ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2010 №1(7)

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.

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

Изд. лиц. ИД. №04000 от 12.02.2001. Подписано к печати 15.03.2010.
Формат $60 \times 84\frac{1}{8}$. Бумага офсетная. Печать офсетная. Гарнитура «Таймс».
Усл. п. л. 13,4. Уч.-изд. л. 15. Тираж 300 экз. Заказ №14.

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

СОДЕРЖАНИЕ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

DOI 10.17223/20710410/7/1

УДК 519.7

О КРИПТОГРАФИЧЕСКИХ СЛАБОСТЯХ НЕКОТОРЫХ КЛАССОВ ПРЕОБРАЗОВАНИЙ ДВОИЧНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ[1]

С. В. Смышляев

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

E-mail: smyshsv@gmail.com

В работе рассматривается ряд вопросов, связанных с использованием совершенно уравновешенных булевых функций в качестве фильтрующих и появлением слабостей в соответствующих криптопримитивах.

Ключевые слова: *совершенно уравновешенные функции, барьеры булевых функций, фильтрующий генератор, криптография.*

Введение

При изучении криптографических примитивов, состоящих из регистра сдвига и подключенной к нему реализующей некоторую булеву функцию схемы (фильтра), одним из важнейших вопросов является принципиальная возможность получить на выходе фильтра произвольную двоичную последовательность. Соответствующее свойство реализуемой булевой функции (фильтрующей) было формализовано С. Н. Сумароковым в понятии совершенной уравновешенности, им же в [1] были получены первые утверждения о классе совершенно уравновешенных функций. Исследование данного класса было продолжено в работе [2], в которой авторами было введено понятие барьера булевой функции, важное для изучения аспектов, связанных с использованием булевых функций в качестве фильтрующих, а также был доказан ряд утверждений о совершенно уравновешенных функциях с барьером. Позже [3, 4], в числе прочего, были доказаны некоторые утверждения о слабостях фильтров, построенных с помощью функций с барьером конечной длины, и построены некоторые классы совершенно уравновешенных функций без барьера. Работы [5, 6] были целиком посвящены методам построения классов совершенно уравновешенных функций, удовлетворяющих определенным наборам требований; в частности, широких классов функций без барьера [6].

В настоящей работе рассматриваются два аспекта использования совершенно уравновешенных булевых функций в качестве фильтрующих, связанные с появлением слабостей в соответствующих криптопримитивах.

В п. 1 и 2 приводятся необходимые определения и предварительные результаты. В п. 3.1 доказывается новый критерий совершенной уравновешенности функции, обосновывающий негативные криптографические качества совершенно уравновешенных булевых функций при использовании соответствующих им фильтров в ситуациях, когда на вход поступают отличные от истинно случайных последовательности — в част-

[1]Работа поддержана РФФИ (проект № 09-01-00653-а).

ности, такие, внутри которых никогда не встречаются некоторые наборы подряд идущих символов.

П. 3.2 посвящен углубленному изучению одного из поднятых в работе [4] вопросов о криптографических слабостях фильтров, построенных с использованием функций с барьером небольшой длины. Рассмотрены вопросы формального построения правил восстановления символов входной последовательности с использованием некоторых подмножеств символов выходной последовательности, доказан ряд особых свойств таких правил в терминах частично определенных булевых функций и в терминах барьеров совершенно уравновешенных булевых функций.

1. Основные определения и обозначения

Для множества двоичных наборов длины n будем использовать обозначение $V_n = \{0,1\}^n$. Через $\mathcal{F}_n$ будем обозначать множество булевых функций от n переменных.

Пусть $n, m \in \mathbb{N}$, $f \in \mathcal{F}_n$. Рассмотрим систему булевых уравнений:

$$f(x_s, x_{s+1}, \ldots, x_{s+n-1}) = y_s,\ s = 1, 2, \ldots, m. \tag{1}$$

Обозначим для $f \in \mathcal{F}_n$ через f_m отображение из V_{m+n-1} в V_m вида

$$f_m(x_1, x_2, \ldots, x_{m+n-1}) = (f(x_1, \ldots, x_n), f(x_2, \ldots, x_{n+1}), \ldots, f(x_m, \ldots, x_{m+n-1})). \tag{2}$$

Нетрудно заметить, что отображение f_m можно понимать как порождаемое m тактами работы фильтра — кодирующего устройства, полученного с помощью подключения входов булевой функции f (называемой, в таком контексте, фильтрующей функцией) к некоторым ячейкам двоичного регистра сдвига. Фильтры часто применяются в криптографических приложениях, связанных с потоковыми шифрами, — например, простейшим примитивом, с помощью которого можно выработать складываемую с битами открытого текста псевдослучайную последовательность, является фильтр, на вход которого подается некоторая линейно-рекуррентная последовательность.

Определение 1 [1]. Булева функция $f \in \mathcal{F}_n$ называется *функцией без запрета* (*функцией дефекта нуль*), если соотношение

$$(f_m)^{-1}(y) \neq \varnothing$$

выполняется для любого $m \in \mathbb{N}$ и любого $y \in V_m$.

Определение 2 [1]. Булева функция $f \in \mathcal{F}_n$ называется *совершенно уравновешенной*, если соотношение

$$\sharp(f_m)^{-1}(y) = 2^{n-1}$$

выполняется для любого $m \in \mathbb{N}$ и любого $y \in V_m$. Множество совершенно уравновешенных функций из $\mathcal{F}_n$ обозначим через $\mathcal{PB}_n$.

Введем понятие барьера булевой функции, тесно связанное с понятием совершенной уравновешенности.

Определение 3 [2]. Булева функция $f \in \mathcal{F}_n$ называется *функцией с правым барьером длины* b, если система уравнений

$$\begin{cases} f(y_1, y_2, \ldots, y_n) = f(z_1, z_2, \ldots, z_n); \\ f(y_2, y_3, \ldots, y_{n+1}) = f(z_2, z_3, \ldots, z_{n+1}); \\ \ldots \\ f(y_{b-1}, y_b, \ldots, y_{b+n-2}) = f(z_{b-1}, z_b, \ldots, z_{b+n-2}); \\ y_1 = z_1; \ldots; y_{n-1} = z_{n-1}; y_n = 0; z_n = 1 \end{cases} \tag{3}$$

имеет решение, а система

$$\begin{cases} f(y_1, y_2, \ldots, y_n) = f(z_1, z_2, \ldots, z_n); \\ f(y_2, y_3, \ldots, y_{n+1}) = f(z_2, z_3, \ldots, z_{n+1}); \\ \ldots \\ f(y_{b-1}, y_b, \ldots, y_{b+n-2}) = f(z_{b-1}, z_b, \ldots, z_{b+n-2}); \\ f(y_b, y_{b+1}, \ldots, y_{b+n-1}) = f(z_b, z_{b+1}, \ldots, z_{b+n-1}); \\ y_1 = z_1; \ldots; y_{n-1} = z_{n-1}; y_n = 0; z_n = 1 \end{cases} \tag{4}$$

решений не имеет.

Булева функция $f \in \mathcal{F}_n$ называется функцией с левым барьером длины b, если $f'(x_1, \ldots, x_n) \equiv f(x_n, \ldots, x_1)$ является функцией с правым барьером длины b.

Булева функция $f \in \mathcal{F}_n$ имеет барьер, если она имеет правый или левый барьер, или оба сразу. При этом длиной барьера функции называется соответственно длина правого барьера, левого барьера или меньшая из длин барьеров.

Замечание 1. Нетрудно заметить, что наличие правого (левого) барьера длины 1 означает линейность функции по последнему (первому) аргументу.

Отметим, что для всех утверждений, в которых упоминается длина правого барьера некоторых функций, могут быть очевидным образом построены аналоги с использованием понятия левого барьера. Ввиду этого далее будем говорить только о правых барьерах функций.

2. Предварительные результаты

Отметим важное для криптографических приложений свойство совершенно уравновешенных функций.

Теорема 1 [7]. Пусть $n \in \mathbb{N}$, $f \in \mathcal{F}_n$. Для последовательности случайных векторов $X_m = (x_1, x_2, \ldots, x_{m+n-1})$, $m = 1, 2, \ldots$, с распределением

$$\Pr\{X_m = (a_1, \ldots, a_{m+n-1})\} = 2^{-(m+n-1)} \text{ для каждого } (a_1, \ldots, a_{m+n-1}) \in V_{m+n-1}$$

случайный вектор $Y_m = f_m(X_m)$ распределен равномерно для любого $m \in \mathbb{N}$ тогда и только тогда, когда f — совершенно уравновешенная функция.

Другими словами, если биты двоичной последовательности, поступающей на вход фильтра, распределены равномерно и независимо, то биты выходной двоичной последовательности обладают этим свойством тогда и только тогда, когда фильтрующая функция совершенно уравновешена.

Для исследования класса совершенно уравновешенных функций часто удобно использовать следующую теорему.

Теорема 2 [1, 7]. Пусть $n \in \mathbb{N}$ и $f \in \mathcal{F}_n$. Тогда следующие утверждения эквивалентны:

— f является совершенно уравновешенной;
— f является функцией без запрета;
— не существует двух различных двоичных последовательностей

$$x = (x_1, x_2, \ldots, x_r),\ z = (z_1, z_2, \ldots, z_r) \in V_r, r \geqslant 2n - 1,$$

таких, что

$$x_1 = z_1,\ x_2 = z_2,\ \ldots,\ x_{n-1} = z_{n-1};\ x_{r-n+2} = z_{r-n+2},\ x_{r-n+3} = z_{r-n+3},\ \ldots,\ x_r = z_r;$$
$$f_{r-n+1}(x) = f_{r-n+1}(z).$$

Связь совершенной уравновешенности с наличием у функции барьера описывается следующим утверждением.

Теорема 3 [2]. Наличие барьера у булевой функции является достаточным, но не необходимым условием совершенной уравновешенности функции.

3. Основные результаты

3.1. Критерий совершенной уравновешенности

Следующее утверждение представляет собой критерий совершенной уравновешенности, суть которого заключается в существовании для каждого фиксированного набора длины k такой выходной последовательности, которая не может быть получена с помощью фильтрующей функции f ни на какой входной последовательности, не содержащей внутри себя этого набора. Иными словами, из запрета на присутствие любого фиксированного набора символов во входной последовательности кодирующего устройства указанного типа в случае совершенно уравновешенной фильтрующей функции незамедлительно следует уменьшение мощности множества возможных выходных последовательностей.

Теорема 4. Функция является совершенно уравновешенной, если и только если для любых $k > 0$, $(y_1, \ldots, y_k) \in V_k$ существуют такие $r \geqslant \max\{1, 1 - n + k\}$ и $z = (z_1, \ldots, z_r) \in V_r$, что выполняются следующие условия:

1) $\exists (x_1, \ldots, x_{r+n-1}) \in V_{r+n-1} \, (f_r(x_1, \ldots, x_{r+n-1}) = (z_1, \ldots, z_r))$;
2) $\forall (x_1, \ldots, x_{r+n-1}) \in V_{r+n-1} (f_r(x_1, \ldots, x_{r+n-1}) = (z_1, \ldots, z_r) \Rightarrow$
$\Rightarrow \exists i \, (1 \leqslant i \leqslant r + n - k \;\&\; (x_i, \ldots, x_{i+k-1}) = (y_1, \ldots, y_k)))$.

Доказательство.

Необходимость. Для произвольной $f \in \mathcal{PB}_n$ предположим противное: существуют $k > 0$ и $(y_1, \ldots, y_k) \in V_k$, такие, что для любого $r \geqslant \max\{1, 1 - n + k\}$ и любого $z = (z_1, \ldots, z_r) \in V_r$ выполнено хотя бы одно из двух условий:

а) $\nexists (x_1, \ldots, x_{r+n-1}) \in V_{r+n-1} \, (f_r(x_1, \ldots, x_{r+n-1}) = (z_1, \ldots, z_r))$;
б) $\exists (x_1, \ldots, x_{r+n-1}) \in V_{r+n-1} (f_r(x_1, \ldots, x_{r+n-1}) = (z_1, \ldots, z_r) \;\&$
$\&\;\; \forall i \, (1 \leqslant i \leqslant r + n - k \Rightarrow (x_i, \ldots, x_{i+k-1}) \neq (y_1, \ldots, y_k)))$.

Покажем, что условие «б» не может выполняться для всех r и z. Представим любой набор $x \in V_{r+n-1}$ в виде

$$x = ((x_1^1, \ldots, x_k^1), (x_1^2, \ldots, x_k^2), \ldots, (x_1^p, \ldots, x_k^p), x_1^{p+1}, \ldots, x_m^{p+1}) \in V_{r+n-1},$$

где $0 \leqslant m < k$, $p = \left[\dfrac{r+n-1}{k}\right]$. Всего таких x, что $(x_1^j, \ldots, x_k^j) \neq (y_1, \ldots, y_k)$ для всех $j = 1, \ldots, p$ (что, очевидно, необходимо для выполнения соотношения

$$\forall i \, (1 \leqslant i \leqslant r + n - k \Rightarrow (x_i, \ldots, x_{i+k-1}) \neq (y_1, \ldots, y_k))$$

из условия «б»), для каждого r существует не более чем $(2^k - 1)^p \cdot 2^{k-1}$.

Верно соотношение $\dfrac{(2^k-1)^p 2^{k-1}}{2^{r+n-1}} \leqslant \left(\dfrac{2^k-1}{2^k}\right)^p 2^{k-1} \underset{r \to +\infty}{\longrightarrow} 0$. Поэтому существует столь большое r', что число последовательностей $x \in V_{r'+n-1}$, не содержащих $(y_1, \ldots, y_k)$, не превосходит $2^{r'+n-1} 2^{-n} = 2^{r'-1}$. Всего возможных последовательностей $z \in V_{r'}$ ровно $2^{r'}$, поэтому хотя бы для одного $z \in V_{r'}$ будет выполнено условие a, противоречащее определению совершенно уравновешенной функции. Полученное противоречие завершает доказательство необходимости.

Достаточность. Предположим противное: $f \notin \mathcal{PB}_n$. Тогда, как следует из теоремы 2, $\exists k > 2n$ $\exists (y'_1, \ldots, y'_k), (y''_1, \ldots, y''_k) \in V_k$ ($y'_1 = y''_1$, …, $y'_n = y''_n$; $y'_{k-n+1} = y''_{k-n+1}$, …, $y'_k = y''_k$; $f_{k-n+1}(y'_1, y'_2, \ldots, y'_k) = f_{k-n+1}(y''_1, y''_2, \ldots, y''_k)$), причем $(y'_1, \ldots, y'_k) \neq (y''_1, \ldots, y''_k)$.

Положим $(y_1, \ldots, y_k) = (y'_1, \ldots, y'_k)$. Если существуют $r > 0$ и $z = (z_1, \ldots, z_r) \in V_r$, такие, что

$$\begin{cases} f(x_1, \ldots, x_n) = z_1; \\ \ldots \\ f(x_r, \ldots, x_{r+n-1}) = z_r \end{cases} \tag{5}$$

для какого-то $(x_1, \ldots, x_{r+n-1})$, где $(x_{i^*}, \ldots, x_{i^*+k-1}) = (y'_1, \ldots, y'_k)$ при некотором i^*, то, просматривая $(x_1, \ldots, x_{r+n-1})$ слева направо и заменяя все вхождения $(y'_1, \ldots, y'_k)$ на $(y''_1, \ldots, y''_k)$, получим решение системы (5), не содержащее внутри себя $(y_1, \ldots, y_k)$. Получаем противоречие с условием, т. е. f совершенно уравновешена. Достаточность доказана. ■

Замечание 2. Логично задаться вопросом: а для всякой ли выходной последовательности $z = (z_1, \ldots, z_r) \in V_r$ существует набор $y = (y_1, \ldots, y_k) \in V_k$, входящий (в указанном смысле) в любую входную последовательность $x = (x_1, \ldots, x_{r+n-1})$ из прообраза z, в случае совершенно уравновешенной функции? Ответ на этот вопрос, вообще говоря, отрицательный. Достаточно рассмотреть функцию $f(x_1, x_2) = x_1 \oplus x_2$ и состоящую из одних нулей выходную последовательность z — в этом случае в прообраз z будет входить как последовательность из единиц, так и последовательность из нулей. Очевидно, ни одного y, входящего в оба эти решения, не существует.

Свойство совершенно уравновешенных функций, доказанное в теореме 4, означает следующее. Важное свойство, благодаря которому при преобразовании двоичных последовательностей с помощью кодирующего устройства с совершенно уравновешенной фильтрующей функцией f в выходных последовательностях сохраняются хорошие свойства близких к истинно случайным входных последовательностей, никаким положительным образом не переносится на случай далеких от истинно случайных входных последовательностей. Если входная последовательность порождена некоторым детерминированным устройством с конечной памятью и имеет явные зависимости между некоторыми битами, то эти зависимости, а точнее, непосредственно следующие из них запреты на появление некоторых наборов внутри входной последовательности, кодирующим устройством с совершенно уравновешенной фильтрующей функцией всегда будут явно отражены в появлении запретов в выходных последовательностях. То есть совершенная уравновешенность фильтрующей функции делает теоретически возможным распознавание по выходной последовательности структурных особенностей входной последовательности, если она была порождена с помощью некоторого конечного автомата. Например, если на вход фильтрующего устройства с совершенно уравновешенной функцией подавать линейно-рекуррентную последовательность, то по выходу теоретически возможно восстановление рекурренты.

Утверждение теоремы 4 можно переформулировать следующим образом: фильтрующая функция сохраняет запреты (в соответствующем смысле) тогда и только тогда, когда она совершенно уравновешена. Нетрудно заметить, что из данного критерия совершенной уравновешенности непосредственно следует, например, утверждение о совершенной уравновешенности булевой функции, полученной из некоторых двух совершенно уравновешенных функций с помощью описанной в работе [5] специально определенной операции композиции фильтрующих функций.

Случай отсутствия у фильтрующей функции совершенной уравновешенности и соответствующая ему способность кодирующего устройства с этой функцией не сохранять некоторые запреты могут быть проиллюстрированы следующим простым примером.

Рассмотрим $f(x_1, x_2) = x_1 x_2 \in \mathcal{F}_2 \backslash \mathcal{PB}_2$ и соответствующее ей фильтрующее устройство. Предположим, что на вход данного устройства могут поступать любые последовательности, за исключением содержащих внутри себя набор $(0, 1, 0)$. Покажем, что появление такого запрета на входе не породит запрета на выходе. Для этого заметим, что равенство

$$f_r(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{i-1}, 0, 1, 0, \varepsilon_{i+3}, \ldots, \varepsilon_{r+1}) = f_r(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{i-1}, 0, 0, 0, \varepsilon_{i+3}, \ldots, \varepsilon_{r+1})$$

выполняется для любых $r \in \mathbb{N}$, $i = 1, \ldots, r-1$, и любых значений ε_j, то есть удаление в произвольной входной последовательности всех наборов $(0, 1, 0)$ и подстановка на их места наборов $(0, 0, 0)$ никак не скажется на выходной последовательности.

