спрятать в компоненте специальный код, так называемую `watermark`, которая может использоваться для проверки источника собственности. Предложено много подходов к осуществлению инкапсуляции `watermark` в компонент. Большинство из них неприменимо из-за отсутствия реальной возможности проверки `watermark`, которая должна основываться только на исследовании устройств, купленных конечным пользователем. Если от обвиняемого разработчика требуется дополнительная информация, проверка не будет юридически значимой.

Атака на устройство с FPGA выполняется не только для кражи интеллектуальной собственности, воплощённой в функциональных возможностях устройства, но и для нанесения ущерба легальному пользователю устройства. Игнорируя последний случай, можно считать, что противником владельцев собственности является пользователь (недобросовестный). В энергозависимых FPGA коммуникация между FPGA и памятью может быть записана, что даёт возможность перехватить конфигурационный файл. Таким образом, из всех видов атак [1] на цифровые СБИС для FPGA наиболее простой оказывается атака на основе конфигурационного файла. Существующие в настоящее время методы защиты конфигурационного файла [1] имеют тот главный недостаток, что ведут к росту энергопотребления и снижению надёжности и скорости работы. Распространена ситуация, когда защита от перехвата конфигурационного файла не оправдывает затраты.

Для защиты информационных ресурсов широко используются технические средства защиты авторских прав (DRM — Digital Rights Management), которые затрудняют создание копий защищаемых произведений, распространяемых в электронной форме. Обычно в качестве технических средств защиты информации используется её кодирование или шифрование. Хотя эти технические средства призваны воспрепятствовать лишь неправомерному копированию произведений, они не допускают либо ограничивают любое копирование, в том числе добросовестное, поскольку невозможно техническими средствами отличить «законное» копирование от «незаконного». Задача разрешить воспроизведение и в то же время запретить копирование представляет собой принципиально неразрешимую проблему, так как если возможно воспроизведение информации, то возможно и её последующее копирование. В процессе проектирования

FPGA кодирование информации получается «даром», если формат результата проектирования является секретом.

Сам по себе конфигурационный файл не представляет ценности для взломщика, но, декомпилируя его в процессе перепроектирования, можно получить представление исходного алгоритма функционирования, пригодное для изменения и переработки. Предотвращение перепроектирования и представляет собой основную линию обороны владельцев интеллектуальной собственности. Известно изречение: «there is no security in obscurity» («упрятывание — не защита»). Однако скрытие информации остаётся действенным способом защиты интеллектуальной собственности в области проектирования на FPGA. В настоящее время все продавцы FPGA держат формат конфигурационного файла своих устройств в секрете. Кроме того, лицензионное соглашение САПР проектирования FPGA прямо запрещает любое их использование для перепроектирования.

2. Обратное проектирование

Процесс обратного проектирования является существенной частью создания конкурентоспособной продукции и обычно служит средством разработки устройств, более эффективных, чем имеющиеся у конкурентов. Побочная область применения обратного проектирования — перепроектирование на современной базе устарелых компонентов, находящихся в составе долговечного оборудования, такого, как военные или космические системы, ядерные реакторы, авиалайнеры и морские суда.

Обычная разработка — это процесс превращения спецификации в продукт, удовлетворяющий этой спецификации. Между спецификацией и продуктом находятся процессы разработки и изготовления, в которых необходим некоторый человеческий творческий потенциал и автоматизация. Процесс разработки удобно определять в терминах «уровней» абстракции. Считается, что разработка включает построение следующих представлений продукта: письменная функциональная спецификация для человеческого потребления — описание «верхнего уровня»; абстрактное структурное описание (текст описания поведения устройства — исходные данные для САПР) — описание «промежуточного уровня»; детализированное структурное описание (сеть элементов) для машинного потребления, но, возможно, постижимое людьми — описание «низкого уровня»; продукт (конфигурация настройки FPGA), строение которого обычно не воспринимается невооруженными органами чувств человека, — самый низкий уровень.

Обратное проектирование в общем случае состоит из следующих стадий: анализа продукта; извлечения описания продукта промежуточного уровня; анализа описания продукта интеллектом человека для построения новой спецификации; разработки нового продукта с использованием построенной спецификации. Обратное проектирование является инверсией обычной разработки в смысле порядка процесса преобразований; его задача заключается в построении спецификации на основе анализа продукта. Результат обратного проектирования не гарантирован, в общем случае невозможно даже в теории построить оригинальную спецификацию, изучая только продукт.

В области разработки полупроводниковых приборов об обратном проектировании в законе США «Semiconductor Chip Protection Act» говорится, что допускается «обратное проектирование масок или схем с целью обучения, анализа или оценки решений или методик...». Подобные законодательства имеют Япония, Европейский союз и другие страны [2].

Изготовители FPGA не разглашают кодирование `bitstream`, хотя они обеспечивают разработчиков инструментами для их обработки; значительное количество инфор-

мации о bitstream содержится в различных документах. Несмотря на актуальность этой информации, сообщения об успешном обратном проектировании FPGA bitstream отсутствуют. Хотя в 1990-х годах появилось сообщение [3] о раскрытии секрета кодирования bitstream Xilinx в компании «NeoCad», однако, согласно сообщению её руководителей, было декомпилировано программное обеспечение Xilinx для построения bitstream, само кодирование bitstream осталось нераскрытым [4]. Основываясь на сложности задачи обратного проектирования bitstream, большинство разработчиков устройств на FPGA, как и компьютерные программисты, игнорируют риск кражи интеллектуальной собственности путём перепроектирования. Задача декомпиляции bitstream, извлечённого из энергонезависимой памяти в готовом устройстве, безнадёжна. Но если перепроектирование выполняется организацией — разработчиком оригинального устройства на FPGA (например, с целью использования более дешёвой в массовом производстве технологии изготовления), то задача декомпиляции в некоторых случаях может быть решена.

3. Защита САПР

Без программных инструментов автоматизированного проектирования и заложенных в них алгоритмов проектирование FPGA невозможно. Эта интеллектуальная собственность нуждается в особых технических средствах защиты авторских прав. Задачу защиты инструментов САПР техническими средствами можно поставить следующим образом. Дана проблема P и её решение S, нужно определить, получена ли S заданным программным инструментом или алгоритмом. Цель защиты интеллектуальной собственности, воплощённой в структуре устройства, состоит в том, чтобы обеспечить механизм, позволяющий проверить, с определённой степенью доверия, является ли подозрительная часть этой интеллектуальной собственности дубликатом (частичным или полным) другой интеллектуальной собственности. В защите инструментов САПР цель состоит в том, чтобы обеспечить возможность проследить использование этой САПР. Другими словами, нужно определить, использовались ли определённые инструменты САПР (или алгоритмы) при проектировании подозрительного устройства.

Технические средства защиты САПР, предотвращающие использование её инструментов с нарушением лицензионного соглашения, трудно осуществимы и практически не применяются из-за небольшого (в сравнении с рынком компиляторов языков программирования) числа покупателей САПР FPGA. Экономически оправдано использование технических средств обнаружения использования инструментов САПР при проектировании конкретного устройства. Для этого в результат проектирования нужно встроить watermark, подтверждающую использование инструментов защищаемой САПР. Сам формат bitstream служит гарантией его получения средствами защищаемой САПР, и в случае, когда целевой платформой является именно FPGA, использование таких watermark излишне. Для защиты от перепроектирования watermark должна сохраняться после декомпиляции bitstream. С другой стороны, сам процесс проектирования сложен, и есть ситуации, когда невозможно техническими средствами отличить «законное» использование инструмента САПР от «незаконного».

Декомпиляция конфигурационного файла требуется в ходе отладки устройства на FPGA. Отладка устройств нацелена на устранение ошибок проектирования и состоит в выполнении тестов. Тесты разрабатываются таким образом, чтобы продемонстрировать работоспособность устройства. В свою очередь, устройство работоспособно только при вполне определённых условиях. Проектировщик не в состоянии заранее предусмотреть все необходимые для исправной работы условия. Одной из задач тестирова-

ния является выявление этих условий. Окончательный вывод о необходимых условиях работоспособности реального устройства обычно даёт натурный эксперимент. Однако из-за сложности современных FPGA натурные эксперименты очень дороги и приходится обходиться моделированием. Для моделирования поведения FPGA с загруженным конфигурационным файлом необходимо построение модели из примитивов FPGA, для удобства она строится на языке описания аппаратуры. Построение этой модели по сути представляет собой декомпиляцию.

Например, в САПР Xilinx [5] конфигурационный файл является одним из представлений базы данных проекта (файл NCD), содержащим алгоритм функционирования в форме размещённых примитивов FPGA с трассированными соединениями, и строится программой `bitgen`. Программой `netgen` можно построить представление базы данных на языке VHDL, являющейся декомпиляцией конфигурационного файла. Исправления, возникающие в ходе отладки, можно выполнять, изменяя базу данных преобразованием в ASCII-формат XDL [6] и назад.

4. Проблема декомпиляции

В криптографии односторонняя функция — это эффективно вычислимая функция, для задачи инвертирования которой не существует эффективных алгоритмов [7]. Под инвертированием понимается массовая задача нахождения по заданному значению функции одного (любого) значения из прообраза (заметим, что обратная функция, вообще говоря, может не существовать). Декомпиляция конфигурационного файла является обратной функции компиляции конфигурационного файла. Если для функции компиляции конфигурационного файла по заданному исходному представлению (скажем, на языке VHDL) не существует обратной функции, то таким преобразованием обеспечена абсолютная защита конфигурационного файла. Аналитическое представление функции компиляции сложно (фактически это программа компилятора), поэтому теоретическая разработка обратной функции неосуществима. Возможность построения этой функции можно оценить по аналогии с декомпиляцией программ и методом проб и ошибок.

В литературе нет сведений, что проблема декомпиляции конфигурационного файла кем-либо была решена. С другой стороны, известно [1], что информацию о настройке LUT можно извлечь непосредственно из конфигурационного файла. Исследование проблемы декомпиляции программ показало [8], что в общем случае декомпиляция невозможна, потому что декомпиляция самомодифицирующихся программ представляет собой алгоритмически неразрешимую проблему. Относительно декомпиляции конфигурационного файла такого типа результаты в литературе неизвестны.

Эксперименты по декомпиляции представления конфигурационного файла в формате XDL в представление на VHDL показывают, что декомпиляция для важных частных случаев — когда исходное представление на VHDL задаёт комбинационную схему — осуществима практически.

5. Динамическая частичная реконфигурация

Частичная реконфигурация (Partial Reconfiguration — PR) — это возможность реконфигурировать (повторно запрограммировать) часть FPGA, в то время как остальная её часть остаётся неизменной и продолжает функционировать. При частичной реконфигурации в FPGA загружается только часть `bitstream`, а не весь `bitstream` целиком. Динамической эта частичная реконфигурация называется потому, что позволяет перепрограммировать часть FPGA «на лету», в процессе нормального функ-

ционирования, не затрагивая остальной части системы. В противоположность этому статическая реконфигурация состоит в перепрограммировании FPGA, когда она находится в состоянии сброса [9]. В FPGA Xilinx допускается частичная реконфигурация устройств, начиная от Spartan-3 и всех последующих FPGA.

САПР FPGA Xilinx имеет два основных метода частичной реконфигурации, называемых модульной (module-based) и дифференциальной реконфигурацией (difference-based). Развитые средства автоматизации проектирования имеются только для модульной реконфигурации [10]. При модульной реконфигурации в сетке элементов FPGA выделяются области (PR region), которые могут быть перепрограммированы, а само устройство описывается в виде набора модулей, причём для каждой области перепрограммирования указывается набор оверлейных модулей (PR module), которые будут сменять друг друга при перепрограммировании этой области. Каждый оверлейный модуль размещается и трассируется в расчёте на помещение в указанную область; `bitstream` оверлейных модулей записывается в конфигурационный файл вслед за основным `bitstream`. Эти `bitstream` используются в процессе динамической реконфигурации по одному для каждой перепрограммируемой области FPGA.