Из всего вышесказанного можно сделать заключение о том, что использование совершенно уравновешенных функций в качестве фильтрующих предпочтительно в тех и только в тех случаях, когда им на вход будут поступать близкие к истинно случайным двоичные последовательности, не обладающие явными зависимостями между битами. В противном случае лучше использовать функции, которые не переносят некоторые возникающие во входных последовательностях запреты в выходные последовательности, а следовательно, не позволяют по выходам кодирующего устройства узнавать дополнительную информацию о структуре входных последовательностей.

3.2. Восстановление символов входной последовательности

Рассмотрим вопрос, связанный с восстановлением символов входной последовательности фильтрующего устройства. Пусть булева функция $f \in \mathcal{F}_n$ имеет правый барьер длины $b < +\infty$. В случае $b = 1$ f представима в виде $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_{n-1}) \oplus x_n$, поэтому при использовании этой функции как фильтрующей в кодирующем устройстве упомянутого типа каждый входной бит последовательности однозначно определяется по предыдущим $n-1$ входным и соответствующему выходному биту: $x_i = \widehat{f}(x_{i-n+1}, \ldots, x_{i-1}, y_{i-n+1}) = g(x_{i-n+1}, \ldots, x_{i-1}) \oplus y_{i-n+1}, i = n, n+1, \ldots$ Возникает вопрос: можно ли для функций с правым барьером большей длины построить аналог локально обратной функции $\widehat{f}$?

Определим для произвольной функции с барьером длины $b < +\infty$ частично определенную булеву функцию $\widehat{f}(x_1, \ldots, x_{n-1}, y_1, \ldots, y_b)$ следующим образом:

$$\widehat{f}(x_1, \ldots, x_{n-1}, y_1, \ldots, y_b) = \\ = \begin{cases} 0, \ \exists (t_2, \ldots, t_b) \in V_{b-1} : f_b(x_1, \ldots, x_{n-1}, 0, t_2, \ldots, t_b) = (y_1, \ldots, y_b); \\ 1, \ \exists (t_2, \ldots, t_b) \in V_{b-1} : f_b(x_1, \ldots, x_{n-1}, 1, t_2, \ldots, t_b) = (y_1, \ldots, y_b); \\ *, \ \nexists (t_1, \ldots, t_b) \in V_b : f_b(x_1, \ldots, x_{n-1}, t_1, \ldots, t_b) = (y_1, \ldots, y_b). \end{cases} \quad (6)$$

Чтобы доказать корректность такого задания функции, покажем, что условие

$$\exists (t_2, \ldots, t_b) \in V_{b-1} : f_b(x_1, \ldots, x_{n-1}, t_1, t_2, \ldots, t_b) = (y_1, \ldots, y_b)$$

может быть выполнено не более чем для одного $t_1 \in \{0, 1\}$. Предположим противное:

$$\exists (t'_2, \ldots, t'_b) \in V_{b-1} \ (f_b(x_1, \ldots, x_{n-1}, 0, t'_2, \ldots, t'_b) = (y_1, \ldots, y_b)), \\ \exists (t''_2, \ldots, t''_b) \in V_{b-1} \ (f_b(x_1, \ldots, x_{n-1}, 1, t''_2, \ldots, t''_b) = (y_1, \ldots, y_b)).$$

В таком случае выполнена система

$$\begin{cases} f(z'_1, z'_2, \ldots, z'_n) = y_1 = f(z''_1, z''_2, \ldots, z''_n); \\ f(z'_2, z'_3, \ldots, z'_{n+1}) = y_2 = f(z''_2, z''_3, \ldots, z''_{n+1}); \\ \ldots \\ f(z'_b, z'_{b+1}, \ldots, z'_{b+n-1}) = y_b = f(z''_b, z''_{b+1}, \ldots, z''_{b+n-1}); \\ z'_1 = z''_1 = x_1, \ldots, z'_{n-1} = z''_{n-1} = x_{n-1}, z'_n = 0, z''_n = 1; \\ z'_{n+1} = t'_2, \ldots, z'_{b+n-1} = t'_b; \\ z''_{n+1} = t''_2, \ldots, z''_{b+n-1} = t''_b, \end{cases}$$

что противоречит определению правого барьера длины b. Следовательно, функция $\widehat{f}$ определена корректным образом и является функцией, которая однозначно восстанавливает символ входной последовательности по $n-1$ предшествующим входным и b последующим выходным символам.

Логично предположить, что частично определенная булева функция, действие которой, в некотором смысле, обратно действию некоторой совершенно уравновешенной функции, сама должна быть сбалансированной.

Соответствующее утверждение сформулировано в теореме 5, доказательство которой опирается на леммы 1 и 2.

Предварительно введем два обозначения: для функции $f \in \mathcal{F}_n$ и натурального l для всякого набора $(x_1, \ldots, x_n) \in V_n$ определим величину $M'_{f_l}(x_1, \ldots, x_n)$:

$$M'_{f_l}(x_1, \ldots, x_n) = \\ = \sharp\{(y_1, \ldots, y_l) : \exists (t_{n+1}, \ldots, t_{l+n-1})\ (f_l(x_1, \ldots, x_n, t_{n+1}, \ldots, t_{l+n-1}) = (y_1, \ldots, y_l))\};$$

для всякого набора $(x_1, \ldots, x_{n-1}) \in V_{n-1}$ определим величину $M''_{f_l}(x_1, \ldots, x_{n-1})$:

$$M''_{f_l}(x_1, \ldots, x_{n-1}) = \\ = \sharp\{(y_1, \ldots, y_l) : \exists (t_n, \ldots, t_{l+n-1})\ (f_l(x_1, \ldots, x_{n-1}, t_n, t_{n+1}, \ldots, t_{l+n-1}) = (y_1, \ldots, y_l))\}.$$

Лемма 1. Пусть $f \in \mathcal{F}_n$ имеет правый барьер длины $b < +\infty$. Если для некоторого $l \geqslant b$ существует набор $(\widetilde{x}_1, \ldots, \widetilde{x}_n) \in V_n$, такой, что

$$M'_{f_l}(\widetilde{x}_1, \ldots, \widetilde{x}_n) \neq \frac{1}{2} M''_{f_l}(\widetilde{x}_1, \ldots, \widetilde{x}_{n-1}),$$

то существует набор $(x'_1 \ldots, x'_{m+n-2}) \in V_{m+n-2}$, для которого выполнена следующая система:

$$\begin{cases} m \geqslant 2; \\ (x'_m, \ldots, x'_{m+n-2}) = (x'_1, \ldots, x'_{n-1}); \\ M'_{f_l}(x'_j, \ldots, x'_{j+n-1}) \leqslant M''_{f_l}(x'_j, \ldots, x'_{j+n-2})/2,\ j = 1, \ldots, m-1; \\ \exists j \in \{1, 2, \ldots, m-1\}\ \left(M'_{f_l}(x'_j, \ldots, x'_{j+n-1}) < M''_{f_l}(x'_j, \ldots, x'_{j+n-2})/2\right). \end{cases}$$

Доказательство. Рассмотрим граф де Брейна порядка $n-1$ (напомним, что графом де Брейна порядка k называется ориентированный граф с 2^k вершинами и 2^{k+1} ориентированными ребрами, причем вершинам взаимно однозначно сопоставляются двоичные наборы длины k; дуга из вершины с набором $(x_1, \ldots, x_k)$ в вершину с набором $(y_1, \ldots, y_k)$ присутствует в графе тогда и только тогда, когда выполнены равенства $y_i = x_{i+1}$ для всех $i = 1, \ldots, k-1$). Введем дополнительные пометки на его дугах.

Дугу из вершины с набором $(x_1, \ldots, x_{n-1})$ в вершину с набором $(x_2, \ldots, x_n)$ пометим в соответствии со следующим правилом:

$$\begin{cases} 0, & M'_{f_l}(x_1, \ldots, x_n) < M''_{f_l}(x_1, \ldots, x_{n-1})/2; \\ 1, & M'_{f_l}(x_1, \ldots, x_n) = M''_{f_l}(x_1, \ldots, x_{n-1})/2; \\ 2, & M'_{f_l}(x_1, \ldots, x_n) > M''_{f_l}(x_1, \ldots, x_{n-1})/2. \end{cases}$$

Тогда условие леммы можно переформулировать следующим образом: в помеченном описанным выше способом графе де Брейна порядка $n-1$ существует хотя бы один цикл, не содержащий дуг с пометкой 2 и содержащий хотя бы одну дугу с пометкой 0.

Из существования у функции f правого барьера длины b следует: для $l \geqslant b$ верно, что для произвольного набора $(\overline{x}_1, \ldots, \overline{x}_{n-1})$ выполняется соотношение

$$\begin{aligned} \{(y_1, \ldots, y_l) : \exists (x_{n+1}, \ldots, x_{l+n-1}) \in V_{l-1} \\ (f_l(\overline{x}_1, \ldots, \overline{x}_{n-1}, 0, x_{n+1}, \ldots, x_{l+n-1}) = (y_1, \ldots, y_l))\} \cap \\ \cap \{(y_1, \ldots, y_l) : \exists (x_{n+1}, \ldots, x_{l+n-1}) \in V_{l-1} \\ (f_l(\overline{x}_1, \ldots, \overline{x}_{n-1}, 1, x_{n+1}, \ldots, x_{l+n-1}) = (y_1, \ldots, y_l))\} = \varnothing. \end{aligned}$$

Так как при этом выполняется соотношение

$$\begin{aligned} \{(y_1, \ldots, y_l) : \exists (x_{n+1}, \ldots, x_{l+n-1}) \in V_{l-1} \\ (f_l(\overline{x}_1, \ldots, \overline{x}_{n-1}, 0, x_{n+1}, \ldots, x_{l+n-1}) = (y_1, \ldots, y_l))\} \cup \\ \cup \{(y_1, \ldots, y_l) : \exists (x_{n+1}, \ldots, x_{l+n-1}) \in V_{l-1} \\ (f_l(\overline{x}_1, \ldots, \overline{x}_{n-1}, 1, x_{n+1}, \ldots, x_{l+n-1}) = (y_1, \ldots, y_l))\} = \\ = \{(y_1, \ldots, y_l) : \exists (x_n, \ldots, x_{l+n-1}) \in V_l \\ (f_l(\overline{x}_1, \ldots, \overline{x}_{n-1}, x_n, x_{n+1}, \ldots, x_{l+n-1}) = (y_1, \ldots, y_l))\}, \end{aligned}$$

то верно

$$M''_{f_l}(\overline{x}_1, \ldots, \overline{x}_{n-1}) = M'_{f_l}(\overline{x}_1, \ldots, \overline{x}_{n-1}, 0) + M'_{f_l}(\overline{x}_1, \ldots, \overline{x}_{n-1}, 1),$$

поэтому в нашем графе сумма пометок на любой паре выходящих из одной вершины дуг равна 2.

Удалим из графа все дуги с пометкой 2. Для завершения доказательства леммы достаточно построить в получившемся таким образом графе цикл, содержащий дугу с пометкой 0.

Начнем обход получившегося графа, начиная из вершины с набором $(\widetilde{x}_1, \ldots, \widetilde{x}_{n-1})$. Перейдем по единственной исходящей из нее дуге (с пометкой 0). Покажем, что из текущей вершины существует путь (возможно, нулевой длины) по дугам с пометкой 1 в некоторую вершину, из которой исходит дуга с пометкой 0. Достаточно рассмотреть два возможных случая: либо из нашей вершины по дугам с пометкой 1 доступны все вершины графа (и, в том числе, подходящая нам вершина с набором $(\widetilde{x}_1, \ldots, \widetilde{x}_{n-1})$), либо некоторая часть графа недоступна, что может произойти только из-за встреченных вершин, из которых в исходном графе исходила дуга с пометкой 2.

Итак, из исходной вершины по пути, состоящему ровно из одной дуги с пометкой 0 и, возможно, простой цепи из дуг с пометками 1, можно попасть в некоторую вершину, из которой тоже исходит дуга с пометкой 0. В случае, если эта новая вершина отлична от исходной, вновь перейдем по дуге с пометкой 0 и дугам с пометками 1 в некоторую вершину с исходящей из нее дугой с пометкой 0. Продолжая действовать таким образом, через некоторое число шагов попадем в одну из ранее встреченных вершин с исходящей дугой с пометкой 0. Искомый цикл построен, лемма доказана. ■

Лемма 2. Если $f \in \mathcal{F}_n$ имеет правый барьер длины $b < +\infty$, то равенство $M'_{f_l}(x_1,\dots,x_{n-1},0) = M'_{f_l}(x_1,\dots,x_{n-1},1) = M''_{f_l}(x_1,\dots,x_{n-1})/2$ выполняется для любого $l \geqslant b$ и любого $(x_1,\dots,x_{n-1}) \in V_{n-1}$.

Доказательство. С помощью вытекающего из определения правого барьера длины b равенства

$$\{(y_1,\dots,y_l) : \exists(t_{n+1},\dots,t_{l+n-1})\ (f_l(x_1,\dots,x_{n-1},0,t_{n+1},\dots,t_{l+n-1}) = (y_1,\dots,y_l))\} \cap$$
$$\cap\{(y_1,\dots,y_l) : \exists(t_{n+1},\dots,t_{l+n-1})\ (f_l(x_1,\dots,x_{n-1},1,t_{n+1},\dots,t_{l+n-1}) = (y_1,\dots,y_l))\} = \varnothing$$

и очевидного равенства

$$\{(y_1,\dots,y_l) : \exists(t_{n+1},\dots,t_{l+n-1})\ (f_l(x_1,\dots,x_{n-1},0,t_{n+1},\dots,t_{l+n-1}) = (y_1,\dots,y_l))\} \cup$$
$$\cup\{(y_1,\dots,y_l) : \exists(t_{n+1},\dots,t_{l+n-1})\ (f_l(x_1,\dots,x_{n-1},1,t_{n+1},\dots,t_{l+n-1}) = (y_1,\dots,y_l))\} =$$
$$= \{(y_1,\dots,y_l) : \exists(t_n,t_{n+1},\dots,t_{l+n-1})\ (f_l(x_1,\dots,x_{n-1},t_n,t_{n+1},\dots,t_{l+n-1}) = (y_1,\dots,y_l))\},$$

которые верны для любого $l \geqslant b$, легко получить, что для доказательства леммы достаточно показать невозможность существования для какого-либо $l \geqslant b$ набора $(\widetilde{x}_1,\dots,\widetilde{x}_n) \in V_n$, для которого выполняется неравенство

$$M'_{f_l}(x_1,\dots,x_n) < M''_{f_l}(x_1,\dots,x_{n-1})/2. \tag{7}$$

Предположим существование такого набора $(\widetilde{x}_1,\dots,\widetilde{x}_n) \in V_n$ для некоторого l, $l \geqslant b$, для которого выполняется (7).

Рассмотрим для произвольных фиксированных $\widehat{x}_1,\dots,\widehat{x}_n$ системы, описывающие отображения $f_l(\widehat{x}_1,\dots,\widehat{x}_{n-1},x_n,x_{n+1},\dots,x_{n+l-1})$, $f_l(\widehat{x}_1,\dots,\widehat{x}_{n-1},\widehat{x}_n,x_{n+1},\dots,x_{n+l})$:

$$\begin{cases} y_1 = f(\widehat{x}_1,\dots,\widehat{x}_{n-2},\widehat{x}_{n-1},x_n); \\ y_2 = f(\widehat{x}_2,\dots,\widehat{x}_{n-1},x_n,x_{n+1}); \\ \dots \\ y_l = f(x_l,\dots,x_{l+n-2},x_{l+n-2},x_{l+n-1}); \end{cases} \qquad \begin{cases} y_1 = f(\widehat{x}_1,\dots,\widehat{x}_{n-2},\widehat{x}_{n-1},\widehat{x}_n); \\ y_2 = f(\widehat{x}_2,\dots,\widehat{x}_{n-1},\widehat{x}_n,x_{n+1}); \\ \dots \\ y_l = f(x_l,\dots,x_{l+n-2},x_{l+n-2},x_{l+n-1}); \\ y_{l+1} = f(x_{l+1},\dots,x_{l+n-2},x_{l+n-1},x_{l+n}). \end{cases}$$

Так как $f(\widehat{x}_1,\dots,\widehat{x}_n)$ во второй системе определена однозначно и не зависит от $x_{n+1}\dots,x_{n+l}$, верно равенство

$$\sharp\{(y_1,\dots,y_{l+1}) : \exists(x_{n+1},\dots,x_{l+n})\ (f_{l+1}(\widehat{x}_1,\dots,\widehat{x}_n,x_{n+1},\dots,x_{l+n}) = (y_1,\dots,y_{l+1}))\} =$$
$$= \sharp\{(y_2,\dots,y_{l+1}) : \exists(x_{n+1},\dots,x_{l+n})\ (f_l(\widehat{x}_2,\dots,\widehat{x}_n,x_{n+1},\dots,x_{l+n}) = (y_2,\dots,y_{l+1}))\} =$$
$$= M''_{f_l}(\widehat{x}_2,\dots,\widehat{x}_n).$$

С другой стороны, вторая система получена из первой фиксацией аргумента x_n и добавлением одной новой строки. Поэтому выполняется неравенство

$$\sharp\{(y_1,\dots,y_{l+1}) : \exists(x_{n+1},\dots,x_{l+n})\ (f_{l+1}(\widehat{x}_1,\dots,\widehat{x}_n,x_{n+1},\dots,x_{l+n}) = (y_1,\dots,y_{l+1}))\} \leqslant$$
$$\leqslant 2\,\sharp\{(y_1,\dots,y_l) : \exists(x_{n+1},\dots,x_{l+n-1})\ (f_l(\widehat{x}_1,\dots,\widehat{x}_n,x_{n+1},\dots,x_{l+n-1}) = (y_1,\dots,y_l))\} =$$
$$= 2M'_{f_l}(\widehat{x}_1,\dots,\widehat{x}_n).$$

Окончательно получим: для всякого набора $(\widehat{x}_1,\dots,\widehat{x}_n) \in V_n$ выполнено

$$M''_{f_l}(\widehat{x}_2,\dots,\widehat{x}_n) \leqslant 2M'_{f_l}(\widehat{x}_1,\dots,\widehat{x}_n). \tag{8}$$

В лемме 1 доказано, что при нашем предположении о существовании набора $(\widetilde{x}_1, \ldots, \widetilde{x}_n) \in V_n$, для которого выполнено (7), существует набор $(x'_1, \ldots, x'_{m+n-2})$, такой, что:

$$\begin{cases} m \geqslant 2; \\ (x'_m, \ldots, x'_{m+n-2}) = (x'_1, \ldots, x'_{n-1}); \\ M'_{f_l}(x'_j, \ldots, x'_{j+n-1}) \leqslant M''_{f_l}(x'_j, \ldots, x'_{j+n-2})/2, \ j = 1, \ldots, m-1; \\ \exists j \in \{1, 2, \ldots, m-1\} \ \left(M'_{f_l}(x'_j, \ldots, x'_{j+n-1}) < M''_{f_l}(x'_j, \ldots, x'_{j+n-2})/2\right). \end{cases} \tag{9}$$

Из (8) следует, что $M''_{f_l}(x'_{j+1}, \ldots, x'_{j+n-1}) \leqslant 2M'_{f_l}(x'_j, \ldots, x'_{j+n-1})$ для $j = 1, \ldots, m-1$. Тогда, с учетом (9), верна следующая цепочка неравенств:

$$M''_{f_l}(x'_1, \ldots, x'_{n-1}) = M''_{f_l}(x'_m, \ldots, x'_{m+n-2}) \leqslant 2M'_{f_l}(x'_{m-1}, \ldots, x'_{m+n-2}) \leqslant$$
$$\leqslant 2M''_{f_l}(x'_{m-1}, \ldots, x'_{m+n-3})/2 = M''_{f_l}(x'_{m-1}, \ldots, x'_{m+n-3}) \leqslant \ldots \leqslant M''_{f_l}(x'_1, \ldots, x'_{n-1}),$$

причем хотя бы одно из неравенств этой цепочки — строгое.

Получили $M''_{f_l}(x'_1, \ldots, x'_{n-1}) < M''_{f_l}(x'_1, \ldots, x'_{n-1})$. Это противоречие завершает доказательство леммы. ■

Теорема 5. Пусть $f \in \mathcal{F}_n$, f имеет правый барьер длины $b < +\infty$. Тогда функция $\widehat{f}$, определенная согласно (6), является уравновешенной (в соответствующем смысле для частично определенных булевых функций):

$$\begin{aligned} &\sharp\{(x_1, \ldots, x_{n-1}, y_1, \ldots, y_b) : \widehat{f}(x_1, \ldots, x_{n-1}, y_1, \ldots, y_b) = 0\} = \\ &= \sharp\{(x_1, \ldots, x_{n-1}, y_1, \ldots, y_b) : \widehat{f}(x_1, \ldots, x_{n-1}, y_1, \ldots, y_b) = 1\}. \end{aligned} \tag{10}$$

Доказательство. Равенство (10), согласно определению $\widehat{f}$, можно переписать в следующем виде:

$$\begin{aligned} \sharp\{(x_1, \ldots, x_{n-1}, y_1, \ldots, y_b) : \exists (t_{n+1}, \ldots, t_{b+n-1}) \\ (f_b(x_1, \ldots, x_{n-1}, 0, t_{n+1}, \ldots, t_{b+n-1}) = (y_1, \ldots, y_b))\} = \\ = \sharp\{(x_1, \ldots, x_{n-1}, y_1, \ldots, y_b) : \exists (t_{n+1}, \ldots, t_{b+n-1}) \\ (f_b(x_1, \ldots, x_{n-1}, 1, t_{n+1}, \ldots, t_{b+n-1}) = (y_1, \ldots, y_b))\}. \end{aligned}$$

По определению величины M'_{f_l} это равенство эквивалентно следующему:

$$\sum_{(x_1, \ldots, x_{n-1}) \in V_{n-1}} M'_{f_l}(x_1, \ldots, x_{n-1}, 0) = \sum_{(x_1, \ldots, x_{n-1}) \in V_{n-1}} M'_{f_l}(x_1, \ldots, x_{n-1}, 1). \tag{11}$$

Последнее равенство верно в силу леммы 2. ■

Замечание 3. Нетрудно заметить, что из леммы 2 следует и более сильное утверждение: уравновешенной (в соответствующим смысле для частично определенных булевых функций) является не только функция $\widehat{f}$, но и любая ее подфункция, полученная фиксацией переменных $x_1, \ldots, x_{n-1}$.

Рассмотренный в замечании 3 факт можно проинтерпретировать следующим образом: если использовать функцию $\widehat{f}$ для восстановления символов входной последовательности кодирующего устройства с фильтрующей функцией f с правым барьером конечной длины, то при восстановлении очередного входного символа при известных предыдущих $n-1$ входных символах ровно половина множества возможных l-грамм последующих выходных символов будет свидетельствовать о равенстве очередного входного символа нулю, ровно половина — о равенстве единице.

Напоследок заметим, что данное свойство нельзя обобщить для произвольной ситуации, в которой возможно восстановление очередного входного символа кодирующего устройства с совершенно уравновешенной фильтрующей функцией по предшествующим $n-1$ входным и последующим l выходным символам. Например, при $f(x_1, x_2, x_3) = x_1 \oplus x_2x_3 \in \mathcal{PB}_3$, $n = 2$ и $l = 2$, если требуется восстановить x_3, зная x_1, x_2, $y_1 = f(x_1, x_2, x_3) = x_1 \oplus x_2x_3$ и $y_2 = f(x_2, x_3, x_4) = x_2 \oplus x_3x_4$, то в ситуации, когда $x_1 = 0$ и $x_2 = 1$, символ x_3 восстановим по паре (y_1, y_2), однако о равенстве x_3 нулю свидетельствует только одна пара — $(y_1, y_2) = (0, 1)$, тогда как о равенстве x_3 единице свидетельствуют две: $(1, 0)$ и $(1, 1)$.

Кроме того, нельзя обобщить данное свойство и на случай произвольного отображения вида $V_{n+b-1} \to V_b$, которое является уравновешенным и у которого n-й входной символ можно восстановить по предшествующим $n-1$ входным и b последующим выходным символам. Далее приведен пример отображения F, для которого аналог функции $\widehat{f}$ не является уравновешенным.

Пусть $n = b = 2$; $F : V_{n+b-1} \to V_b$; $\widehat{f} : V_{n+b-1} \to \{0, 1, *\}$.

$F(x_1, x_2, x_3) = (y_1, y_2)$

x_1	x_2	x_3	y_1	y_2
0	0	0	0	1
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	0

$\widehat{f}(x_1, y_1, y_2) = x_2$

x_1	y_1	y_2	x_2
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	*
1	0	0	1
1	0	1	*
1	1	0	1
1	1	1	0

ЛИТЕРАТУРА

1. *Сумароков С. Н.* Запреты двоичных функций и обратимость для одного класса кодирующих устройств // Обозрение прикладной и промышленной математики. 1994. Т. 1. Вып. 1. С. 33–55.
2. *Логачев О. А., Смышляев С. В., Ященко В. В.* Новые методы изучения совершенно уравновешенных булевых функций // Дискретная математика. 2009. Т. 21. Вып. 2. С. 51–74.
3. *Смышляев С. В.* О некоторых свойствах совершенно уравновешенных булевых функций // Материалы Четвертой Междунар. научн. конф. по проблемам безопасности и противодействия терроризму (МГУ им. М. В. Ломоносова, Москва, 30–31 октября 2008). М.: МЦНМО, 2009. С. 57–64.
4. *Смышляев С. В.* Барьеры совершенно уравновешенных булевых функций // Дискретная математика (в печати).
5. *Логачев О. А.* Об одном классе совершенно уравновешенных булевых функций // Материалы Третьей Междунар. научн. конф. по проблемам безопасности и противодействия терроризму (МГУ им. М. В. Ломоносова, Москва, 25–27 октября 2007). М.: МЦНМО, 2008. С. 137–141.
6. *Смышляев С. В.* О совершенно уравновешенных булевых функциях без барьера // Материалы Восьмой Междунар. научн. конф. «Дискретные модели в теории управляющих систем» (МГУ им. М. В. Ломоносова, Москва, 6–9 апреля 2009). М.: МАКС Пресс, 2009. С. 278–284.
7. *Логачев О. А., Сальников А. А., Ященко В. В.* Булевы функции в теории кодирования и криптологии. М.: МЦНМО, 2004.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/7/2

УДК 004.94

АНАЛИЗ В РАМКАХ БАЗОВОЙ РОЛЕВОЙ ДП-МОДЕЛИ БЕЗОПАСНОСТИ СИСТЕМ С ПРОСТЫМИ ТРАЕКТОРИЯМИ ФУНКЦИОНИРОВАНИЯ[1]

П. Н. Девянин

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

E-mail: peter_devyanin@hotmail.com

В рамках базовой ролевой ДП-модели анализируются условия передачи прав доступа и реализации информационных потоков по памяти. При этом рассматриваются только простые траектории функционирования системы, когда взаимодействует произвольное число субъект-сессий, и они не получают доступа владения друг к другу с использованием информационных потоков по памяти к функционально ассоциированным с субъект-сессиями сущностям.

Ключевые слова: *компьютерная безопасность, ролевая модель, ДП-модели.*

1. Основные элементы базовой ролевой ДП-модели

На основе базовой ролевой ДП-модели (БР ДП-модели) [1, 2] рассмотрим условия передачи прав доступа и реализации информационных потоков по памяти для случая, когда на траекториях функционирования системы субъект-сессии не получают доступа владения друг к другу с использованием информационных потоков по памяти к функционально ассоциированным с субъект-сессиями сущностям.

Основными элементами БР ДП-модели являются:

$E = O \cup C$ — множество сущностей, где O — множество объектов, C — множество контейнеров и $O \cap C = \varnothing$;

U — множество пользователей;

L_U — множество доверенных пользователей;

N_U — множество недоверенных пользователей;

$S \subseteq E$ — множество субъект-сессий пользователей;

L_S — множество доверенных субъект-сессий;

N_S — множество недоверенных субъект-сессий;

R — множество ролей;

AR — множество административных ролей;

$R_r = \{read_r, write_r, append_r, execute_r, own_r\}$ — множество видов прав доступа;

$R_a = \{read_a, write_a, append_a, own_a\}$ — множество видов доступа;

$R_f = \{write_m, write_t\}$ — множество видов информационных потоков;

$A \subseteq S \times E \times R_a$ — множество доступов субъект-сессий к сущностям;

$F \subseteq E \times E \times R_f$ — множество информационных потоков между сущностями;

$P \subset E \times R_r$ — множество прав доступа к сущностям;

[1]Работа выполнена при поддержке гранта МД–2.2010.10.

$UA : U \to 2^R$ — функция авторизованных ролей пользователей;

$AUA : U \to 2^{AR}$ — функция авторизованных административных ролей пользователей;

$PA : R \to 2^P$ — функция прав доступа ролей;

$user : S \to U$ — функция принадлежности субъект-сессии пользователю;

$roles : S \to 2^R \cup 2^{AR}$ — функция текущих ролей субъект-сессий;

$can\_manage\_rights : AR \to 2^R$ — функция администрирования прав доступа ролей;

$H_E : E \to 2^E$ — функция иерархии сущностей;

$H_R : R \to 2^R$ — функция иерархии ролей;

$H_{AR} : AR \to 2^{AR}$ — функция иерархии административных ролей;

$G = (PA, user, roles, A, F, H_E)$ — состояние системы;

$\Sigma(G^*, OP)$ — система, при этом G^* — множество всех возможных состояний, OP — множество правил преобразования состояний;

$G \vdash_{op} G'$ — переход системы $\Sigma(G^*, OP)$ из состояния G в состояние G' с использованием правила преобразования состояний $op \in OP$;

$\Sigma(G^*, OP, G_0)$ — система $\Sigma(G^*, OP)$ с начальным состоянием G_0;

$[s] \subset E \cup U$ — множество сущностей, функционально ассоциированных с субъект-сессией s (при этом по определению выполняется условие $s \in [s]$), и пользователей, каждый из которых может создать субъект-сессию, являющуюся функционально ассоциированной сущностью с субъект-сессией s;

$fa : U \times E \to 2^E \cup 2^U$ — функция, задающая множества сущностей, функционально ассоциированных с субъект-сессией, при ее создании пользователем (или от имени пользователя другой субъект-сессией) из сущности. При этом если пользователь $u \in U$ или субъект-сессия от имени пользователя u не могут создать из сущности $e \in E$ новую субъект-сессию, то по определению $fa(u, e) = \varnothing$. Кроме того, если для пользователя $u \in U$ и сущности $e \in E$ существует пользователь $x \in U$, такой, что выполняется условие $x \in fa(u, e)$, то по определению будем считать, что пользователь x может создать субъект-сессию, которая будет являться функционально ассоциированной сущностью с субъект-сессией, создаваемой пользователем u из сущности e. По определению выполняется условие: для каждой субъект-сессии $s \in S$ существует единственная сущность $e_s \in E$, такая, что справедливо равенство $fa(user(s), e_s) = [s]$;

$y(E) \subset L_S \times E$ — множество пар вида (доверенная субъект-сессия, сущность), относительно которых корректна доверенная субъект-сессия y;

$de\_facto\_roles : S \to 2^{R \cup AR}$ — функция фактических текущих ролей субъект-сессий, при этом по определению в каждом состоянии системы $G = (PA, user, roles, A, F, H_E)$ для каждой субъект-сессии $s_1 \in S$ выполняется равенство:

$de\_facto\_roles(s_1) = roles(s_1) \cup \{r \in R \cup AR : \exists s_2 \in S\ [(s_1, s_2, own_a) \in A\ \&\ \&\ r \in roles(s_2)]\}$;

$de\_facto\_rights : S \to 2^P$ — функция фактических текущих прав доступа субъект-сессий, при этом по определению в каждом состоянии системы $G = (PA, user, roles, A, F, H_E)$ для каждой субъект-сессии $s \in S$ выполняется равенство:

$de\_facto\_rights(s) = \{p \in P : \exists r \in de\_facto\_roles(s)\ [p \in PA(r)]\}$;

$de\_facto\_actions : S \to 2^P \times 2^R$ — функция фактических возможных действий субъект-сессий, при этом по определению в каждом состоянии системы $G = (PA, user, roles, A, F, H_E)$ для каждой субъект-сессии $s_1 \in S$ выполняется равенство:

$de\_facto\_actions(s_1) = (PA(roles(s_1)) \times can\_manage\_rights(roles(s_1) \cap AR)) \cup \cup\{(p,r) \in P \times R : \exists s_2 \in S\ \exists (s_1, s_2, own_a) \in A\ [r \in can\_manage\_rights(roles(s_2) \cap AR)\ \&\ \&\ p \in PA(roles(s_2))]\}$.

В БР ДП-модели определены следующие правила преобразования состояний:

— монотонные: $take\_role(x,r)$, $grant\_right(x,r,(y,\alpha_r))$, $create\_entity(x,r,y,z)$, $create\_first\_session(u,r,y,z)$, $create\_session(x,w,r,y,z)$, $rename\_entity(x,y,z)$, $control(x,y,z)$, $access\_own(x,y)$, $take\_access\_own(x,y,z)$, $access\_read(x,y)$, $access\_write(x,y)$, $access\_append(x,y)$, $flow(x,y,y',z)$, $find(x,y,z)$, $post(x,y,z)$, $pass(x,y,z)$, $take\_flow(x,y)$;

— немонотонные: $remove\_role(x,r)$, $remove\_right(x,r,(y,\alpha_r))$, $delete\_entity(x,y,z)$.

Используем следующие предположение и определения БР ДП-модели.

Предположение 1. Каждые пользователь или субъект-сессия системы $\Sigma(G^*, OP)$ вне зависимости от имеющихся у них авторизованных ролей являются либо доверенными, либо недоверенными. Доверенные пользователи или субъект-сессии не создают новых субъект-сессий. Каждые недоверенный пользователь или субъект-сессия могут создать только недоверенную субъект-сессию.

Определение 1. Назовем траекторию функционирования системы $\Sigma(G^*, OP)$ траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, если при ее реализации используются только монотонные правила преобразования состояний и доверенные субъект-сессии:

— не берут роли в множество текущих ролей;

— не дают другим ролям права доступа к сущностям;

— не получают доступа владения к субъект-сессиям.

Определение 2. Траекторию без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, назовем простой, если при ее реализации для $0 \leqslant i \leqslant N$ каждое правило op_i не является правилом вида $control(x,y,z)$, использующим для получения доступа владения субъект-сессией x к субъект-сессии y информационный поток по памяти $(x, z, write_m)$, где $z \in [y]$.

Определение 3. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь $x \in N_U$ и субъект-сессия или недоверенный пользователь $y \in N_U \cup S_0$, такие, что $x \neq y$. Определим предикат $simple\_can\_access\_own(x,y,G_0)$, который будет истинным тогда и только тогда, когда существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существуют субъект-сессии s_x, $s_y \in S_N$, такие, что $user_N(s_x) = x$, или $s_y = y$, или $user_N(s_y) = y$ и выполняется условие $(s_x, s_y, own_a) \in A_N$.

Определение 4. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют субъект-сессии или недоверенные пользователи $x, y \in N_U \cup S$. Определим предикат $simple\_directly\_access\_own(x,y,G)$, который будет истинным тогда и только тогда, когда или $x = y$, или выполняется одно из следующих условий 1–6.

1. Если $y \in N_U$ и $x \in N_U$, то существуют сущность $e_y \in E$ и роль $r_y \in R$, такие, что $(e_y, execute_r) \in PA(UA(y))$, $r_y \in can\_manage\_rights(AUA(y))$ и выполняется одно из условий:

— $r_y \in UA(x)$;
— $x \in fa(y, e_y)$.

2. Если $y \in N_U$ и $x \in N_S \cap S$, то существуют сущность $e_y \in E$ и роль $r_y \in R$, такие, что $(e_y, execute_r) \in PA(UA(y))$, $r_y \in can\_manage\_rights(AUA(y))$ и выполняется одно из условий:

— $r_y \in UA(user(x))$;
— $x \in fa(y, e_y)$.

3. Если $y \in N_U$ и $x \in L_S \cap S$, то существуют сущность $e_y \in E$ и роль $r_y \in R$, такие, что $(e_y, execute_r) \in PA(UA(y))$, $r_y \in can\_manage\_rights(AUA(y))$ и выполняется одно из условий:

— $r_y \in roles(x)$;
— $x \in fa(y, e_y)$.

4. Если $y \in S$ и $x \in N_U$, то выполняется одно из условий:

— $(y, own_r) \in PA(UA(x))$;
— $x \in [y]$.

5. Если $y \in S$ и $x \in N_S \cap S$, то выполняется одно из условий:

— $(y, own_r) \in PA(UA(user(x)))$;
— $x \in [y]$;
— $(x, y, own_a) \in A$.

6. Если $y \in S$ и $x \in L_S \cap S$, то выполняется одно из условий:

— $(y, own_r) \in PA(roles(x))$;
— $x \in [y]$;
— $(x, y, own_a) \in A$.