Размер полного `bitstream` для Spartan-3 порядка 0,5 Мб, поэтому временные затраты на статическую реконфигурацию значительны, так как этот объём данных нужно прочитать из энергонезависимой памяти. Размер `bitstream` оверлейных модулей пропорционален числу конфигурируемых ресурсов — размеру области перепрограммирования, поэтому время, затрачиваемое на динамическую реконфигурацию, значительно меньше времени статической реконфигурации.

При статической реконфигурации соединения между модулями размещаются так же, как и другие соединения. При динамической реконфигурации соединения оверлейных модулей имеют ограничения, они должны использовать те же самые физические провода. Это обеспечивается шинным макросом (bus macro). Шинный макрос в сетке элементов FPGA — это область трассированных соединений, обеспечивающая всем оверлейным модулям одни и те же каналы коммуникации. Реализация шинного макроса обеспечивается САПР Xilinx для каждого типа FPGA.

Частичная реконфигурация осуществляется под управлением внешнего процессора через интерфейс JTAG, когда FPGA находится в состоянии сброса, или под внутренним управлением с помощью специальной логики или встроенного процессора, например PowerPC в FPGA типа Virtex II Pro. Самореконфигурация, то есть частичное перепрограммирование посредством внутренней схемы, устраняет потребность во внешнем процессоре. Самореконфигурация в Xilinx FPGA обычно требует использования интерфейса ICAP (Internal Configuration Access Port). Этот интерфейс обеспечивается встроенным в FPGA процессором или создаётся в FPGA специальным статическим модулем. Этот статический модуль является реализацией конечного автомата, который выполняет чтение частичных `bitstream` из энергонезависимой памяти и передачу данных через интерфейс ICAP, устраняя нужду в процессоре. Частичный `bitstream` содержит всю необходимую информацию о реконфигурации заданной области FPGA. То, что формат `bitstream` является секретом, усложняет аппаратуру самореконфигурации и процесс проектирования динамически реконфигурируемого устройства.

Перепроектирование устройства на FPGA с частичной динамической реконфигурацией требует выделения контроллера самореконфигурации и его удаления, модификации поведения статических модулей, модификации оверлейных модулей в статические и изменения функционирования шинного макроса. Значительную часть этой работы трудно выполнить автоматически. Такие устройства похожи на самомодифицирующи-

еся программы, следовательно, в общем случае декомпиляция `bitstream` принципиально невозможна.

Заключение

Область задач охраны интеллектуальной собственности при проектировании устройств на FPGA очень широка и включает технологические и архитектурные аспекты защиты, уязвимости, специфические для FPGA, ограничения возможностей устройств на FPGA в отношении защиты, проблемы создания технических средств защиты устройств на FPGA и САПР для их проектирования.

Доминирующим подходом для защиты интеллектуальной собственности, воплощённой в FPGA, является методика «водяных знаков» — `watermark`. Предложены методики для включения `watermark` в описания на всех уровнях абстракции; в большинстве из них используются побочные каналы. Побочные каналы исследования устройства основаны на использовании информации о физических процессах в устройстве: задержке, энергопотреблении, электромагнитном излучении, шуме основания, температуре. Эти методики не подходят для создания `watermark`, сохраняющихся при перепроектировании. Создание водяных знаков является намного более трудной задачей для FPGA, чем для интегральных схем специального назначения.

Создателями интеллектуальной собственности при проектировании FPGA являются разработчики программных инструментов автоматизированного проектирования и заложенных в них алгоритмов. Самый радикальный способ защиты САПР — это дистанционное управление, когда продавец САПР может удалённо управлять тем, какое действие может или не может выполнить пользователь. Однако этот метод пока не нашёл широкого распространения.

Сложность задачи обратного проектирования для FPGA признаётся одним из преимуществ FPGA по сравнению с другими реализациями с точки зрения защиты интеллектуальной собственности [11]. Задача декомпиляции `bitstream` в общем случае принципиально неразрешима. Однако эксперименты по декомпиляции представления конфигурационного файла в формате XDL в представление на VHDL показывают, что декомпиляция для важного частного случая — когда исходное представление на VHDL задает схему из триггеров и логических элементов — осуществима практически.

ЛИТЕРАТУРА

1. *Wollinger T., Guarjardo J., and Paar C.* Security on FPGAs: State-of-the-Art implementations and attacks // ACM Trans. Embedded Comput. Systems. 2004. V. 3. No. 3. P. 534–574.
2. *Musker D. C.* Reverse engineering // Protecting & Exploiting Intellectual Property in Electronics, IBC Conferences, 10 June 1998. `www.jenkins-ip.com/serv/serv_6.htm`
3. *Wollinger T. and Parr C.* How secure are FPGAs in cryptographic applications // LNCS. 2003. V. 2887. P. 91–100.
4. *Trimberger S.* Trusted design in FPGAs // Proc. 44th Annual Design Automation Conf. (DAC'07). N. Y.: ACM, 2007. P. 5–8.
5. *Note J.-B. and Rannaud E.* From the bitstream to the netlist // Proc. 16th Int. ACM/SIGDA Symp. on FPGA. N. Y.: ACM, 2008. P. 264–264.
6. *Beckhoff C., Koch D., and Torresen J.* The Xilinx Design Language (XDL): Tutorial and use cases // 6th Int. Workshop ReCoSoC'11, Montpellier, France, 2011. P. 1–8.
7. Введение в криптографию / под ред. В. В. Ященко. СПб.: Питер, 2001. 288 с.
8. *Cifuentes C. and Gough K. J.* Decompilation of binary programs // Software — Practice & Experience. 1995. V. 25. No. 7. P. 811–829.

9. *Beckhoff C., Koch D., and Torresen J.* Go ahead: a partial reconfiguration framework // 20th Annual IEEE Int. Symp. on Field-Programmable Custom Comp. Machines, April 29 – May 1, 2012, Toronto, Canada. P. 27–44.
10. Partial Reconfiguration User Guide UG702 (v13.1). Xilinx Inc., March 1, 2011. 124 p.
11. *Majzoobi M., Koushanfar F., and Potkonjak M.* FPGA-oriented security // Introduction to Hardware Security and Trust / eds. M. Tehranipour and M. Wang. Springer, 2011. P. 195–231.

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/24/11

УДК 519.7

ОДНОВРЕМЕННЫЙ ПОИСК НЕСКОЛЬКИХ ДВОИЧНЫХ ШАБЛОНОВ В ПОТОКЕ С ПОМОЩЬЮ КОНЕЧНОГО АВТОМАТА

И. В. Панкратов

г. Томск, Россия

E-mail: ivan.pankratov2010@yandex.ru

Рассматривается задача поиска булевых векторов в потоке данных. Предлагается метод построения конечного автомата, который ищет одновременно несколько векторов, совершая только две простые операции на каждый бит или группу битов, например байт данных. При этом с увеличением количества искомых шаблонов объём требуемой памяти растёт медленнее, чем суммарная длина шаблонов, а трудоёмкость не изменяется совсем. Приводятся оценки размеров таблиц переходов и выходов автомата. Рассматриваются известные подходы к решению этой задачи. Есть возможность обобщить алгоритм построения поискового автомата на поиск не полностью определённых булевых векторов, однако в этом случае объём требуемой памяти может превышать найденную в данной работе оценку.

Ключевые слова: *поиск битовых последовательностей, синхропосылка, поиск подстроки, КМП-поиск, алгоритм Ахо — Корасик.*

Введение

Рассматриваемую задачу можно сформулировать так: имеем двоичную последовательность (поток данных) и набор из n булевых векторов (*шаблонов*) V_1, V_2, …, V_n длин k_1, k_2, …, k_n соответственно. Необходимо найти вхождения всех шаблонов в последовательность.

Поиск заданных слов в потоке данных применяется в различных анализаторах трафика, для нахождения синхропосылки в шифртексте и в системах беспроводной цифровой связи.

1. Известные подходы к задаче

Рассмотрим стандартные подходы к этой задаче. Оценивать их эффективность будем в предположении, что значения битов в потоке равновероятны и независимы друг от друга.

Можно «прикладывать» шаблоны, начиная с каждого бита потока, и побитно сравнивать их с содержимым потока. Оценим количество сравнений для одного шаблона длиной k битов. Первый бит шаблона нужно сравнивать с битом потока каждый раз, второй — только если первый бит совпал, то есть в среднем в половине случаев, и так далее. Всего потребуется $1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{k-1}} = 2 - \frac{1}{2^{k-1}} \approx 2$ сравнения на каждый бит потока. Если всего шаблонов n, то на каждый бит потока в среднем потребуется $2n$ сравнений.

Другой подход основан на *регистре сдвига*, в который последовательно подаются биты из потока. После подачи каждого бита содержимое регистра сравнивается со всеми шаблонами. Тут потребуется ровно n сравнений, но уже не битовых, а целочисленных, если содержимое регистра помещается в элементарный тип данных. С точки зрения реализации тут есть преимущество: сравнивать значение регистра с шаблоном проще и быстрее, чем значения отдельных битов. С другой стороны, если длина регистра отлична от длины шаблона, нужно обнулять старшие биты регистра перед сравнением, что занимает какое-то время, особенно если искомые шаблоны имеют различные длины. Если же регистр не помещается в элементарный тип данных, эта схема затрудняется ещё сильней.

2. Поиск шаблона с помощью конечного автомата

В данной работе предлагается строить конечный автомат [1], на вход которого подаются биты или байты потока, а на выходе получается информация о найденных в потоке шаблонах. На каждый бит или байт потока автомат совершает две простейших операции: изменение состояния по таблице переходов и определение выхода по таблице выходов. При этом автомат осуществляет одновременный поиск произвольного количества шаблонов любой длины, в том числе различной.

К недостаткам такого подхода относится необходимость хранить таблицы переходов и выходов автомата, которые могут оказаться довольно объёмными. Кроме того, при использовании автомата сложно реализовать «мягкий» поиск, допускающий некоторое количество ошибок в потоке.

2.1. Поисковый автомат с битовым входом

Сначала определим *автомат с битовым входом*, или *битовый автомат*.

Входной алфавит автомата — $\{0, 1\}$.

Выходной алфавит должен позволять выдавать информацию о позициях и номерах найденных шаблонов, однако для битового автомата достаточно указать лишь номера найденных шаблонов, поскольку за один такт работы автомата каждый шаблон может быть найден лишь один раз и может заканчиваться только в позиции последнего обработанного бита. Следовательно, выходной алфавит можно задать, например, множеством булевых векторов $\{0, 1\}^n$, где каждому биту сопоставлен взаимно однозначно некоторый шаблон из заданного набора, и бит принимает значение 1, если в текущей позиции закончился этот шаблон.

Определим множество состояний. Если автомат ищет один шаблон, то всё, что ему нужно знать, — это информацию о позициях, в которых этот шаблон мог начаться. Если длина шаблона k битов, а на вход автомата подали i-й бит, где $i \geqslant k$, то нас уже не интересует, мог ли шаблон начаться в $(i - k)$-м бите или ранее, так как если шаблон начался в $(i - k)$-м бите, то он закончился в $(i - 1)$-м, и автомат уже выдал информацию о его нахождении. Таким образом, нужно знать, могло ли вхождение шаблона начаться в позициях с номерами $i - k + 1, \ldots, i - 1$.