Определение 5. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существует субъект-сессия или недоверенный пользователь $x \in N_U \cup S$. Назовем множество $X \subset N_U \cup S$ островом субъект-сессии или недоверенного пользователя x, если $X = \{x\} \cup \{y \in (N_U \cup S) \setminus \{x\}$: существует последовательность $s_1 = x$ и $s_2, \ldots, s_m \in (N_U \cup S) \setminus \{x\}$, где $s_m = y$ и $m \geqslant 2$, такая, что для каждого i, $1 \leqslant i < m$, истинен предикат $simple\_directly\_access\_own(s_i, s_{i+1}, G)\}$. Определим функцию $island : N_U \cup S \to 2^{N_u \cup S}$, задающую для каждых субъект-сессии или недоверенного пользователя соответствующий им остров. При этом используются следующие обозначения:

$island\_roles : N_U \cup (N_S \cap S) \to 2^R \cup 2^{AR}$ — функция, задающая для каждых недоверенного пользователя или недоверенной субъект-сессии $x \in N_U \cup (N_S \cap S)$ роли, которыми обладают все субъект-сессии или недоверенные пользователи, принадлежащие острову x. При этом по определению справедливо равенство $island\_roles(x) = \{r \in R \cup AR : \exists y \in island(x)\ [(y \in N_U \,\&\, r \in UA(y) \cup AUA(y)) \vee (y \in N_S \cap S\ \&\ \&\ r \in UA(user(y)) \cup AUA(user(y))) \vee (y \in L_S \cap S \,\&\, r \in roles(y))]\}$;

$island\_rights : N_U \cup (N_S \cap S) \to 2^P$ — функция, задающая для каждых недоверенного пользователя или недоверенной субъект-сессии $x \in N_U \cup (N_S \cap S)$ права доступа, которыми обладают все субъект-сессии или недоверенные пользователи, принадлежащие острову x. При этом по определению справедливо равенство $island\_rights(x) = \{p \in P : \exists r \in island\_roles(x)\ [p \in PA(r)]\}$;

$island\_actions : N_U \cup (N_S \cap S) \to 2^P \times 2^R$ — функция, задающая для каждых недоверенного пользователя или недоверенной субъект-сессии $x \in N_U \cup$

$\cup(N_S \cap S)$ возможные действия, которыми обладают все субъект-сессии или недоверенные пользователи, принадлежащие острову x. При этом по определению справедливо равенство $island\_actions(x) = \{(p,r) \in P \times R : \exists y \in island(x)$ $[(y \in N_U \ \& \ (p,r) \in PA(UA(y)) \times can\_manage\_rights(AUA(y))) \vee (y \in N_S \cap S \ \&$ $\& \ (p,r) \in PA(UA(user(y))) \times can\_manage\_rights(AUA(user(y)))) \vee (y \in L_S \cap S \ \&$ $\& \ (p,r) \in PA(roles(y)) \times can\_manage\_rights(roles(y) \cap AR))]\}$.

Определение 6. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенная субъект-сессия или недоверенный пользователь $x \in N_U \cup (N_S \cap S)$ и субъект-сессии или недоверенные пользователи $y, z \in N_U \cup S$. Будем говорить, что субъект-сессия или недоверенный пользователь y соединяются простым мостом с субъект-сессией или недоверенным пользователем z через недоверенную субъект-сессию или недоверенного пользователя x, если $z \in island(x)$ и существует роль $r_y \in R$, такая, что выполняются следующие два условия.

1. Или $y \in N_U$ и $r_y \in UA(y)$, или $y \in N_S \cap S$ и $r_y \in UA(user(y))$, или $y \in L_S \cap S$ и $r_y \in roles(y)$.

2. Или $z \in N_U$ и $r_y \in can\_manage\_rights(AUA(z))$, или $z \in N_S \cap S$ и $r_y \in can\_manage\_rights(AUA(user(z)))$, или $z \in L_S \cap S$ и $r_y \in can\_manage\_rights(roles(z) \cap AR)$.

При этом используется следующее обозначение:

$is\_simple\_bridge : (N_U \cup (N_S \cap S)) \times (N_U \cup S) \times (N_U \cup S) \to \{\text{true}, \text{false}\}$ — функция, для которой по определению справедливо равенство $is\_simple\_bridge(x, y, z) = \text{true}$ тогда и только тогда, когда y соединен простым мостом с z через x, где $x \in N_U \cup \cup(N_S \cap S)$, $y, z \in N_U \cup S$.

Определение 7. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенная субъект-сессия или недоверенный пользователь $x \in N_U \cup (N_S \cap S)$ и субъект-сессии или недоверенные пользователи $y, z \in N_U \cup S$. Будем говорить, что субъект-сессия или недоверенный пользователь y соединяются мостом с субъект-сессией или недоверенным пользователем z через недоверенную субъект-сессию или недоверенного пользователя x, когда существуют субъект-сессии или недоверенные пользователи $v, w \in N_U \cup S$ и роли $r_v, r_y \in R$, такие, что $v, w, z \in island(x)$, $w, z \in island(v)$, $z \in island(w)$ и выполняются следующие условия.

1. Или $y \in N_U$ и $r_y \in UA(y)$, или $y \in N_S \cap S$ и $r_y \in UA(user(y))$, или $y \in L_S \cap S$ и $r_y \in roles(y)$.

2. Или $v \in N_U$ и $r_v \in UA(v)$, $r_y \in can\_manage\_rights(AUA(v))$, или $v \in N_S \cap S$ и $r_v \in UA(user(v))$, $r_y \in can\_manage\_rights(AUA(user(v)))$, или $v \in L_S \cap S$ и $r_v \in roles(v)$, $r_y \in can\_manage\_rights(roles(v) \cap AR)$.

3. Или $w \in N_U$ и $r_v \in can\_manage\_rights(AUA(w))$, или $w \in S$ и $(w, own_r) \in \in PA(r_v)$.

При этом используется следующее обозначение:

$is\_bridge : (N_U \cup (N_S \cap S)) \times (N_U \cup S) \times (N_U \cup S) \to \{\text{true}, \text{false}\}$ — функция, для которой по определению справедливо равенство $is\_bridge(x, y, z) = \text{true}$ тогда и только тогда, когда y соединен мостом с z через x, где $x \in N_U \cup (N_S \cap S)$, $y, z \in N_U \cup S$.

В [2] обоснованы алгоритмически проверяемые необходимые и достаточные условия истинности предиката $simple\_can\_access\_own(x, y, G_0)$.

Утверждение 1. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь или недоверенная

субъект-сессия $x \in N_U \cup (N_S \cap S_0)$ и субъект-сессия или недоверенный пользователь $y \in island(x) \setminus \{x\}$. Тогда справедливо одно из предложений:

— если $x \in N_U$, то истинен предикат $simple\_can\_access\_own(x, y, G_0)$;
— если $x \in N_S \cap S_0$, то истинен предикат $simple\_can\_access\_own(user_0(x), y, G_0)$.

Следствие 1. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь или недоверенная субъект-сессия $x \in N_U \cup (N_S \cap S_0)$. Тогда существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует недоверенная субъект-сессия $s_x \in N_S \cap S_N$, такая, что либо $user_N(s_x) = x$, либо $s_x = x$ и выполняются условия:

— $island\_roles(x) = island\_roles(s_x) \subset de\_facto\_roles_N(s_x)$ (множество фактических ролей субъект-сессии s_x включает все роли субъект-сессий или пользователей, принадлежащих ее острову);
— $island\_rights(x) = island\_rights(s_x) \subset de\_facto\_rights_N(s_x)$ (множество фактических прав доступа субъект-сессии s_x включает все права доступа субъект-сессий или пользователей, принадлежащих ее острову);
— $island\_actions(x) = island\_actions(s_x) \subset de\_facto\_actions_N(s_x)$ (множество фактических возможных действий субъект-сессии s_x включает все возможные действия субъект-сессий или пользователей, принадлежащих ее острову).

Теорема 1. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь $x \in N_U$ и субъект-сессия или недоверенный пользователь $y \in N_U \cup S_0$, такие, что $x \neq y$. Предикат $simple\_can\_access\_own(x, y, G_0)$ является истинным тогда и только тогда, когда существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_m \in N_U \cup (N_S \cap S_0)$ и субъект-сессий или недоверенных пользователей $y_1, \ldots, y_m \in N_U \cup S_0$, где $m \geqslant 1$, таких, что $x_1 = x$, $y_m = y$, $y_i \in island(x_i)$ для $1 \leqslant i \leqslant m$ и выполняются следующие условия.

1. Если $m \geqslant 2$, то справедливо равенство $is\_bridge(x_m, y_{m-1}, y) = \text{true}$.

2. Если $m \geqslant 3$, то для каждого i, $2 \leqslant i < m$, или $is\_bridge(x_i, y_{i-1}, y_i) = \text{true}$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = \text{true}$.

2. Условия передачи прав доступа

В рамках БР ДП-модели рассмотрим условия передачи прав доступа с участием произвольного числа субъект-сессий для простых траекторий функционирования системы. Дадим определение.

Определение 8. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют пользователь $x \in U_0$ и право доступа к сущности $(e, \alpha) \in P_0$. Определим предикат $simple\_can\_share((e, \alpha), x, G_0)$, который будет истинным тогда и только тогда, когда существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует субъект-сессия $s_x \in S_N$, такая, что $user_N(s_x) = x$ и выполняется условие $(e, \alpha) \in de\_facto\_rights_N(s_x)$.

Определим и обоснуем алгоритмически проверяемые необходимые и достаточные условия истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$ для случая, когда $x \in N_U$.

Теорема 2. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь $x \in N_U$ и право доступа к сущности $(e, \alpha) \in P_0$. Предикат $simple\_can\_share((e, \alpha), x, G_0)$ является истинным тогда и только тогда, когда выполняется одно из условий.

1. Выполняется условие $(e, \delta) \in PA_0(UA_0(x))$, где $\delta \in \{\alpha, own_r\}$.

2. Существует субъект-сессия или недоверенный пользователь $y \in N_U \cup S_0$, истинен предикат $simple\_can\_access\_own(x, y, G_0)$ и выполняется одно из условий:

— $y \in N_U$ и $(e, \alpha) \in PA_0(UA_0(y))$;
— $y \in N_S \cap S_0$ и $(e, \alpha) \in PA_0(UA_0(user_0(y)))$;
— $y \in L_S \cap S_0$ и $(e, \alpha) \in PA_0(roles_0(y))$.

3. Существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_m \in N_U \cup (N_S \cap S_0)$, субъект-сессий или недоверенных пользователей $y_1, \ldots, y_m \in N_U \cup S_0$, где $m \geqslant 2$, таких, что $x_1 = x$, $y_i \in island(x_i)$, где $1 \leqslant i \leqslant m$, и выполняется одно из условий:

— $y_m \in N_U$ и $(e, own_r) \in PA_0(UA_0(y_m))$;
— $y_m \in N_S \cap S_0$ и $(e, own_r) \in PA_0(UA_0(user_0(y_m)))$;
— $y_m \in L_S \cap S_0$ и $(e, own_r) \in PA_0(roles_0(y_m))$.

При этом справедливо равенство $is\_simple\_bridge(x_m, y_{m-1}, y_m) = \text{true}$, и для каждого $2 \leqslant i \leqslant m$ справедливо равенство или $is\_bridge(x_i, y_{i-1}, y_i) = \text{true}$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = \text{true}$.

Доказательство. Докажем достаточность выполнения условий теоремы для истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$ при $x \in N_U$.

Пусть выполнено условие 1 теоремы. Тогда по предположению 1 существуют сущность $e_x \in E_0$, административная роль $ar_x \in AUA_0(x)$ и роль $r_x \in can\_manage\_rights(ar_x)$, такие, что $(e_x, execute_r) \in PA_0(UA_0(x))$. Положим

$op_1 = create\_first\_session(x, r_x, e_x, s_x)$.

По условию 1 существует роль $r_e \in UA_0(x)$, такая, что $(e, \delta) \in PA_0(r_e)$, где $\delta \in \{\alpha, own_r\}$. Тогда положим

$op_2 = take\_role(s_x, r_e)$.

Если $\delta = \alpha$, то положим $N = 2$.

Если $\delta = own_r$, то положим

$op_3 = take\_role(s_x, ar_x)$;

$op_4 = grant\_right(s_x, r_e, (e, \alpha))$;

$N = 4$.

Таким образом, существует $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$ — простая траектория без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и в состоянии G_N выполняется условие $user_N(s_x) = x$ и право доступа к сущности $(e, \alpha) \in de\_facto\_rights_N(s_x)$. Следовательно, по определению 8 предикат $simple\_can\_share((e, \alpha), x, G_0)$ является истинным.

Пусть выполнено условие 2 теоремы. Тогда существует субъект-сессия или недоверенный пользователь $y \in N_U \cup S_0$, истинен предикат $simple\_can\_access\_own(x, y, G_0)$ и по определению 3 существуют состояния $G_1, \ldots, G_M$ и правила преобразования состояний $op_1, \ldots, op_M$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_M} G_M$, где $M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, существуют субъект-сессии $s_x, s_y \in S_M$, такие, что $user_M(s_x) = x$, или $s_y = y$, или $user_M(s_y) = y$, и выполняется условие $(s_x, s_y, own_a) \in A_M$. По условию 1 теоремы возможны три случая.

Первый случай: выполняются условия $y \in N_U$ и $(e, \alpha) \in PA_0(UA_0(y))$. Тогда существует роль $r_y \in UA_0(y) = UA_M(user_M(s_y))$, такая, что $(e, \alpha) \in PA_0(r_y)$. Положим

$op_{M+1} = take\_role(s_y, r_y)$;

$N = M + 1$.

Второй случай: выполняются условия $y \in N_S \cap S_0$ и $(e, \alpha) \in PA_0(UA_0(user_0(y)))$. Тогда существует роль $r_y \in UA_0(user_0(y))$, такая, что $(e, \alpha) \in PA_0(r_y)$. Положим

$op_{M+1} = take\_role(y, r_y)$;

$N = M + 1$.

Третий случай: выполняется условие $y \in L_S \cap S_0$ и $(e, \alpha) \in PA_0(roles_0(y))$. Тогда положим $N = M$. Следовательно, траектория $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа и выполняется условие $(e, \alpha) \in de\_facto\_rights_N(s_x)$. Значит, по определению 8 предикат $simple\_can\_share((e, \alpha), x, G_0)$ является истинным.

Пусть выполнено условие 3 теоремы, тогда выполняются условия $m \geqslant 2$ и $is\_simple\_bridge(x_m, y_{m-1}, y_m) = \text{true}$. Выберем минимальное $2 \leqslant k \leqslant m$, такое, что справедливы равенства $is\_simple\_bridge(x_l, y_{l-1}, y_l) = \text{true}$, где $k \leqslant l \leqslant m$. Для каждого простого моста, соединяющего субъект-сессию или недоверенного пользователя y_{l-1} с субъект-сессией или недоверенным пользователем y_l через недоверенную субъект-сессию или недоверенного пользователя x_l, выполняется условие $y_l \in island(x_l)$ и существует роль $r_{y_{l-1}}$, удовлетворяющая условиям определения 6, где $k \leqslant l \leqslant m$.

По утверждению 1 и определению 3 существуют состояния $G_1, \ldots, G_M$ и правила преобразования состояний $op_1, \ldots, op_M$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_M} G_M$, где $M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и для каждого $k \leqslant l \leqslant m$ существует субъект-сессия $s_{x_l} \in S_M$, такая, что или $user_M(s_{x_l}) = x_l$, или $s_{x_l} = x_l$, существует субъект-сессия $s_{y_l} \in S_M$, такая, что $user_M(s_{y_l}) = y$ или $s_{y_l} = y_l$, и выполняется условие $(s_{x_l}, s_{y_l}, own_a) \in A_M$. Положим

$op_{M+1} = grant\_right(s_{x_m}, r_{y_{m-1}}, (e, own_r))$;

$\ldots$

$op_{M+m-k} = grant\_right(s_{x_{k+1}}, r_{y_k}, (e, own_r))$;

$op_{M+m-k+1} = grant\_right(s_{x_k}, r_{y_{k-1}}, (e, \alpha))$;

$K = M + m - k + 1$.

Возможны два случая.

Первый случай: справедливо равенство $k = 2$. По условию теоремы $y_1 \in island(x)$, следовательно, по утверждению 1 существуют состояния $G_{K+1}, \ldots, G_N$ и правила преобразования состояний $op_{K+1}, \ldots, op_N$, такие, что $G_K \vdash_{op_{K+1}} G_{K+1} \vdash_{op_{K+2}} \ldots \vdash_{op_N} G_N$, где $N \geqslant K$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, существуют субъект-сессии $s_x, s_{y_1} \in S_N$, такие, что выполняются условия:

— $user_N(s_x) = x$ или $s_x = x$;
— существует субъект-сессия $s_{y_1} \in S_N$, такая, что $user_N(s_{y_1}) = y_1$ или $s_{y_1} = y_1$;
— $(s_x, s_{y_1}, own_a) \in A_N$.

Второй случай: выполняется неравенство $k > 2$. Тогда выполняется условие $is\_bridge(x_{k-1}, y_{k-2}, y_{k-1}) = \text{true}$ и последовательность недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_{k-1} \in N_U \cup (N_S \cap S_0)$, субъект-сессий или недоверенных пользователей $y_1, \ldots, y_{k-1} \in N_U \cup S_0$ удовлетворяет условиям теоремы 1.

Следовательно, истинен предикат $simple\_can\_access\_own(x, y_{k-1}, G_0)$, и по определению 3 существуют состояния $G_{K+1}, \ldots, G_N$ и правила преобразования состояний $op_{K+1}, \ldots, op_N$, такие, что $G_K \vdash_{op_{K+1}} G_{K+1} \vdash_{op_{K+2}} \ldots \vdash_{op_N} G_N$, где $N \geqslant K$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, существуют субъект-сессии $s_x, s_{y_{k-1}} \in S_N$, такие, что выполняются условия:

— $user_N(s_x) = x$;
— $s_{y_{k-1}} = y_{k-1}$ или $user_N(s_{y_{k-1}}) = y_{k-1}$;
— $(s_x, s_{y_{k-1}}, own_a) \in A_N$.

Таким образом, в обоих случаях выполняется условие $(e, \alpha) \in de\_facto\_rights_N(s_x)$, и траектория $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$ является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа. Значит, по определению 8 предикат $simple\_can\_share((e, \alpha), x, G_0)$ является истинным.

Обоснована достаточность выполнения условий теоремы для истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$ при $x \in N_U$.

Докажем необходимость выполнения условий теоремы для истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$ при $x \in N_U$. По определению 8 существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует субъект-сессия $s_x \in S_N$, такая, что $user_N(s_x) = x$ и выполняется условие $(e, \alpha) \in$ $\in de\_facto\_rights_N(s_x)$.