Вместо запоминания позиций возможного начала шаблона в потоке можно хранить длины уже найденных префиксов шаблона. Соответствие между длинами префиксов и позициями такое: шаблон мог начаться в позиции $(i - l)$, если и только если найден его префикс длины l после обработки $(i - 1)$-го бита. Нас интересуют только длины префиксов от 1 до $(k-1)$. Состояние такого автомата можно задать булевым вектором длины $(k - 1)$. Состояние автомата для поиска нескольких шаблонов можно задать набором аналогичных векторов.

Однако такой способ задания состояний избыточен, поскольку некоторые значения векторов бессмысленны. Например, для шаблона 101 невозможно найти одновременно префиксы длины 1 и 2 бита, следовательно, векторы, имеющие единицы в соответствующих позициях, задают невозможные состояния. Возможные состояния автомата можно закодировать целыми числами.

2.2. Построение битового автомата

Опишем алгоритм построения битового автомата.

Строится автомат по индукции. Сначала вводится нулевое состояние, соответствующее ситуации, когда не найден ни один префикс ни одного шаблона. Это состояние естественно для начала входной последовательности. Затем алгоритм поочерёдно обрабатывает все состояния условной подачей на вход автомата 0 и 1, попутно сохраняя новые полученные состояния и строя таблицы переходов и выходов автомата. Обработав таким образом все состояния автомата, получим таблицы переходов и выходов, а также таблицу соответствия номеров состояний их описаниям: состоянию s сопоставляется набор $T_s = (T_{s,1}, T_{s,2}, \ldots, T_{s,n})$, где $T_{s,j}$ содержит множество длин найденных префиксов шаблона номер j.

Обозначим l-й бит вектора V через $V[l]$, биты нумеруем с нуля.

Выходной алфавит автомата — множество булевых векторов длины n. В выходном векторе F бит $F[j-1]$ соответствует j-му шаблону.

Алгоритм построения битового поискового автомата

Вход: набор булевых векторов (шаблонов) V_1, V_2, …, V_n, их длины k_1, k_2, …, k_n
Выход: функции переходов ψ и выходов φ поискового автомата

1. Запоминаем начальное состояние $T_0 := (\varnothing, \varnothing, \ldots \varnothing)$, $s := 0$.
2. Обрабатываем состояние s условной подачей нуля и единицы. Подаваемый бит обозначим b. Сначала зададим $b := 0$.
3. Строим новое состояние T, в которое автомат должен перейти после подачи бита b в состоянии T_s, и соответствующий выходной символ — вектор F; сначала полагаем $F := 00\ldots0 = 0^n$.
   Для каждого j от 1 до n:
   3.1. Зададим $A := T_{s,j} \cup \{0\}$, $B_j := \varnothing$.
   3.2. **Для всех** $l \in A$ **если** $b = V_j[l]$, **то** $B_j := B_j \cup \{l+1\}$.
   Теперь множество B_j содержит длины всех найденных префиксов вектора V_j после подачи бита b в состоянии T_s.
   3.3. **Если** $k_j \in B_j$,**то** вектор V_j найден; полагаем $B_j := B_j \backslash \{l+1\}$, $F[j-1] := 1$.
4. Получили набор $T := (B_1, B_2, \ldots, B_n)$, описывающий следующее состояние, и выходной символ автомата — вектор F. Ищем набор T среди имеющихся наборов $T_0, \ldots, T_s$.
   Если нашли, что $T = T_h$, **то** присваиваем $s' := h$;
   если такого состояния ещё нет, **то** добавляем его в таблицу в новую ячейку. Пусть номер этой ячейки s'. Тогда состояние $T_{s'} = T$.
5. Запоминаем $\psi(s, b) := s'$, $\varphi(s, b) := F$.
6. **Если** $b = 0$, **то** $b := 1$ и переход на шаг 3.
7. $s := s + 1$. **Если** в таблице есть состояние T_s, **то** переход на шаг 2.
8. Ответ: функции ψ и φ.

2.3. Пример поискового битового автомата

Приведём пример битового автомата для поиска шаблонов 0111 и 1101 (таблица). Построенный автомат имеет семь состояний, в двух из которых возможен ненулевой выходной символ. Два состояния автомата (0 и 2) являются несущественными, то есть в них автомат может находиться только в начале обработки последовательности.

Таблица переходов и выходов автомата

Состояния		Функция ψ		Функция φ	
		0	1	0	1
$T_0 = (\varnothing, \varnothing)$	0	1	2	00	00
$T_1 = (\{1\}, \varnothing)$	1	1	3	00	00
$T_2 = (\varnothing, \{1\})$	2	1	4	00	00
$T_3 = (\{2\}, \{1\})$	3	1	5	00	00
$T_4 = (\varnothing, \{1, 2\})$	4	6	4	00	00
$T_5 = (\{3\}, \{1, 2\})$	5	6	4	00	10
$T_6 = (\{1\}, \{3\})$	6	1	3	00	01

2.4. Построение байтового автомата

Имея автомат, принимающий на вход биты, можно построить автомат, принимающий на вход сразу пачки битов, например байты или полубайты. В любом случае будем называть такой автомат *байтовым*, а его входные векторы — байтами. Множество состояний у него будет таким же, входной алфавит $\{0, 1\}^q$, где q — число битов в байте; выходной алфавит будет чуть сложнее, поскольку в байтовом автомате возможно нахождение сразу нескольких вхождений одного шаблона в различных позициях. От выходного алфавита требуется способность передавать списки найденных шаблонов вместе с позициями. В программной реализации автор использовал целочисленный выходной алфавит и дополнительную таблицу соответствия номера выхода спискам найденных шаблонов.

При наличии битового автомата таблицы переходов и выходов байтового автомата строятся просто: по очереди обрабатываются все состояния автомата и все возможные значения байта. Для каждой пары входного байта и состояния анализ происходит так: битовый автомат устанавливается в соответствующее состояние и на его вход побитно подаётся входной байт. Все выходные сигналы автомата собираются вместе и запоминаются с учётом того, при подаче какого бита был получен выходной сигнал. Новое состояние байтового автомата должно соответствовать состоянию, в которое перешёл битовый автомат после подачи всех битов байта.

3. Затраты по памяти

Одним из ограничений применения поисковых автоматов является необходимость хранения таблиц переходов и выходов, которые могут занимать значительные объёмы памяти, поэтому их стоит оценить.

3.1. Оценка количества состояний поискового автомата

Состояние автомата соответствует набору длин найденных префиксов искомых шаблонов. Пусть в некотором состоянии s самый длинный найденный префикс имеет длину l, и это префикс шаблона V_j. Тогда последние l битов, обработанных автоматом, совпадают с префиксом шаблона V_j и, следовательно, присутствие префиксов длины не больше l определено однозначно. Следовательно, любое состояние полностью определяется парой (j, l), соответствующей самому длинному найденному префиксу.

Если имеем всего n векторов с длинами k_1, k_2, …, k_n, то возможно всего $(k_1 - 1)$ пар вида $(1, l)$, $(k_2 - 1)$ пар вида $(2, l)$ и т. д. Помимо описанных, есть ещё нулевое состояние. Получается, что всего возможно не больше $\sum_{j=1}^{n} k_j - n + 1$ состояний.

Автомат, ищущий один шаблон длины k, всегда имеет ровно k состояний, то есть оценка достижима. Отметим, что в примере из п. 2.3 эта оценка также достигается.

3.2. Оценка объёма таблиц переходов и выходов

В программной реализации таблицы переходов и выходов объединены в одну таблицу, в ячейках которой вместе содержатся номер следующего состояния и индекс выхода. Сами выходы организованы в виде списков пар (номер шаблона, отступ позиции), описывающих найденные автоматом шаблоны, и хранятся в линейном массиве, называемом *массивом выходных сигналов*. Объём этого массива сильно зависит от вида шаблонов.

Минимальный объём массива выходных сигналов байтового автомата можно оценить числом nq, поскольку в любом бите байта может начаться любой шаблон. Однако среди выходов автомата могут встречаться не только единичные найденные шаблоны, но и списки нескольких найденных шаблонов, что может существенно увеличить объём массива. Практика показывает, что при достаточно большой длине шаблонов (как минимум, превышающей длину байта) этого обычно не происходит.

Для байтового автомата, полученного из битового автомата в примере из п. 2.3, объём массива выходных сигналов оказался очень значительным, что объясняется возможностью нахождения большого количества различных комбинаций искомых шаблонов после обработки очередного байта. Однако использование байтового автомата для поиска столь коротких шаблонов в любом случае не представляется разумным.

Оценим объём таблицы переходов и выходов. Число строк этой таблицы равно количеству состояний автомата, число столбцов — мощности входного алфавита. Для битового автомата оно равно 2, для байтового — 2^q.

В системах с восьмибитными байтами получаем 256 столбцов. Если состояние автомата и индекс выхода помещаются в машинное слово, то объём одной ячейки объединённой таблицы переходов и выходов занимает 4 байта, строка байтового автомата — 1024 байта. При большом количестве состояний объём таблиц может оказаться слишком большим для внутреннего высокоскоростного кэша процессора, особенно если речь идёт о специализированных процессорах сетевого оборудования или портативных устройств цифровой передачи данных. В этом случае стоит использовать автомат, получающий на вход половинки байтов, и обрабатывать каждый байт за два этапа. Это снизит скорость обработки приблизительно в 2 раза и немного усложнит реализацию, но уменьшит объём таблицы в 16 раз. Кроме того, уменьшится массив выходных сигналов.

4. Возможные обобщения

Описанную методику несложно обобщить на случай поиска неполностью определённых булевых векторов, заданных троичными векторами. Однако в этом случае оценка количества состояний будет отличаться, и, вероятно, в большую сторону. Другое возможное обобщение — это использование отличного от $\{0, 1\}$ алфавита для последовательности и искомых шаблонов.

5. Связь с известными алгоритмами

Описанный подход имеет много общего с алгоритмом поиска подстроки в строке Кнута — Морриса — Пратта (КМП) [2]. Оба метода предполагают предварительные вычисления, а затем требуют одну простую операцию на каждый символ входной последовательности.

Фактически битовый автомат для поиска одного шаблона реализует КМП-алгоритм: каждому состоянию поискового автомата можно сопоставить число — максимальную длину найденного префикса шаблона. Это число совпадёт с длиной найденного префикса шаблона в алгоритме КМП; назовём его номером состояния. При совпадении очередного бита шаблона с битом потока в алгоритме КМП длина найденного префикса увеличивается на единицу, как и номер состояния поискового автомата. При несовпадении автомат откатится назад в состояние с номером, соответствующим значению префикс-функции алгоритма КМП.

Однако есть и существенные отличия:

1) Поисковый автомат способен искать сразу несколько шаблонов. Для поиска n шаблонов можно использовать n автоматов, ищущих по одному шаблону, однако, как видно из оценки количества состояний, суммарный объём таблиц этих автоматов неизбежно будет превышать объём автомата, ищущего сразу все n шаблонов. Кроме того, набору из n автоматов на каждый входной символ (бит, полубайт или байт) необходимо производить в n раз больше действий, чем одному автомату.
2) Поисковые автоматы способны принимать символы последовательности не только по одному, но и пачками.

Существует также сходство описываемых в данной работе поисковых автоматов с алгоритмом поиска подстроки в строке Ахо — Корасик [3], который тоже позволяет искать сразу набор шаблонов. Алфавит в этом алгоритме не обязательно двоичный, может быть произвольным. В алгоритме Ахо — Корасик используется конечный автомат, состояния которого образуют дерево. Узлы дерева соответствуют префиксам искомых шаблонов, как и в поисковых автоматах. Функционирование автомата Ахо — Корасик происходит аналогично функционированию описываемых в данной работе автоматов.