Среди всех траекторий выберем ту, у которой длина N является минимальной. Проведем доказательство индукцией по длине траекторий N.

Пусть $N = 0$, тогда существует недоверенная субъект-сессия $s_x \in N_S \cap S_0$, такая, что $user_0(s_x) = x$, и выполняется условие $(e, \alpha) \in de\_facto\_rights_0(s_x)$. Значит, существует роль $r \in R \cup AR$, такая, что $(e, \alpha) \in PA_0(r)$. Возможны два случая.

Первый случай: выполняется условие $r \in roles_0(s_x) \subset UA_0(x)$. Следовательно, условие 1 теоремы выполнено.

Второй случай: существует субъект-сессия $y \in S_0$, такая, что $r \in roles_0(y)$ и $(s_x, y, own_a) \in A_0$. Следовательно, по определению 3 истинен предикат $simple\_can\_access\_own(x, y, G_0)$. Таким образом, условие 2 теоремы выполнено.

Пусть $N = 1$, тогда из минимальности N следует, что выполняются условия $(e, \alpha) \notin de\_facto\_rights_0(s_x)$ и $(e, \alpha) \in de\_facto\_rights_1(s_x)$. Возможны пять случаев.

Первый случай: существует недоверенная субъект-сессия $y \in N_S \cap S_0$, такая, что либо $y = s_x$, либо $(s_x, y, own_a) \in A_0$, и существует роль $r_e \in UA_0(user_0(y))$, такая, что $(e, \alpha) \in PA_0(r_e)$ и $op_1 = take\_role(y, r_e)$. Если $y = s_x$, то выполнено условие 1 теоремы. Если $(s_x, y, own_a) \in A_0$, то по определению 3 предикат $simple\_can\_access\_own(x, y, G_0)$ является истинным. Следовательно, условие 2 теоремы выполнено.

Второй случай: существует субъект-сессия $y_1 \in S_0$, такая, что либо $y_1 = s_x$, либо $(s_x, y_1, own_a) \in A_0$, и существуют недоверенная субъект-сессия $x_2 \in N_S \cap S_0$ и роль $r_e \in roles_0(y_1)$, такие, что $((e, own_r), r_e) \in de\_facto\_actions_0(x_2)$, и $op_1 = grant\_right(x_2, r_e, (e, \alpha))$. Значит, существует субъект-сессия $y_2 \in S_0$, такая, что либо $y_2 = x_2$, либо $(x_2, y_2, own_a) \in A_0$, и выполняются условия $(e, own_r) \in PA_0(roles_0(y_2))$ и $r_e \in can\_manage\_rights(roles_0(y_2) \cap AR))$. Следова-

тельно, справедливо равенство $is\_simple\_bridge(x_2, y_1, y_2) = \text{true}$. По определению 5 выполняются условия $y_1 \in island(x)$ и $y_2 \in island(x_2)$. Положим $m = 2$. Следовательно, условие 3 теоремы выполнено.

Третий случай: существуют субъект-сессии $y, z \in S_0$, такие, что $z \in [y]$ и либо $s_x = z$, либо $(s_x, z, own_a) \in A_0$, и $(e, \alpha) \in PA_0(roles_0(y))$, $op_1 = control(s_x, y, z)$.

Четвертый случай: существует субъект-сессия $y \in S_0$, такая, что $(y, own_r) \in \in de\_facto\_rights_0(s_x)$, $(e, \alpha) \in PA_0(roles_0(y))$ и $op_1 = access\_own(s_x, y)$. Значит, существует субъект-сессия $z \in S_0$, такая, что $(s_x, z, own_a) \in A_0$ и $(y, own_r) \in \in PA_0(roles_0(z))$.

Пятый случай: существуют субъект-сессии $y, z \in S_0$, такие, что $\{(s_x, z, own_a), (z, y, own_a)\} \in A_0$, и $(e, \alpha) \in PA_0(roles_0(y))$, $op_1 = take\_access\_own(s_x, z, y)$.

В третьем, четвертом и пятом случаях по определению 5 выполняется условие $y \in island(x) \setminus \{x\}$. Следовательно, по утверждению 1 истинен предикат $simple\_can\_access\_own(x, y, G_0)$, и условие 2 теоремы выполнено.

Пусть $N > 1$ и утверждение теоремы верно для всех траекторий длины $l < N$. Докажем, что при длине траектории N если истинен предикат $simple\_can\_share((e, \alpha), x, G_0)$, то выполняются условия теоремы.

Из минимальности N следует, что существует недоверенная субъект-сессия $s_x \in N_S \cap S_{N-1}$, такая, что $user_{N-1}(s_x) = x$, и выполняются условия $(e, \alpha) \notin de\_facto\_rights_{N-1}(s_x)$ и $(e, \alpha) \in de\_facto\_rights_N(s_x)$. Возможны пять случаев.

Первый случай: существует недоверенная субъект-сессия $s_{y'} \in N_S \cap S_{N-1}$, такая, что либо $s_{y'} = s_x$, либо $(s_x, s_{y'}, own_a) \in A_{N-1}$, и существует роль $r_e \in UA_{N-1}(user_{N-1}(s_{y'}))$, такая, что $(e, \alpha) \in PA_{N-1}(r_e)$ и $op_N = take\_role(s_{y'}, r_e)$. Положим $y' = user_{N-1}(s_{y'})$, тогда по предположению 1 выполняется условие $r_e \in UA_0(y')$. Возможны две ситуации.

Первая ситуация: выполняется условие $(e, \alpha) \in PA_0(r_e)$. Если $s_{y'} = s_x$, то выполняется условие $(e, \alpha) \in PA_0(UA_0(x))$ и выполнено условие 1 теоремы. Если $(s_x, s_{y'}, own_a) \in A_{N-1}$, то положим $y = y'$. Тогда $(e, \alpha) \in PA_0(UA_0(y))$ и по определению 3 истинен предикат $simple\_can\_access\_own(x, y, G_0)$. Следовательно, выполнено условие 2 теоремы.

Вторая ситуация: выполняется условие $(e, \alpha) \notin PA_0(r_e)$. Если $s_{y'} = s_x$, то положим $k = 1$, $y_1 = s_{y'}$. Пусть $(s_x, s_{y'}, own_a) \in A_{N-1}$, тогда по определению 3 истинен предикат $simple\_can\_access\_own(x, y', G_0)$. Следовательно, по теореме 1 существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_k \in N_U \cup (N_S \cap S_0)$, субъект-сессий или недоверенных пользователей $y_1, \ldots, y_k \in N_U \cup S_0$, где $k \geqslant 1$, таких, что $x_1 = x$, $y_k = s_{y'}$, $y_i \in island(x_i)$, где $1 \leqslant i \leqslant m$, и выполняются условия:

— если $k \geqslant 2$, то справедливо равенство $is\_bridge(x_k, y_{k-1}, y_k) = \text{true}$;
— если $k \geqslant 3$, то для каждого $2 \leqslant i < k$ справедливо равенство или $is\_bridge(x_i, y_{i-1}, y_i) = \text{true}$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = \text{true}$.

При этом существуют $1 \leqslant M < N$ и недоверенная субъект-сессия $s_{x'} \in N_S \cap \cap S_{M-1}$, такие, что $((e, own_r), r_e) \in de\_facto\_actions_{M-1}(s_{x'})$ и $op_M = grant\_right(s_{x'}, r_e, (e, \alpha))$. Значит, существует субъект-сессия $s_{y''} \in S_{M-1}$, такая, что либо $s_{x'} = s_{y''}$, либо $(s_{x'}, s_{y''}, own_a) \in A_{M-1}$. При этом выполняются условия $r_e \in can\_manage\_rights(roles_{M-1}(s_{y''}) \cap AR))$ и $(e, own_r) \in PA_{M-1}(roles_{M-1}(s_{y''}))$. Положим $x' = = user_{M-1}(s_{x'}) \in N_U$. Если $s_{y''} \in N_S$, то положим $y_{k+1} = user_{M-1}(s_{y''}) \in N_U$. Тогда выполняется условие $r_e \in can\_manage\_rights(UA_0(y_{k+1}) \cap AR))$. Если $s_{y''} \in L_S$, то положим $y_{k+1} = s_{y''}$, по предположению 1 выполняются условия $y_{k+1} \in L_S \cap S_0$ и

$r_e \in can\_manage\_rights(roles_0(y_{k+1}) \cap AR)$). Значит, либо $x' = y_{k+1}$, либо истинен предикат $simple\_can\_access\_own(x', y_{k+1}, G_0)$, и по теореме 1 существует недоверенный пользователь $x_{k+1} \in N_U$, такой, что $y_{k+1} \in island(x_{k+1})$. Следовательно, по определению 6 справедливо равенство $is\_simple\_bridge(x_{k+1}, y_k, y_{k+1}) = \text{true}$. При этом так как $y_{k+1} \in island(x_{k+1})$, то возможен выбор такого x_{k+1}, что истинен предикат $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ с длиной траектории меньше N.

Значит, по предположению индукции для предиката $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ и из выполнения одного из условий

— $y_{k+1} \in N_U$ и $(e, own_r) \in PA_{M-1}(UA_{M-1}(y_{k+1}))$;
— $y_{k+1} \in L_S \cap S_0$ и $(e, own_r) \in PA_{M-1}(roles_{M-1}(y_{k+1}))$

следует, что для предиката $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ выполнено условие 1 или условие 3 теоремы.

Если для предиката $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ выполнено условие 1 теоремы, то $(e, own_r) \in PA_0(UA_0(x_{k+1}))$ и $y_{k+1} = x_{k+1}$. Положим $m = k + 1$.

Если для предиката $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ выполнено условие 3 теоремы, то существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_{k+1}, \ldots, x_m \in N_U \cup (N_S \cap S_0)$, субъект-сессий или недоверенных пользователей $y_{k+1}, \ldots, y_m \in N_U \cup S_0$, где $m \geqslant k + 1$, таких, что $y_i \in island(x_i)$, где $k + 1 \leqslant i \leqslant m$, и выполняется одно из условий:

— $y_m \in N_U$ и $(e, own_r) \in PA_0(UA_0(y_m))$;
— $y_m \in N_S \cap S_0$ и $(e, own_r) \in PA_0(UA_0(user_0(y_m)))$;
— $y_m \in L_S \cap S_0$ и $(e, own_r) \in PA_0(roles_0(y_m))$.

При этом справедливо равенство $is\_simple\_bridge(x_m, y_{m-1}, y_m) = \text{true}$, и для каждого $k + 2 \leqslant i \leqslant m$ справедливо равенство или $is\_bridge(x_i, y_{i-1}, y_i) = \text{true}$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = \text{true}$.

Таким образом, в первом случае во второй ситуации выполнено условие 3 теоремы.

Второй случай: существует субъект-сессия $s_{y'} \in S_{N-1}$, такая, что либо $s_{y'} = s_x$, либо $(s_x, s_{y'}, own_a) \in A_{N-1}$, и существуют недоверенная субъект-сессия $s_{x'} \in N_S \cap S_{N-1}$ и роль $r_e \in roles_{N-1}(s_{y'})$, такие, что $((e, own_r), r_e) \in de\_facto\_actions_{N-1}(s_{x'})$, и $op_N = grant\_right(s_{x'}, r_e, (e, \alpha))$. Значит, существует субъект-сессия $s_{y''} \in S_{N-1}$, такая, что либо $s_{y''} = s_{x'}$, либо $(s_{x'}, s_{y''}, own_a) \in A_{N-1}$, и выполняются условия $(e, own_r) \in PA_{N-1}(roles_{N-1}(s_{y''}))$, $r_e \in can\_manage\_rights(roles_{N-1}(s_{y''}) \cap AR)$).

Если $s_{y'} = s_x$, то положим $k = 1$, $y_1 = s_{y'}$. Пусть $(s_x, s_{y'}, own_a) \in A_{N-1}$, тогда по определению 3 истинен предикат $simple\_can\_access\_own(x, s_{y'}, G_0)$. Следовательно, по теореме 1 существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_k \in N_U \cup (N_S \cap S_0)$, субъект-сессий или недоверенных пользователей $y_1, \ldots, y_k \in N_U \cup S_0$, где $k \geqslant 1$, таких, что $x_1 = x$, $y_k = s_{y'}$, $y_i \in island(x_i)$, где $1 \leqslant i \leqslant m$, и выполняются условия:

— если $k \geqslant 2$, то справедливо равенство $is\_bridge(x_k, y_{k-1}, y_k) = \text{true}$;
— если $k \geqslant 3$, то для каждого $2 \leqslant i < k$ справедливо равенство или $is\_bridge(x_i, y_{i-1}, y_i) = \text{true}$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = \text{true}$.

Положим $x' = user_{N-1}(s_{x'}) \in N_U$. Если $s_{y''} \in N_S$, то положим $y_{k+1} = user_{N-1}(s_{y''}) \in N_U$. Тогда выполняется условие $r_e \in can\_manage\_rights(UA_0(y_{k+1}) \cap AR))$. Если $s_{y''} \in L_S$, то положим $y_{k+1} = s_{y''}$, по предположению 1 выполняются условия $y_{k+1} \in L_S \cap S_0$ и $r_e \in can\_manage\_rights(roles_0(y_{k+1}) \cap AR))$. Значит, либо $x' = y_{k+1}$,

либо истинен предикат $simple\_can\_access\_own(x', y_{k+1}, G_0)$, и по теореме 1 существует недоверенный пользователь $x_{k+1} \in N_U$, такой, что $y_{k+1} \in island(x_{k+1})$. Следовательно, по определению 6 справедливо равенство $is\_simple\_bridge(x_{k+1}, y_k, y_{k+1})$. При этом так как $y_{k+1} \in island(x_{k+1})$, то возможен выбор такого x_{k+1}, что истинен предикат $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ с длиной траектории меньше N.

Значит, по предположению индукции для предиката $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ и из выполнения одного из условий

— $y_{k+1} \in N_U$ и $(e, own_r) \in PA_{N-1}(UA_{N-1}(y_{k+1}))$;
— $y_{k+1} \in L_S \cap S_0$ и $(e, own_r) \in PA_{N-1}(roles_{N-1}(y_{k+1}))$

следует, что для предиката $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ выполнено условие 1 или условие 3 теоремы.

Если для предиката $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ выполнено условие 1 теоремы, то $(e, own_r) \in PA_0(UA_0(x_{k+1}))$ и $y_{k+1} = x_{k+1}$. Положим $m = k + 1$.

Если для предиката $simple\_can\_share((e, own_r), x_{k+1}, G_0)$ выполнено условие 3 теоремы, то существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_{k+1}, \ldots, x_m \in N_U \cup (N_S \cap S_0)$, субъект-сессий или недоверенных пользователей $y_{k+1}, \ldots, y_m \in N_U \cap S_0$, где $m \geqslant k + 1$, таких, что $y_i \in island(x_i)$, где $k + 1 \leqslant i \leqslant m$, и выполняется одно из условий:

— $y_m \in N_U$ и $(e, own_r) \in PA_0(UA_0(y_m))$;
— $y_m \in N_S \cap S_0$ и $(e, own_r) \in PA_0(UA_0(user_0(y_m)))$;
— $y_m \in L_S \cap S_0$ и $(e, own_r) \in PA_0(roles_0(y_m))$.

При этом справедливо равенство $is\_simple\_bridge(x_m, y_{m-1}, y_m) = \text{true}$ и для каждого $k + 2 \leqslant i \leqslant m$ справедливо равенство или $is\_bridge(x_i, y_{i-1}, y_i) = \text{true}$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = \text{true}$.

Таким образом, во втором случае выполнено условие 3 теоремы.

Третий случай: существуют субъект-сессии $s_{y'}, s_z \in S_{N-1}$, такие, что $s_z \in [s_{y'}]$ и либо $s_x = s_z$, либо $(s_x, s_z, own_a) \in A_{N-1}$, и $(e, \alpha) \in PA_{N-1}(roles_{N-1}(s_{y'}))$, $op_N = control(s_x, s_{y'}, s_z)$.

Четвертый случай: существует субъект-сессия $s_{y'} \in S_{N-1}$, такая, что $(s_{y'}, own_r) \in de\_facto\_rights_{N-1}(s_x)$, $(e, \alpha) \in PA_{N-1}(roles_{N-1}(s_{y'}))$ и $op_N = access\_own(s_x, s_{y'})$.

Пятый случай: существуют субъект-сессии $s_{y'}, s_z \in S_{N-1}$, такие, что $\{(s_x, s_z, own_a), (s_z, s_{y'}, own_a)\} \subset A_{N-1}$, и $(e, \alpha) \in PA_{N-1}(roles_0(s_{y'}))$, $op_N = take\_access\_own(s_x, s_z, s_{y'})$.

Таким образом, в третьем, четвертом и пятом случае существует роль $r_e \in UA_{N-1}(user_{N-1}(s_{y'}))$, такая, что $(e, \alpha) \in PA_{N-1}(r_e)$. При этом, если $s_{y'} \in N_S$, то положим $y' = user_{N-1}(s_{y'})$, если $s_{y'} \in L_S$, то $y' = s_{y'}$ и по предположению 1 $y' \in L_S \cap S_0$. Значит, истинен предикат $simple\_can\_access\_own(x, y', G_0)$, и выполнение условия 2 или 3 теоремы обосновывается аналогично первому случаю.

Следовательно, доказан шаг индукции при длине траектории, равной N. Доказательство необходимости выполнения условия теоремы для истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$ при $x \in N_U$ выполнено.

Теорема доказана. ■

Определим и обоснуем алгоритмически проверяемые необходимые и достаточные условия истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$ для случая, когда $x \in L_U$.

Теорема 3. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют доверенный пользователь $x \in L_U$ и право доступа

к сущности $(e, \alpha) \in P_0$. Предикат $simple\_can\_share((e, \alpha), x, G_0)$ является истинным тогда и только тогда, когда существует доверенная субъект-сессия $s_x \in L_S \cap S_0$, такая, что $user_0(s_x) = x$ и выполняется одно из следующих условий:

1. Выполняется условие $(e, \alpha) \in PA_0(roles_0(s_x))$.

2. Существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_m \in N_U \cup (N_S \cap S_0)$, субъект-сессий или недоверенных пользователей $y_1, \ldots, y_m \in N_U \cup S_0$, где $m \geqslant 1$, таких, что $y_i \in island(x_i)$, где $1 \leqslant i \leqslant m$, и выполняется одно из условий:

— $y_m \in N_U$ и $(e, own_r) \in PA_0(UA_0(y_m))$;
— $y_m \in N_S \cap S_0$ и $(e, own_r) \in PA_0(UA_0(user_0(y_m)))$;
— $y_m \in L_S \cap S_0$ и $(e, own_r) \in PA_0(roles_0(y_m))$.