В [3] при подаче на вход автомата символа, который не может продолжить префикс, соответствующий текущему узлу, происходит дополнительный переход по так называемой *суффиксной ссылке*, что немного замедляет работу автомата. Вместо таблиц переходов и выходов хранится дерево состояний автомата. Количества состояний автомата Ахо — Корасик и поискового автомата, очевидно, совпадают. Для каждого состояния необходимо хранить всю информацию о соответствующем узле дерева состояний, которая включает набор ссылок на потомков и суффиксную ссылку. В случае двоичного алфавита суммарно будет не более двух ссылок. Дополнительно требуется хранить так называемую *словарную суффиксную ссылку* и признак, является ли узел *словарным* и какому шаблону он соответствует.

Однако вместо этого можно закодировать узлы дерева — они же состояния автомата — и составить таблицы переходов и выходов, получив таким образом тот же самый поисковый автомат, который предлагается в данной работе. Требуемый объём памяти и вычислительная сложность будут совпадать с поисковым автоматом.

В данной работе предлагается прямой алгоритм построения таблиц переходов и выходов без использования дерева состояний, а также автомат, принимающий на вход

сразу пачки битов или байты и позволяющий искать неполностью определённые булевы векторы.

Заключение

Предлагаемый поисковый автомат позволяет с очень высокой эффективностью одновременно искать в потоке данных несколько булевых векторов (шаблонов) некоторых длин. При этом с увеличением количества шаблонов объём требуемой памяти растёт медленнее, чем их суммарная длина, а трудоёмкость не изменяется совсем.

Для увеличения скорости обработки потока данных можно подавать биты потока на вход автомата пачками или байтами. При подаче битов пачками по q битов скорость обработки возрастёт в q раз, а объём требуемой памяти — в 2^q раз.

Есть возможность обобщить алгоритм построения поискового автомата на поиск неполностью определённых булевых векторов, однако в этом случае объём требуемой памяти может превышать найденную оценку.

ЛИТЕРАТУРА

1. *Агибалов Г. П., Оранов А. М.* Лекции по теории конечных автоматов. Томск: Изд-во Том. ун-та, 1984. 185 с.
2. *Knuth D. E., Morris J. H. Jr., and Pratt V. R.* Fast pattern matching in strings // SIAM J. Comput. 1977. No. 6(2). P. 323–350.
3. *Aho A. V. and Corasick M. J.* Efficient string matching: An aid to bibliographic search // Commun. ACM. 1975. No. 18(6). P. 333–340.

СВЕДЕНИЯ ОБ АВТОРАХ

АВЕЗОВА Яна Эдуардовна — студентка Национального исследовательского ядерного университета «МИФИ», г. Москва. E-mail: **avezovayana@gmail.com**

ЗАКАБЛУКОВ Дмитрий Владимирович — аспирант кафедры информационной безопасности Московского государственного технического университета им. Н. Э. Баумана, г. Москва. E-mail: **dmitriy.zakablukov@gmail.com**

ЛОСЕВ Александр Сергеевич — кандидат физико-математических наук, младший научный сотрудник Института прикладной математики ДВО РАН, старший преподаватель Дальневосточного федерального университета, г. Владивосток.
E-mail: **alexax@bk.ru**

НОРМОВ Андрей Иванович — аспирант Российского экономического университета им. Г. В. Плеханова, г. Москва. E-mail: **normch88@mail.ru**

ОСИПОВА Марина Анатольевна — кандидат физико-математических наук, доцент, научный сотрудник Института прикладной математики ДВО РАН, доцент Дальневосточного федерального университета, г. Владивосток. E-mail: **mao1975@list.ru**

ПАНКРАТОВ Иван Владимирович — г. Томск.
E-mail: **ivan.pankratov2010@yandex.ru**

ПОДОЛЬКО Дмитрий Константинович — аспирант Московского государственного университета им. М. В. Ломоносова, г. Москва. E-mail: **podolko_dk@mail.ru**

САДЫКОВ Тимур Мрадович — доктор физико-математических наук, профессор Российского экономического университета им. Г. В. Плеханова, г. Москва.
E-mail: **SadykovTM@rsute.ru**

СЕРЕБРЯКОВ Евгений Михайлович — Центр специальных разработок, г. Москва.
E-mail: **ryback_casey@mail.ru**

СМОЛЬЯНИНОВ Владимир Юрьевич — научный сотрудник Учебно-методического объединения по информационной безопасности, г. Москва.
E-mail: **VladimirSmall@gmail.com**

ФОМИЧЕВ Владимир Михайлович — доктор физико-математических наук, профессор, профессор Финансового университета при Правительстве Российской Федерации, профессор Национального исследовательского ядерного университета «МИФИ», г. Москва. E-mail: **fomichev@nm.ru**

ЦИЦИАШВИЛИ Гурами Шалвович — доктор физико-математических наук, профессор, заведующий лабораторией Института прикладной математики ДВО РАН, профессор Дальневосточного федерального университета, г. Владивосток.
E-mail: **guram@iam.dvo.ru**

ЧЕРЕМИСИНОВ Дмитрий Иванович — кандидат технических наук, доцент, ведущий научный сотрудник Объединенного института проблем информатики НАН Беларуси, г. Минск. E-mail: **cher@newman.bas-net.by**

ЧЕРЕМУШКИН Александр Васильевич — доктор физико-математических наук, член-корреспондент Академии криптографии РФ, заведующий кафедрой Института криптографии, связи и информатики, г. Москва. E-mail: **avc238@mail.ru**

АННОТАЦИИ СТАТЕЙ НА АНГЛИЙСКОМ ЯЗЫКЕ

Avezova Y. E., Fomichev V. M. **COMBINATORIAL PROPERTIES OF RECTANGULAR 0,1-MATRIX SYSTEMS.** The combinatorial properties of a multiplicative partial semi-group generated by a system of non-negative rectangular matrices are investigated. The concept of primitiveness is extended from systems of square non-negative matrices to the systems of rectangular matrices. Some estimations for exponent of non-negative rectangular matrices are given.