При этом справедливо равенство $is\_simple\_bridge(x_1, s_x, y_1) = \text{true}$ и для каждого $2 \leqslant i \leqslant m$ справедливо равенство $is\_simple\_bridge(x_i, y_{i-1}, y_i) = \text{true}$.

Доказательство. Докажем достаточность выполнения условий теоремы для истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$ при $x \in L_U$.

Пусть выполнено условие 1 теоремы. Тогда по определению 8 предикат $simple\_can\_share((e, \alpha), x, G_0)$ является истинным.

Пусть выполнено условие 2 теоремы. Докажем истинность предиката $simple\_can\_share((e, \alpha), x, G_0)$ индукцией по длине m последовательностей субъект-сессий или недоверенных пользователей.

Пусть $m = 1$. Тогда по условию теоремы справедливо равенство $is\_simple\_bridge(x_1, s_x, y_1) = \text{true}$, и по определению 6 $y_1 \in island(x_1)$ и существует роль $r_e \in roles_0(s_x)$, такая, что выполняется одно из условий:

— $y_1 \in N_U$ и $(e, own_r) \in PA_0(UA_0(y_1))$ и $r_e \in can\_manage\_rights(AUA_0(y_1))$;
— $y_1 \in N_S \cap S_0$ и $(e, own_r) \in PA_0(UA_0(user_0(y_1)))$ и $r_e \in can\_manage\_rights(AUA_0(user_0(y_1)))$;
— $y_1 \in L_S \cap S_0$ и $(e, own_r) \in PA_0(roles_0(y_1))$ и $r_e \in can\_manage\_rights(roles_0(y_1) \cap \cap AR)$.

По утверждению следствия 1 существуют состояния $G_0, \ldots, G_M$ и правила преобразования состояний $op_0, \ldots, op_M$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_M} G_M$, где $M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует недоверенная субъект-сессия $s_{x_1} \in N_S \cap S_M$, такая, что либо $user_M(s_{x_1}) = x_1$, либо $s_{x_1} = x_1$, и в состоянии G_M выполняется условие $island\_actions(x_1) = island\_actions(s_{x_1}) \in de\_facto\_actions_M(s_{x_1})$. Значит, выполняется условие $((e, own_r), r_e) \in de\_facto\_actions_M(s_{x_1})$. Положим

$op_{M+1} = grant\_right(s_{x_1}, r_e, (e, \alpha))$.

Тогда в состоянии G_{M+1}, таком, что $G_M \vdash_{op_{M+1}} G_{M+1}$, выполняется условие $(e, \alpha) \in PA_{M+1}(r_e) \in PA_{M+1}(roles_{M+1}(s_x))$. Так как траектория $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_{M+1}} G_{M+1}$ является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, то по определению 8 является истинным предикат $simple\_can\_share((e, \alpha), x, G_0)$.

Заметим, что при $m = 1$ условие $s_x \in L_S \cap S_0$ не является существенным, т. е. передача прав доступа возможна в случае, когда $s_x \in N_S \cap S_0$.

Пусть $m > 1$ и утверждение верно для всех последовательностей длины $l < m$. Докажем достаточность условий теоремы при длине последовательности, равной m.

С учетом сделанного замечания и из предположения индукции следует, что существуют состояния $G_0, \ldots, G_M$ и правила преобразования состояний $op_0, \ldots, op_M$, такие,

что $G_0 \vdash_{op_1} \ldots \vdash_{op_M} G_M$, где $M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует субъект-сессия $s_{y_1} \in S_M$, такая, что либо $user_M(s_{y_1}) = y_1$, либо $s_{y_1} = y_1$, и в состоянии G_M выполняется условие $(e, \alpha) \in PA_M(roles_M(s_{y_1}))$. При этом по условию теоремы и по определению 6 справедливо равенство $is\_simple\_bridge(x_1, s_x, s_{y_1}) = \text{true}$. Выполняя рассуждения, аналогичные использованным при обосновании случая $m = 1$, получаем, что предикат $simple\_can\_share((e, \alpha), x, G_0)$ является истинным.

Таким образом, для $x \in L_U$ доказана достаточность выполнения условий теоремы для истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$.

Докажем для $x \in L_U$ необходимость выполнения условий теоремы для истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$.

Пусть истинен предикат $simple\_can\_share((e, \alpha), x, G_0)$. Тогда по определению 8 существуют состояния $G_1, \ldots, G_N = (PA_N, user_N, roles_N, A_N, F_N, H_{E_N})$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует субъект-сессия $s_x \in S_N$, такая, что $user_N(s_x) = x$ и выполняется условие $(e, \alpha) \in de\_facto\_rights_N(s_x)$. Так как $x \in L_U$, то по предположению 1 и по определениям 1 и 2 выполняются условия $s_x \in L_S \cap S_0$ и $(e, \alpha) \in PA_N(rolesN(s_x))$.

Среди всех траекторий выберем ту, у которой длина N является минимальной. Проведем доказательство индукцией по длине траекторий N.

Пусть $N = 0$, тогда $(e, \alpha) \in PA_0(roles_0(s_x))$. Следовательно, выполнено условие 1 теоремы.

Пусть $N = 1$, тогда из минимальности N следует, что $(e, \alpha) \notin PA_0(roles_0(s_x))$. Тогда существуют недоверенная субъект-сессия $x_1 \in N_S \cap S_0$ и роль $r_e \in roles_0(s_x)$, такие, что $((e, own_r), r_e) \in de\_facto\_actions_0(x_1)$, и $op_1 = grant\_right(x_1, r_e, (e, \alpha))$. Значит, существует субъект-сессия $y_1 \in S_0$, такая, что либо $y_1 = x_1$, либо $(x_1, y_1, own_a) \in A_0$, и выполняются условия $(e, own_r) \in PA_0(roles_0(y_1))$ и $r_e \in can\_manage\_rights(roles_0(y_1) \cap AR))$. Следовательно, справедливо равенство $is\_simple\_bridge(x_1, s_x, y_1) = \text{true}$. По определению 5 выполняется условие $y_1 \in island(x_1)$. Положим $m = 1$. Следовательно, условие 2 теоремы выполнено.

Пусть $N > 1$ и утверждение теоремы верно для всех траекторий длины $l < N$. Докажем, что при длине траектории N если истинен предикат $simple\_can\_share((e, \alpha), x, G_0)$, то выполняются условия теоремы.

Тогда существуют недоверенная субъект-сессия $s_{x'} \in N_S \cap S_{N-1}$ и роль $r_e \in roles_{N-1}(s_x)$, такие, что $((e, own_r), r_e) \in de\_facto\_actions_{N-1}(s_{x'})$, и $op_N = grant\_right(s_{x'}, r_e, (e, \alpha))$. Значит, существует субъект-сессия $s_{y'} \in S_{N-1}$, такая, что либо $s_{y'} = s_{x'}$, либо $(s_{x'}, s_{y'}, own_a) \in A_{N-1}$, и выполняются условия $(e, own_r) \in PA_{N-1}(roles_{N-1}(s_{y'}))$, $r_e \in can\_manage\_rights(roles_{N-1}(s_{y'}) \cap AR))$.

Положим $x' = user_{N-1}(s_{x'}) \in N_U$. Если $s_{y'} \in N_S$, то положим $y_1 = user_{N-1}(s_{y'}) \in N_U$. Тогда выполняется условие $r_e \in can\_manage\_rights(UA_0(y_1) \cap AR))$. Если $s_{y'} \in L_S$, то положим $y_1 = s_{y'}$, по предположению 1 выполняются условия $y_1 \in L_S \cap S_0$ и $r_e \in can\_manage\_rights(roles_0(y_1) \cap AR))$. Значит, либо $x' = y_1$, либо истинен предикат $simple\_can\_access\_own(x', y_1, G_0)$, и по теореме 1 существует недоверенный пользователь $x_1 \in N_U$, такой, что $y_1 \in island(x_1)$. Следовательно, по определению 6 справедливо равенство $is\_simple\_bridge(x_1, s_x, y_1)$.

Если выполняется одно из условий:

— $y_1 \in N_U$ и $(e, own_r) \in PA_0(UA_0(y_1))$;

— $y_1 \in N_S \cap S_0$ и $(e, own_r) \in PA_0(UA_0(user_0(y_1)))$;
— $y_1 \in L_S \cap S_0$ и $(e, own_r) \in PA_0(roles_0(y_1))$,

то положим $m = 1$, и условие 2 теоремы выполнено.

Пусть ни одно из данных условий не выполняется, тогда заметим, что доверенность субъект-сессии s_x не является существенной при получении принадлежащей ей ролью r_e права доступа (e, α). Следовательно, так как $(e, own_r) \in PA_{N-1}(roles_{N-1}(s_{y'}))$, где либо $y_1 \in N_U$ и $y_1 = user_{N-1}(s_{y'})$, либо $y_1 = s_{y'} \in L_S \cap S_0$, то, многократно повторяя выполненные рассуждения, получаем, что существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_2, \ldots, x_m \in N_U \cup (N_S \cap S_0)$, субъект-сессий или недоверенных пользователей $y_2, \ldots, y_m \in N_U \cup S_0$, где $m \geqslant 2$, таких, что $y_i \in island(x_i)$, где $2 \leqslant i \leqslant m$, и выполняется одно из условий:

— $y_m \in N_U$ и $(e, own_r) \in PA_0(UA_0(y_m))$;
— $y_m \in N_S \cap S_0$ и $(e, own_r) \in PA_0(UA_0(user_0(y_m)))$;
— $y_m \in L_S \cap S_0$ и $(e, own_r) \in PA_0(roles_0(y_m))$.

При этом справедливо равенство $is\_simple\_bridge(x_2, y_1, y_2) = \text{true}$, и для каждого $3 \leqslant i \leqslant m$ справедливо равенство $is\_simple\_bridge(x_i, y_{i-1}, y_i) = \text{true}$.

Таким образом, выполнено условие 2 теоремы. Следовательно, доказан шаг индукции при длине траектории, равной N.

Доказательство необходимости выполнения условия теоремы для истинности предиката $simple\_can\_share((e, \alpha), x, G_0)$ при $x \in L_U$ выполнено.

Теорема доказана. ■

3. Условия реализации информационных потоков по памяти

В рамках БР ДП-модели рассмотрим условия реализации информационных потоков по памяти с участием произвольного числа субъект-сессий для простых траекторий функционирования системы. Дадим определение.

Определение 9. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют сущности или недоверенные пользователи $x, y \in N_U \cup E_0$, где $x \neq y$. Определим предикат $simple\_can\_write\_memory(x, y, G_0)$, который будет истинным тогда и только тогда, когда существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и выполняется условие $(x', y', write_m) \in F_N$, где верно следующее:

— если $x \in E_0$, то $x' = x$; если $x \in N_U$, то $x' \in S_N$ и $user_N(x') = x$;
— если $y \in E_0$, то $y' = y$; если $y \in N_U$, то $y' \in S_N$ и $user_N(y') = y$.

Так как недоверенный пользователь может создать субъект-сессию, то в отличие от дискреционных или мандатных ДП-моделей в рамках БР ДП-модели он может являться источником или приемником информационного потока. С учетом условий функционирования существующих и перспективных КС будем считать, что в дальнейшем выполняется следующее предположение.

Предположение 2. У каждой доверенной субъект-сессии всегда имеется роль, обладающая правами доступа на чтение и запись к некоторой сущности. У каждого недоверенного пользователя имеется авторизованная роль, обладающая правами доступа на чтение и запись к некоторой сущности.

Обоснуем необходимые и достаточные условия истинности предиката $simple\_can\_write\_memory(x, y, G_0)$.

Теорема 4. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют сущности или недоверенные пользователи $x, y \in N_U \cup E_0$, где $x \neq y$. Предикат $simple\_can\_write\_memory(x, y, G_0)$ истинен тогда и только тогда, когда существует последовательность недоверенных пользователей или сущностей $e_1, \ldots, e_m \in N_U \cup E_0$, где $e_1 = x$, $e_m = y$ и $m \geqslant 2$, таких, что выполняется одно из условий.

1. $m = 2$ и $(x', y', write_m) \in F_0$, где выполняются условия:

— если $x \in E_0$, то $x' = x$; если $x \in N_U$, то $x' \in S_0$ и $user_0(x') = x$;
— если $y \in E_0$, то $y' = y$; если $y \in N_U$, то $y' \in S_0$ и $user_0(y') = y$.

2. Для каждого $i = 1, \ldots, m-1$ выполняется одно из условий:

— $e_i \in N_U \cup S_0$, $e_{i+1} \in N_U \cup E_0$ и $(e'_i, e'_{i+1}, write_m) \in F_0$, где верно следующее:
  - если $e_i \in S_0$, то $e'_i = e_i$; если $e_i \in N_U$, то $e'_i \in S_0$ и $user_0(e'_i) = e_i$;
  - если $e_{i+1} \in E_0$, то $e'_{i+1} = e_{i+1}$; если $e_{i+1} \in N_U$, то $e'_{i+1} \in S_0$ и $user_0(e'_{i+1}) = e_{i+1}$;
— $e_i \in N_U \cup S_0, e_{i+1} \in E_0 \setminus S_0$ и истинен предикат $simple\_can\_share((e_{i+1}, \alpha), e'_i, G_0)$, где $\alpha \in \{write_r, append_r\}$, и верно следующее:
  - если $e_i \in N_U$, то $e'_i = e_i$;
  - если $e_i \in S_0$, то $e'_i = user_0(e_i)$;
— $e_{i+1} \in N_U \cup S_0, e_i \in E_0 \setminus S_0$ и истинен предикат $simple\_can\_share((e_i, read_r), e'_{i+1}, G_0)$, где верно следующее:
  - если $e_{i+1} \in N_U$, то $e'_{i+1} = e_{i+1}$;
  - если $e_{i+1} \in S_0$, то $e'_{i+1} = user_0(e_{i+1})$;
— $e_i \in N_U \cup (N_S \cap S_0)$, $e_{i+1} \in N_U \cup S_0$ и истинен $simple\_can\_access\_own(e'_i, e_{i+1}, G_0)$, где верно следующее:
  - если $e_i \in N_U$, то $e'_i = e_i$;
  - если $e_i \in N_S \cap S_0$, то $e'_i = user_0(e_i)$;
— $e_{i+1} \in N_U \cup (N_S \cap S_0)$, $e_i \in N_U \cup S_0$ и истинен предикат $simple\_can\_access\_own(e'_{i+1}, e_i, G_0)$, где верно следующее:
  - если $e_{i+1} \in N_U$, то $e'_{i+1} = e_{i+1}$;
  - если $e_{i+1} \in N_S \cap S_0$, то $e'_{i+1} = user_0(e_{i+1})$.

Доказательство. Докажем достаточность выполнения условий теоремы для истинности предиката $simple\_can\_write\_memory(x, y, G_0)$.

Пусть выполнено условие 1 теоремы. Тогда по определению 9 предикат $simple\_can\_write\_memory(x, y, G_0)$ является истинным.

Пусть выполнено условие 2 теоремы. Тогда осуществим доказательство индукцией по длине m последовательности недоверенных пользователей или сущностей.

Пусть $m = 2$. Возможны пять случаев.

Первый случай: $x \in N_U \cup S_0$, $y \in N_U \cup E_0$ и $(x', y', write_m) \in F_0$, где выполняются условия:

— если $x \in S_0$, то $x' = x$; если $x \in N_U$, то $x' \in S_0$ и $user_0(x') = x$;
— если $y \in E_0$, то $y' = y$; если $y \in N_U$, то $y' \in S_0$ и $user_0(y') = y$.

Следовательно, по определению 9 предикат $simple\_can\_write\_memory(x, y, G_0)$ является истинным.

Второй случай: $x \in N_U \cup S_0, y \in E_0 \setminus S_0$ и истинен предикат $simple\_can\_share((y, \alpha), x', G_0)$, где $\alpha \in \{write_r, append_r\}$, и верно следующее:

— если $x \in N_U$, то $x' = x$;
— если $x \in S_0$, то $x' = user_0(x)$.

Тогда по определению 8 существуют состояния $G_1, \ldots, G_M$ и правила преобразования состояний $op_1, \ldots, op_M$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_M} G_M$, где $M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует субъект-сессия $s_x \in S_M$, такая, что $user_M(s_x) = x'$ и выполняется условие $(y, \alpha) \in de\_facto\_rights_M(s_x)$, где $\alpha \in \{write_r, append_r\}$. Если $\alpha = write_r$, то положим

$op_{M+1} = access\_write(s_x, y)$;

$N = M + 1$.

Если $\alpha = append_r$, то положим

$op_{M+1} = access\_append(s_x, y)$;

$N = M + 1$.

Таким образом, существует $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$ — простая траектория без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и в состоянии G_N выполняется условие $(s_x, y, write_m) \in F_N$, где $user_M(s_x) = x$, либо $s_x = x$. Следовательно, по определению 9 предикат $simple\_can\_write\_memory(x, y, G_0)$ является истинным.

В третьем случае: $y \in N_U \cup S_0$, $x \in E_0 \setminus S_0$ и истинен предикат $simple\_can\_share((x, read_r), y', G_0)$, где верно следующее:

— если $y \in N_U$, то $y' = y$;

— если $y \in S_0$, то $y' = user_0(y)$,

истинность предиката $simple\_can\_write\_memory(x, y, G_0)$ обосновывается аналогично второму случаю.

Четвертый случай: $x \in N_U \cup (N_S \cap S_0)$, $y \in N_U \cup S_0$ и истинен $simple\_can\_access\_own(x', y, G_0)$, где верно следующее:

— если $x \in N_U$, то $x' = x$;

— если $x \in N_S \cap S_0$, то $x' = user_0(x)$.

Тогда по определению 3 существуют состояния $G_1, \ldots, G_M$ и правила преобразования состояний $op_1, \ldots, op_M$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_M} G_M$, где $M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существуют субъект-сессии $s_x, s_y \in S_M$, такие, что $user_M(s_x) = x'$, или $s_y = y$, или $user_M(s_y) = y$ и выполняется условие $(s_x, s_y, own_a) \in A_M$.

Если $s_y \in L_S \cap S_M$, то по предположению 2 существуют роль $r \in roles_M(s_y)$ и сущность $e \in E_M$, такие, что $\{(e, read_r), (e, write_r)\} \subset PA_M(r)$. Положим

$op_{M+1} = access\_write(s_y, e)$;

$op_{M+2} = take\_flow(s_x, s_y)$;

$op_{M+3} = post(s_x, e, s_y)$;

$N = M + 3$.

Если $s_y \in N_S \cap S_M$, то по предположению 2 существуют роль $r \in UA_M(user_M(s_y))$ и сущность $e \in E_M$, такие, что $\{(e, read_r), (e, write_r)\} \subset PA_M(r)$. Положим

$op_{M+1} = take\_role(s_y, r)$;

$op_{M+2} = access\_write(s_y, e)$;

$op_{M+3} = take\_flow(s_x, s_y)$;

$op_{M+4} = post(s_x, e, s_y)$;

$N = M + 4$.

Таким образом, существует $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$ — простая траектория без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и в состоянии G_N выполняется условие $(s_x, s_y, write_m) \in F_N$, где верно следующее:

— или $s_x = x$, или $user_M(s_x) = x$;
— или $s_y = y$, или $user_N(s_y) = y$.

Следовательно, по определению 9 предикат $simple\_can\_write\_memory(x, y, G_0)$ является истинным.

В пятом случае: $y \in N_U \cup (N_S \cap S_0)$, $x \in N_U \cup S_0$ и истинен $simple\_can\_access\_own(y', x, G_0)$, где верно следующее:

— если $y \in N_U$, то $y' = y$;
— если $y \in N_S \cap S_0$, то $y' = user_0(y)$,

истинность предиката $simple\_can\_write\_memory(x, y, G_0)$ обосновывается аналогично четвертому случаю.

Пусть $m > 2$ и утверждение верно для всех последовательностей длины $l < m$. Докажем достаточность условий теоремы при длине последовательности, равной m.

По условию теоремы существует последовательность недоверенных пользователей или сущностей $e_1, \ldots, e_m \in N_U \cup E_0$, где $e_1 = x$, $e_m = y$, таких, что выполняется условие 2. Возможны четыре случая.

Первый случай: $x \in N_U \cup S_0$, $e_{m-1} \in N_U \cup S_0$. Тогда по предположению индукции истинны предикаты $simple\_can\_write\_memory(x, e_{m-1}, G_0)$ и $simple\_can\_write\_memory(e_{m-1}, y, G_0)$, и по определению 9 существуют состояния $G_1, \ldots, G_M$ и правила преобразования состояний $op_1, \ldots, op_M$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_M} G_M$, где $M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и выполняется условие $(x', e'_{m-1}, write_m) \in F_M$ и $(e'_{m-1}, y', write_m) \in F_M$, где выполняются условия:

— если $x \in S_0$, то $x' = x$; если $x \in N_U$, то $x' \in S_M$ и $user_M(x') = x$;
— если $e_{m-1} \in S_0$, то $e'_{m-1} = e_{m-1}$; если $e'_{m-1} \in N_U$, то $e'_{m-1} \in S_M$ и $user_M(e'_{m-1}) = e_{m-1}$;
— если $y \in E_0$, то $y' = y$; если $y \in N_U$, то $y' \in S_M$ и $user_M(y') = y$.

Положим

$op_{M+1} = find(x', e'_{m-1}, y')$;

$N = M + 1$.

Таким образом, существует $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$ — простая траектория без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и в состоянии G_N выполняется условие $(x', y', write_m) \in F_N$. Следовательно, по определению 9 предикат $simple\_can\_write\_memory(x, y, G_0)$ является истинным.

Второй случай: $x \in N_U \cup S_0$, $e_{m-1} \in E_0 \setminus S_0$. Тогда по предположению индукции истинен предикат $simple\_can\_write\_memory(x, e_{m-1}, G_0)$ и по определению 9 существуют состояния $G_1, \ldots, G_K$ и правила преобразования состояний $op_1, \ldots, op_K$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_K} G_K$, где $K \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и выполняется условие $(x', e'_{m-1}, write_m) \in F_K$, где, если $x \in S_0$, то $x' = x$; если $x \in N_U$, то $x' \in S_K$ и $user_K(x') = x$.

Кроме того, по условию теоремы $y \in N_U \cup S_0$ и выполняется одно из условий:

— $y \in S_0$ и $(e_{m-1}, read_r) \in PA_0(roles_0(y))$;
— $y \in N_U$ и истинен предикат $simple\_can\_share((e_{m-1}, read_r), y, G_0)$;
— $y \in N_S \cap S_0$ и истинен предикат $simple\_can\_share((e_{m-1}, read_r), user_0(y), G_0)$.

При выполнении первого условия положим $y' = y$ и $M = K$. При выполнении второго или третьего условия по определению 8 существуют состояния $G_{K+1}, \ldots, G_M$ и правила преобразования состояний $op_{K+1}, \ldots, op_M$, такие, что $G_K \vdash_{op_{K+1}} G_{K+1} \vdash_{op_{K+2}}$

$\ldots \vdash_{op_M} G_M$, где $M \geqslant K$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует субъект-сессия $y' \in S_M$, такая, что или $y' = y$, или $user_M(y') = y$, и выполняется условие $(e_{m-1}, read_r) \in de\_facto\_rights_M(y')$. Положим

$op_{M+1} = post(x', e'_{m-1}, y')$;

$N = M + 1$.

Таким образом, существует $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$ — простая траектория без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и в состоянии G_N выполняется условие $(x', y', write_m) \in F_N$. Следовательно, по определению 9 предикат $simple\_can\_write\_memory(x, y, G_0)$ является истинным.

Индуктивный шаг в третьем случае: $x \in E_0 \setminus S_0$, $e_{m-1} \in N_U \cup S_0$, обосновывается аналогично второму случаю.

Четвертый случай: $x, e_{m-1} \in E_0 \setminus S_0$. Тогда по условию теоремы $m \geqslant 4$ и $e_{m-2}, y \in \in N_U \cup S_0$. Далее индуктивный шаг обосновывается аналогично второму случаю.

Таким образом, доказана достаточность выполнения условий теоремы для истинности предиката $simple\_can\_write\_memory(x, y, G_0)$.

Докажем необходимость выполнения условий теоремы для истинности предиката $simple\_can\_write\_memory(x, y, G_0)$. По определению 9 существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и выполняется условие $(x', y', write_m) \in F_N$, где верно следующее:

— если $x \in E_0$, то $x' = x$; если $x \in N_U$, то $x' \in S_N$ и $user_N(x') = x$;
— если $y \in E_0$, то $y' = y$; если $y \in N_U$, то $y' \in S_N$ и $user_N(y') = y$.

Среди всех траекторий выберем ту, у которой длина N является минимальной. Проведем доказательство индукцией по длине траекторий N.

Пусть $N = 0$, тогда $(x', y', write_m) \in F_0$ и выполнено условие 1 теоремы.

Пусть $N = 1$, тогда из минимальности N следует, что $x', y' \in E_0$, $(x', y', write_m) \notin F_0$ и $(x', y', write_m) \in F_1$. Возможны семь случаев.

Первый случай: $y' \in S_0$ и $op_1 = access\_read(y', x)$, где $x' = x \in E_0 \setminus S_0$. Тогда $(x, read_r) \in de\_facto\_rights_0(y')$ и по определению 8 истинен предикат $simple\_can\_share((x, read_r), user_0(y'), G_0)$. Положим $m = 2$. Следовательно, условие 2 теоремы выполнено.

Во втором случае: $x' \in S_0$ и $op_1 = access\_write(x', y)$, и третьем случае: $x' \in S_0$ и $op_1 = access\_append(x', y)$, где $y' = y \in E_0 \setminus S_0$, выполнение условий теоремы обосновывается аналогично первому случаю.

Четвертый случай: $x' \in S_0$ и существует субъект-сессия $e_2 \in S_0$, такая, что $op_1 = find(x', e_2, y')$. Тогда $(x', e_2, write_m) \in F_0$, и или $(e_2, y', write_m) \in F_0$, или $y' = y \in E_0 \setminus S_0$, $(y, \beta) \in de\_facto\_rights_0(e_2)$, где $\beta \in \{write_r, append_r\}$. Значит, либо $(e_2, y', write_m) \in F_0$, либо $y' = y \in E_0 \setminus S_0$, $(y, \beta) \in de\_facto\_rights_0(e_2)$ и по определению 8 истинен предикат $simple\_can\_share((y, \beta), user_0(e_2), G_0)$. Положим $m = 3$. Следовательно, условие 2 теоремы выполнено.

В пятом случае: существует субъект-сессия $e_2 \in S_0$, такая, что $op_1 = pass(x', e_2, y')$, и шестом случае: $x', y' \in S_0$ и существует сущность $e_2 \in E_0$, такая, что $op_1 = post(x', e_2, y')$, выполнение условий теоремы обосновывается аналогично четвертому случаю.

Седьмой случай: $x' \in N_S \cap S_0$ и существует субъект-сессия $e_2 \in S_0$, такая, что $op_1 = take\_flow(x', e_2)$. Тогда $(x', e_2, own_a) \in A_0$ и $(e_2, y', write_m) \in F_0$. Следовательно, по определению 3 истинен предикат $simple\_can\_access\_own(user_0(x'), e_2, G_0)$. Положим $m = 3$. Следовательно, условие 2 теоремы выполнено.

Пусть $N > 1$ и утверждение теоремы верно для всех траекторий длины $l < N$. Докажем, что при длине траектории N если истинен предикат $simple\_can\_write\_memory(x, y, G_0)$, то выполняются условия теоремы.

Из минимальности N следует, что выполняется условие $(x', y', write_m) \notin F_{N-1}$. Возможны семь случаев.

Первый случай: $y' \in S_{N-1}$ и $op_N = access\_read(y', x)$, где $x' = x \in E_0 \setminus S_0$. Тогда $(x, read_r) \in de\_facto\_rights_{N-1}(y')$. Если $y' \in L_S \cap S_{N-1}$, то по предположению 1 выполняются условия $y = y' \in L_S \cap S_0$ и по определению 8 истинен предикат $simple\_can\_share((x, read_r), user_0(y), G_0)$. Если $y' \in N_S \cap S_{N-1}$, то либо $y = user_{N-1}(y')$ и по определению 8 истинен предикат $simple\_can\_share((x, read_r), y, G_0)$, либо $y = y' \in N_S \cap S_0$ и по определению 8 истинен предикат $simple\_can\_share((x, read_r), user_0(y), G_0)$. Положим $m = 2$. Следовательно, условие 2 теоремы выполнено.

Во втором случае: $x' \in S_{N-1}$ и $op_N = access\_write(x', y)$, и третьем случае: $x' \in S_{N-1}$ и $op_N = access\_append(x', y)$, где $y' = y \in E_0 \setminus S_0$, выполнение условий теоремы обосновывается аналогично первому случаю.

Четвертый случай: $x' \in S_{N-1}$ и существует субъект-сессия $s' \in S_{N-1}$, такая, что $op_N = find(x', s', y')$. Тогда $(x', s', write_m) \in F_{N-1}$, и или $(s', y', write_m) \in F_{N-1}$, или $y' = y \in E_0 \setminus S_0$, $(y, \beta) \in de\_facto\_rights_{N-1}(s')$, где $\beta \in \{write_r, append_r\}$.

Так как $(x', s', write_m) \in F_{N-1}$, то истинен предикат $simple\_can\_write\_memory(x, s, G_0)$ с длиной траектории меньше N, где либо $s = s' \in S_0$, либо $s = user_{N-1}(s') \in N_U$. По предположению индукции существует последовательность недоверенных пользователей или сущностей $e_1, \ldots, e_k \in N_U \cup E_0$, где $e_1 = x$, $e_k = s$ и $k \geqslant 2$, удовлетворяющих условию 1 или 2 теоремы.

Если $(s', y', write_m) \in F_{N-1}$, то аналогично существует последовательность недоверенных пользователей или сущностей $e_k, \ldots, e_m \in N_U \cup E_0$, где $e_k = s$, $e_m = y$ и $m - k \geqslant 1$, удовлетворяющих условию 1 или 2 теоремы, где либо $s = s' \in S_0$, либо $s = user_{N-1}(s') \in N_U$.

Если $y' = y \in E_0 \setminus S_0$, $(y, \beta) \in de\_facto\_rights_{N-1}(s')$, где $\beta \in \{write_r, append_r\}$, то по определению 8 истинен предикат $simple\_can\_share((y, \beta), s, G_0)$, где $s = user_{N-1}(s')$. Положим $m = k + 1$, $e_m = y$.

Таким образом, в четвертом случае существует последовательность недоверенных пользователей или сущностей $e_1, \ldots, e_m \in N_U \cup E_0$, где $e_1 = x$, $e_m = y$ и $m \geqslant 2$, удовлетворяющих условию 2 теоремы.

В пятом случае: существует субъект-сессия $s' \in S_{N-1}$, такая, что $op_N = pass(x', s', y')$, выполнение условий теоремы обосновывается аналогично четвертому случаю.

Шестой случай: $x', y' \in S_{N-1}$ и существует сущность $e \in E_{N-1}$, такая, что $op_N = post(x', e, y')$. Из минимальности N и предположения 2 следует, что $e \in E_0$. Значит, выполнение условий теоремы обосновывается аналогично четвертому случаю.

Седьмой случай: $x' \in N_S \cap S_{N-1}$ и существует субъект-сессия $s' \in S_{N-1}$, такая, что $op_N = take\_flow(x', s')$. Тогда $(x', s', own_a) \in A_{N-1}$ и $(s', y', write_m) \in F_{N-1}$. Если $s' \in S_0$, то положим $s = s'$; если $s' \notin S_0$, то положим $s = user_{N-1}(s') \in N_U$. Следовательно, либо $x = user_{N-1}(x')$ и по определению 3 истинен предикат $simple\_can\_access\_own(x, s, G_0)$, либо $x = x' \in N_S \cap S_0$ и по определению 3 исти-

нен предикат $simple\_can\_access\_own(user_0(x), s, G_0)$. Далее обоснование выполнения условий теоремы осуществляется аналогично четвертому случаю.

Значит, доказан шаг индукции при длине траектории, равной N. Доказательство необходимости выполнения условия теоремы для истинности предиката $simple\_can\_write\_memory(x, y, G_0)$ выполнено.

Теорема доказана. ■

Таким образом, в рамках БР ДП-модели для систем с простыми траекториями функционирования обоснованы необходимые и достаточные условия передачи прав доступа или реализации информационных потоков по памяти. В дальнейшем с применением техники доказательства теорем 1–4 планируется описать и обосновать условия передачи прав доступа, реализации информационных потоков по памяти и по времени для произвольных траекторий функционирования систем.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Базовая ролевая ДП-модель // Прикладная дискретная математика. 2008. № 1(1). С. 64–70.
2. *Девянин П. Н.* Анализ условий получения доступа владения в рамках базовой ролевой ДП-модели без информационных потоков по памяти // Прикладная дискретная математика. 2009. № 3(5). С. 69–84.

2010 Математические основы компьютерной безопасности №1(7)

DOI 10.17223/20710410/7/3

УДК 004.942

О ВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ ВРЕМЕННЫХ ЛОГИК И ВЕРИФИКАТОРОВ МОДЕЛЕЙ В ЗАДАЧЕ ОБНАРУЖЕНИЯ ОПАСНЫХ ОТКЛОНЕНИЙ СИСТЕМЫ

С. Е. Прокопьев

г. Москва

E-mail: prszs@mail.ru

В настоящей статье предложен вариант использования аппарата временных логик и верификаторов моделей в задаче контроля отклонений сложной информационной системы от «нормального» поведения, а также алгоритм слепого поиска закономерностей «черного ящика», представимых в рамках формализма временных логик.

Ключевые слова: *временные логики, верификаторы моделей, контроль отклонений системы, слепой поиск закономерностей.*

1. О проблемах использования временных логик и верификаторов моделей в анализе сложных систем

Временные логики являются мощным способом формального специфицирования свойств систем. Для этого исследуемая система представляется в виде одной из разновидностей автомата — т. н. модели Крипке $M{=}(S, S_0, R, L)$, где S — множество состояний системы; S_0 — множество начальных состояний системы; R — отношение переходов; L — функция разметки, сопоставляющая каждому состоянию некторый набор предикатов, истинных в данном состоянии.

Проблема верификации на модели формулируется следующим образом [1]. Пусть анализируемая система задана в виде модели Крипке M и есть формула временной логики f, специфицирующая некоторое свойство системы. Требуется найти в множестве S_0 подмножество всех состояний, в которых выполняется f, т. е. множество $\{s{\in}S_0 \mid M, s{\models}f\}$. Если указанному подмножеству принадлежат все начальные состояния системы, то модель M удовлетворяет свойству f.

В настоящее время существуют эффективные алгоритмы верификации на модели, например, в рамках логики ветвящегося времени (CTL) удается исследовать системы, имеющие число состояний порядка 10^{120}. Естественным является желание использовать этот мощный инструмент анализа последовательностей состояний в системах с бóльшим числом состояний, например, в информационных системах на базе современных операционных систем. Очевидно, что для сложных систем необходимо искать их упрощенные модели (абстракции), однако зачастую для абстракции, интересующей нас с точки зрения специфицирования свойств системы в рамках формализма временной логики, задачи нахождения корректного множества достижимых состояний, не превышающего порога применимости верификатора моделей, и корректного алгоритма переходов между ними оказываются нерешаемыми ввиду сложности анализируемой системы.

Возможным выходом может являться динамическое построение абстракции: в информационную систему (которая рассматривается как «серый ящик»: при необходимости мы можем заглянуть внутрь и посмотреть алгоритм работы системы в интересу-

ющем нас месте) внедряется набор датчиков, снимающих значения некоторых ее параметров в интересующие нас моменты. По полученным последовательностям показаний датчиков можно пытаться восстанавливать модель Крипке, соответствующую интересующей нас абстракции сложной системы, с последующим анализом заданных на этой абстракции свойств с использованием аппарата временных логик и верификаторов моделей. При этом чем сильнее (в допустимых пределах) будет комбинаторный взрыв корректных (т. е. достижимых в реальной системе) траекторий переходов в динамически построенной модели Крипке, тем эффективнее будет анализ свойств системы с использованием данного подхода — по сравнению с простой проверкой наблюдаемых траекторий на соответствие правилам, заданным в рамках отдельных траекторий, — так как в полной мере будут задействованы возможности современных алгоритмов верификации на модели. Тем не менее на практике для сложной информационной системы задача формулирования требуемых свойств последовательностей показаний датчиков, извлекаемых из «серого ящика», также может быть нерешаемой.

С учетом вышесказанного, имеет смысл подход, связанный с контролем отклонений абстракций от некоторого заданного «нормального» поведения. Под «нормальным» понимается поведение системы, наблюдавшееся в течение некоторого предварительного тестового периода — т. н. периода обучения. Если в ходе рабочей эксплуатации системы наблюдаются отклонения от зафиксированного «нормального» поведения, то их можно рассматривать как нарушения свойств безопасности системы. Определение «нормального» поведения системы предполагает обнаружение — желательно автоматическое — набора отдельных «нормальных» закономерностей ее поведения.