Keywords: *system of rectangular matrices, partial semi-group, primitive system of matrices, exponent.*

Podolko D. K. **ON ONE CONTINUAL SET OF β-CLOSED CLASSES OF THE MULTIVALUED LOGIC FUNCTIONS.** The paper studies β-closed classes of the multivalued logic functions, where the β-closure operator is defined on the basis of the functions encoding in the binary number system. A continual set of β-closed classes, which contain only functions taking no more than three values, is given and some of its properties are proved.

Keywords: *multivalued logic functions, superposition, closed classes, β-closure.*

Serebryakov E. M. **RECOVERY OF A POLYNOMIALLY COMPLICATED LINEAR RECURRING SEQUENCE OVER GALOIS RING BY ITS SENIOR COORDINATE.** Maximal peroid linear recurring sequences (LRS) over a Galois ring, which are complicated with a polynomial over this ring, are considered. An algorithm is proposed for recovering the initial vector of a LRS by the senior coordinate of its complicated sequence.

Keywords: *LRS of maximal period, complicated polynomial, senior coordinate sequence, recovery of initial vector.*

Cheremushkin A. V. **COMPUTATION OF NONLINEARITY DEGREE FOR DISCRETE FUNCTIONS ON PRIMARY CYCLIC GROUPS.** A method is proposed for computing the nonlinearity degree of a discrete functions defined on a cyclic group of order p^n. The method is based on Newton expansion for a discrete function. Theorem 1 presents the values of nonlinearity degree for all basic functions in Newton expansion. Theorems 2 and 3 illustrate number distributions for functions on cyclic groups of order p^2 and p^3 according to their nonlinearity degrees.

Keywords: *discrete functions, nonlinearity degree, Newton expansion.*

Smolyaninov V. Y. **ANALYSIS OF THE CONDITIONS FOR GRANTING AND OBTAINING ACCESS RIGHTS IN THE MS SQL SERVER ACCESS CONTROL MODEL.** In this paper, the MS SQL Server access control model, based on the DBMS DP-model, is introduced. For taking into account the access control features of Microsoft SQL Server, the model includes roles, permissions to user accounts and roles, ownership chaining, user impersonation and activating procedures and triggers on behalf of the specified user accounts. The statement of the equivalence of the possibilities to execute arbitrary SQL-code on behalf of a specified account and to obtain the right of its impersonation is proved. Some necessary and sufficient conditions for obtaining and granting access

rights by entities in the absence of cooperation between sessions are proved.
Keywords: *computer security, MS SQL Server access control model, database management system.*

Normov A. I., Sadykov T. M. **ANALYTIC COMPLEXITY OF CLUSTER TREES.** The notion of the analytic complexity of a binary tree is introduced. This is a nonnegative integer reflecting the combinatorial structure of a tree and its most concise analytic representation. The properties of the analytic complexity of a tree are described, and how to calculate it algorithmically is explained. The developed methods are used to compare cluster trees.
Keywords: *cluster analysis, binary tree, analytic complexity.*

Fomichev V. M. **ESTIMATES FOR EXPONENT OF SOME GRAPHS BY FROBENIUS'S NUMBERS OF THREE ARGUMENTS.** A formula for Frobenius's numbers of three arguments is given. Estimates for exponent of some superconnected digraphs are obtained using this formula. It is shown that the given estimation is the best in many cases.
Keywords: *Frobenius's number, additive semigroup generated by set of numbers, exponent of graph.*

Tsitsiashvili G. Sh., Osipova M. A., Losev A. S. **PROOF OF ASYMPTOTIC CONSTANTS IN DISCONNECTION PROBABILITY FOR WEIGHTED PLANAR GRAPH.** In this paper, formulas for the calculation of asymptotic constants in the disconnection probability for a weighted planar graph with high reliable edges are proved.
Keywords: *disconnection probability, weighted planar graphs.*

Zakablukov D. V. **FAST SYNTHESIS OF INVERTIBLE CIRCUITS BASED ON PERMUTATION GROUP THEORY.** Various algorithms for the synthesis of invertible logic circuits are considered and their main characteristics are presented. A new fast synthesis algorithm based on permutation group theory is proposed. This algorithm allows to synthesize schemes with the gate complexity $\mathrm{O}(n2^m)$ and with the time complexity $\mathrm{O}(n2^m)$ without using additional inputs. Here, n is the number of scheme's inputs and m is the upper bound for $\log k$ where k is the number of non-fixed points of the given invertible transformation.
Keywords: *invertible logic, synthesis algorithm, permutation groups.*

Cheremisinov D. I. **PROTECTING INTELLECTUAL PROPERTY IN FPGA XILINX DESIGN.** The problem of intellectual property protection in design process of FPGA Xilinx is considered. Soft- and hardware destined to control license agreements are discussed, because such an agreement for Xilinx FPGA CAD forbids the use of this tool for a reverse engineering. It is shown, that there are classes of FPGA devices whose configuration files are not decompilable. In particular, such a class consists of devices with partial dynamic reconfiguration by means of an internal reconfiguration controller.
Keywords: *reverse engineering, FPGA, intellectual property protection.*

Pankratov I. V. **SIMULTANEOUS SEARCH FOR SEVERAL BINARY PATTERNS IN A STREAM WITH FINITE-STATE AUTOMATON.** The task under consideration in this paper is to search for binary subsequencies in a data stream. The article offers the finite automaton able to search for a set of binary vectors simultaneously performing only two operations per every bit or even byte of the data stream. Increasing the number of searched vectors causes a slower increase in memory usage compared to

overall vector's length growth, and computational complexity does not increase at all. The automaton is described by the transition and output tables. Estimates of the size of the automaton's tables are given. Known approaches to the problem are discussed. There is a possibility to generalize the automaton building algorithm to search for partially defined Boolean patterns, but the amount of required memory may be greater than the estimate found in the paper.

Keywords: *bit subsequences search, synchronization, string matching, Knuth — Morris — Pratt algorithm, Aho — Corasick string matching.*

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 journals.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 надёжности 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 её приложениям*