В настоящее время основным инструментом поиска закономерностей поведения системы на этапе обучения и контроля сохранения этих закономерностей на этапе ее рабочей эксплуатации являются нейронные сети. Однако с учетом того, что многие важные закономерности функционирования системы могут быть описаны в терминах временных логик, а также с учетом эффективности современных алгоритмов верификации на модели, для решения задачи контроля отклонений предлагается использовать аппарат временных логик и верификаторов моделей. Для этого предлагается осуществлять полуавтоматический подбор (с использованием верификаторов моделей и частичных знаний об алгоритме работы «серого ящика») формул временной логики, выполнимых на модели системы, построенной на этапе обучения по снимаемым последовательностям значений датчиков. Настоящая статья носит постановочный характер, рассматривая некоторые возможные подходы к построению системы обнаружения отклонений на базе временных логик.

2. Общее описание системы обнаружения отклонений

Первичными элементами системы обнаружения отклонений (СОО) являются датчики $(D_0, ..., D_{m-1})$, внедренные в контролируемую систему для регистрации происходящих в системе событий, замера значений интересующих параметров системы и т. д. СОО, в соответствии с некоторым алгоритмом (через заданные интервалы времени, при наступлении определенных событий и т. д.), снимает показания датчиков и вычисляет текущее значение вектора предикатов $\bar{P}=(P_0, ..., P_i, ..., P_{n-1})$, где $P_i=P(D_0^{Pi}, ..., D_{k(Pi)-1}^{Pi})$ и $k(Pi) < m$ — число датчиков, от которых зависит предикат P. В простейшем случае предикат имеет вид «датчик D_j сработал», в более сложном — зависит от нескольких датчиков и накладывает на их значения некоторое условие.

В рамках задачи построения СОО необходимо расставить в системе датчики и определить предикаты так, чтобы по снимаемым обучающим последовательностям

векторов предикатов можно было построить модель Крипке, соответствующую интересующей нас абстракции контролируемой системы. Неформально, под абстракцией системы понимается скоррелированный с ней (через датчики) автомат. Наиболее интересны абстракции с меньшим, чем у реальной системы, числом состояний и выраженными в более компактном виде, чем в реальной системе, правилами переходов.

Исходными данными для СОО являются последовательности упорядоченных по времени векторов значений предикатов $\{(\bar{p}_{t_0}, ..., \bar{p}_{t_i}, ..., \bar{p}_{t_{\text{cur}}})\}$, где $\bar{p}_{t_i}=(p^0_{t_i}, ..., p^{n-1}_{t_i})$, $p^j_{t_i}$ — значение предиката P_j в момент времени t_i, t_0 — время первого, а t_{cur} — последнего замера значений датчиков в данной последовательности. Каждая последовательность соответствует одной траектории вычислений, прошедшей через абстракцию.

На этапе обучения на основе траекторий, извлекаемых из последовательностей векторов предикатов, восстанавливается модель Крипке, соответствующая данной абстракции. Далее перебором проверяется выполнимость на ней различных формул временных логик с использованием современных алгоритмов верификации на модели (например, описанных в [1]), а также частичного знания алгоритма работы «серого ящика». Формулы, выполнимые на построенной модели, являются контролирующими формулами СОО и используются для проверки сохранения выявленных темпорально-логических закономерностей на множестве последовательностей векторов значений предикатов, наблюдаемых в ходе рабочей эксплуатации системы.

3. Алгоритм слепого поиска закономерностей, представимых в рамках формализма временных логик

В качестве одного из вариантов поиска абстракции и построения ее модели Крипке предлагается описанный ниже алгоритм.

В основу данного алгоритма положены следующие интуитивные рассуждения. Как было указано выше, под абстракцией системы понимается скоррелированный с этой системой автомат. Таким образом, для того чтобы найти абстракцию, необходимо из снимаемых с реальной системы последовательностей векторов значений предикатов извлечь информацию о ее состояниях и правилах переходов. Предположим, что часть координат обучающих векторов несут информацию о текущих состояниях (текущих вершинах графа переходов), а часть из оставшихся — о значениях переменных разметки этих состояний. Тогда можно использовать следующий подход к поиску абстракции и построению ее модели Крипке. Сначала разобъем каждый обучающий вектор на два подвектора: *подвектор состояний* (составлен из координат состояний) и *подвектор разметки* (составлен из оставшихся координат). Затем из последовательностей пар подвекторов состояний и разметки строим лес деревьев, состояния которого соответствуют подвекторам состояний и размечены множествами подвекторов разметки. Далее, для каждого множества подвекторов разметки каждого состояния находим координаты, в которых эти подвекторы отличаются, и вычеркиваем их глобально из разметки всего леса (интуитивно, это означает, что эти предикаты не скоррелированы с данной абстракцией). Если после вычеркивания подвекторы разметки не оказались пустыми, то склеиваем совпадающие поддеревья и замыкаем конечные состояния сами в себя. Получаем модель Крипке, в которой множеством начальных состояний будет множество корней каждого дерева в исходном лесе. Описание данного алгоритма в более формальном виде будет представлено ниже.

Определение 1. Пусть выбран критерий C, определяющий, подходит ли множество двоичных векторов для описания состояний абстракции (например, использующий описанные выше стратегии). Будем говорить, что множество двоичных векторов

образует группу состояний относительно критерия C, если критерий C выполняется на этом множестве. *Правилом прореживания* критерия выявления группы состояний C будем называть способ выбора подмножества множества, образующего группу состояний относительно критерия C.

Пример 1. Определим критерий C_{example} следующим образом: множество двоичных векторов удовлетворяет критерию C_{example}, если каждый вектор имеет вес 0 или 1. Определим правило прореживания критерия C_{example} как исключение из этого множества всех нулевых векторов.

Можно предложить следующие стратегии определения критериев группы состояний (могут применяться одновременно): 1) искать множества подвекторов с наибольшим расстоянием Хэмминга (интуитивно, чем в большем числе предикатов отражается состояние абстракции, тем сильнее подвекторы состояний будут отличаться друг от друга); 2) искать множества подвекторов минимальной мощности (интуитивно, чем больше совпадений подвекторов состояний, тем больше вероятность того, что мы нашли абстракцию); 3) по количеству совпадений поддеревьев (аналогично предыдущему).

Пусть задан критерий выявления группы состояний C с правилом прореживания R, а также алгоритм генерации A конечного множества темпорально-временных формул по конечному набору предикатов. Пусть имеется множество $W = \{V(k) : k = 1, \ldots, K\}$ последовательностей разной длины, составленных из двоичных векторов длины n. Предлагается следующий алгоритм слепого поиска абстракций и темпорально-логических закономерностей в них:

1. Для каждого $i = 1, \ldots, n-1$, для каждого подмножества $Pos_j^{(i,n)}$ мощности i множества всех координат $\{1, \ldots, n\}$, $j = 1, \ldots, C_n^i$, выполняем шаги 2–11.
2. Для множества векторов $W_{Pos_j^{(i,n)}}$, полученного ограничением векторов множества W на координатах $Pos_j^{(i,n)}$, проверяем, образует ли оно группу состояний относительно критерия C. Если нет, то переходим на следующую итерацию шага 3.
3. Множество $ST = Pos_j^{(i,n)}$ будем называть *текущим множеством координат состояния*, а множество LB оставшихся координат — *текущим множеством координат разметки*. Построим из множества W множество $WW = \{(V_{ST}(k), V_{LB}(k)) : k = 1, \ldots, K\}$ пар последовательностей подвекторов, полученных ограничением последовательностей векторов $V(k)$ на множествах координат ST и LB.
4. Из множества пар последовательностей WW получаем множество пар последовательностей $WW' = \{(V'_{ST}(k), V'_{LB}(k)) : k = 1, \ldots, K\}$ путем применения для каждого k правила прореживания R к последовательности $V_{ST}(k)$ и удаления из последовательности $V_{LB}(k)$ подвекторов с теми же порядковыми номерами, что и у подвекторов, удаленных из $V_{ST}(k)$.
5. Множество пар последовательностей WW' будем рассматривать как множество последовательностей пар $WW' = \left\{V_{(ST,LB)}(k) = \{(v_{ST}(k,l), v_{LB}(k,l))\}_l : k = 1, \ldots, K\right\}$. Вектор $v_{ST}(k,l)$ будем называть левой половиной элемента l последовательности $V_{(ST,LB)}(k)$, а $v_{LB}(k,l)$ — правой. Из множества WW' индуктивно по $k = 1, \ldots, K$ строим лес F_K, каждое состояние которого размечено парой $(v_{ST}, \{v_{LB}\})$, где v_{ST} — подвектор состояния (назовем его левой половиной разметки вершины леса), а $\{v_{LB}\}$ — множество подвекторов разметки (назовем его правой половиной разметки вершины леса), следующим образом:
   а) Число вершин F_1 равно длине последовательности $V_{(ST,LB)}(1)$. Каждой вершине F_1 взаимно однозначно соответствует элемент последовательности

$V_{(ST,LB)}(1)$, который является ее разметкой. В графе F_1 вершины w' и w'' с разметками (v'_{ST}, v'_{LB}) и (v''_{ST}, v''_{LB}) соединяются направленным ребром от w' к w'', если (v''_{ST}, v''_{LB}) следует сразу за (v'_{ST}, v'_{LB}) в последовательности $V_{(ST,LB)}(1)$. Получаем лес F_1, состоящий из одного дерева, состоящего из одной ветви.

б) Лес F_k строится из F_{k-1} следующим образом. Если в лесе F_{k-1} нет дерева с корнем, левая половина разметки которого совпадает с левой половиной первого элемента $V_{(ST,LB)}(k)$, то F_k будет состоять из F_{k-1} и дерева (из одной ветви), построенного из $V_{(ST,LB)}(k)$ способом, описанным выше для $k = 1$. Если такое дерево есть, то идем из его корня в соответствии с направлениями ребер, последовательно извлекая по одному элементу из $V_{(ST,LB)}(k)$ и сравнивая левые половины разметки преемников текущей вершины дерева с левой половиной текущего элемента $V_{(ST,LB)}(k)$. Пусть w — текущая вершина дерева, а (v_{ST}, v_{LB}) — текущий элемент последовательности $V_{(ST,LB)}(k)$. Если у w есть вершина-преемник, левая половина разметки которой равна v_{ST}, то добавляем v_{LB} в его правую половину (в множество его подвекторов разметки). Если таких преемников у w нет, создаем новую вершину, размеченную парой (v_{ST}, v_{LB}), и направляем в нее из w ребро.

6. Избавляемся от множественной разметки: для каждого состояния леса F_K находим в векторах из его правой половины разметки координаты, в которых эти векторы отличаются, и вычеркиваем эти координаты из множества LB и векторов разметки всех состояний дерева. Если после вычеркивания всех конфликтующих координат LB стало пустым, переходим на следующую итерацию шага 3. Иначе имеем разметку каждого состояния, состоящую ровно из одного вектора.
7. Запоминаем S_0 — текущее множество вершин леса F_K.
8. Преобразуем лес F_K в граф G путем склеивания совпадающих поддеревьев в F_K и замыкания конечных состояний F_K самих в себя.
9. Обозначим через S множество состояний графа G, определим отношение переходов R в соответствии с переходами графа G, определим функцию разметки L как ограничение разметки графа G на подвекторах разметки (т. е. удалим подвекторы состояний). Получим модель Крипке $M = (S, S_0, R, L)$.
10. Используя алгоритм A, генерируем конечный набор формул TF, определенных на множестве предикатов, соответствующих оставшимся координатам текущих подвекторов разметки. Перебираем для модели M все формулы из множества TF. Если никакая формула из TF не выполнима на M, переходим на следующую итерацию шага 3, иначе получаем множество выполнимых формул $TF_{\text{sat}} \subset TF$.
11. Считаем, что темпорально-временная закономерность найдена, если $TF_{\text{sat}} \neq \varnothing$.

В конечном итоге, если мы смогли построить достаточно компактную модель Крипке, в которую «уложилось» множество обучающих последовательностей, и нашли некоторый набор выполнимых на ней формул, то можно предполагать, что мы узнали о существовании и получили (возможно, неполное) описание абстракции, скоррелированной с системой через построенный граф. При этом подбор темпорально-логических формул является способом поиска по этому графу закономерностей абстракции.

Описанный выше алгоритм является крайним вариантом, рассматривающим систему как «черный ящик». Очевидно, что слепой поиск закономерностей, представимых в виде формул временной логики, имеет неподъемную трудоемкость для нетривиальных абстракций и нетривиальных формул даже с учетом эффективности современных алгоритмов на графах и алгоритмов верификации на модели. На практике выбор датчиков, предикатов состояний, предикатов разметки, множества проверяемых формул

должен происходить с учетом частичного знания алгоритма работы системы («серого ящика») и предположений о возможных закономерностях системы. В простейшем виде предикаты состояний могут иметь вид «произошло событие $event$» или «была вызвана функция $func$». Тогда при поиске абстракции можно использовать критерий образования группы состояний и правило прореживания из примера 1.

Алгоритм построения графа можно дополнить эвристическим поиском циклов перед склеиванием совпадающих поддеревьев на шаге 3, а также склеиванием «похожих» поддеревьев согласно некоторому критерию. При этом будет происходить потеря корректности построенного графа относительно абстракции, но зато может сильно сократиться его объем. Заметим, что допустимыми будут некорректные сокращения модели, инвариантные относительно выполнимости на ней формул, описывающих найденные закономерности.

ЛИТЕРАТУРА

1. *Кларк Э.М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking: пер. с англ. / под ред. Р. Смелянского. М.: МЦНМО, 2002. 416 с.

DOI 10.17223/20710410/7/4
УДК 004.432.2

ВНЕДРЕНИЕ ПОЛИТИК БЕЗОПАСНОСТИ В КОМПЬЮТЕРНЫЕ СИСТЕМЫ МЕТОДОМ АОП НА ПРИМЕРЕ FTP-СЕРВЕРА APACHE

Д. А. Стефанцов, А. Е. Филимонов

Томский государственный университет, г. Томск, Россия

E-mail: dastephantsov@mail.tsu.ru, filimonov1987@gmail.com

Аспектно-ориентированное программирование (АОП) является одной из важнейших парадигм программирования в области написания политик безопасности для компьютерных систем (КС). С его помощью удаётся модифицировать и встраивать политики безопасности в защищаемую КС без модификации её исходного кода. В данной статье авторы на основе собственного опыта работы в этом направлении формулируют основные требования и некоторые рекомендации к реализации и успешному внедрению политик безопасности в защищаемые КС методом АОП. По этим рекомендациям осуществлено внедрение базовой политики ролевого разграничения доступа в FTP-сервер Apache. Технология этого внедрения подробно изложена в статье.

Ключевые слова: *компьютерные системы, политика безопасности, аспектно-ориентированное программирование, Apache Ftp Server.*

Введение

Реализации политик безопасности (ПБ) можно встретить во многих компьютерных системах — от операционных систем общего назначения [1] до узкоспециализированных Web-систем хранения данных.

Несмотря на годы практики, большинство методов реализации ПБ являются грубым нарушением принципа «разделения аспектов» изучаемой области [2]. Процесс реализации защищённого приложения традиционными методами предполагает как совместное проектирование ПБ и основной части приложения, так и их совместную реализацию. Если впоследствии появится необходимость в изменении политики безопасности, то внедрение этих изменений будет осложнено необходимостью анализа, повторного проектирования и реализации всех точек пересечения модулей ПБ и главной части приложения.

Отметим, что изменение ПБ зачастую является единственной адекватной мерой противодействия новым видам атак и/или угроз, и, несмотря на это, политики безопасности современных компьютерных систем остаются неизменными. Вероятно, причиной тому служит излишняя сложность внесения изменений.

Некоторые исследователи способов проектирования программного обеспечения (ПО) сходятся во мнении, что наличие данных препятствий к модификации программы является следствием ограничений, накладываемых современными средствами реализации, в частности существующими парадигмами программирования [3, 4]. В этих же источниках указывается один из возможных путей решения этой проблемы — аспектно-ориентированное программирование (АОП).

Согласно принципу модульности АОП, вся функциональность политики безопасности должна располагаться в отдельных модулях — аспектах. Важной особенностью

данного принципа является возможность разработки ПБ отдельно от основной функциональности программы, без изменения исходных кодов последней. Один из путей осуществления такой возможности описан в [5]. На этом пути для успешного присоединения аспекта политики безопасности к уже существующей КС необходимо отдельно от КС реализовать в виде модуля основную функциональность ПБ и соединительный модуль, который бы непосредственно связывал между собой модуль КС и модуль ПБ. Именно присоединение ПБ к КС при помощи соединительного модуля без модификации исходного кода КС и понимается здесь под внедрением ПБ в защищаемую КС.

В настоящей статье авторы на основе собственного опыта работы в этом направлении формулируют в разд. 2 и 4 некоторые рекомендации к реализации и успешному присоединению ПБ к защищаемым КС методом АОП. По этим рекомендациям осуществлено внедрение базовой политики ролевого разграничения доступа (РРД) [6] в FTP-сервер Apache. Технология этого внедрения продемонстрирована в разд. 6 и 7, где описываются соответственно реализация основной функциональности РРД — пакет RBAC, содержащий реализации на языке Java основных объектов политики РРД, таких, как Пользователь (User.java), Роль (Role.java) и т. д., и реализация пакета RBAC-AJ — соединительного модуля между сервером и пакетом RBAC и инициализация исходных данных политики РРД сервера — файл MainConf.xml. Необходимые понятия из АОП и сведения о FTP-сервере Apache приводятся в разд. 3 и 5 соответственно. Изложение начинается с краткой характеристики компьютерных систем, защита которых возможна подобным образом.

1. Особенности защищаемых компьютерных систем

АОП является удобным и перспективным инструментом для разработки и внедрения политик безопасности в КС. Однако его использование накладывает ряд ограничений на защищаемую компьютерную систему, а также на модуль ПБ. Первое и самое важное ограничение состоит в доступности исходных кодов защищаемой КС, что позволяет детально изучить заложенную в ней функциональность и определить необходимые срезы точек выполнения, которые будут использоваться для соединения компьютерной системы с политикой безопасности. В эпоху развития свободного программного обеспечения (СПО) данное ограничение не является препятствием для внедрения политики безопасности и позволяет использовать все преимущества СПО.

Поскольку методом внедрения политики выбрано аспектно-ориентированное программирование, то данный способ применим только к тем программным продуктам, аспектные расширения языка которых имеют законченный вид и библиотеки функций в которых полностью отлажены. Например, аспектное расширение языка Java [7] — AspectJ является родоначальником АОП, его библиотеки хорошо отлажены и находятся в свободном доступе [8]. Однако этого нельзя сказать про PHPAspect [9].

2. Рекомендации по разработке и реализации политики безопасности

В настоящее время трудно представить себе приложение, которое не использовало бы механизм передачи данных по сети. Следовательно, необходимо протестировать приложение на возможные ошибки при многопоточном использовании функционала политики безопасности. Некоторые языки программрования предоставляют встроенную возможность синхронизации доступа к ячейкам памяти и критическим участкам кода [1].

Одной из рекомендаций к реализации политики безопасности является наличие двух уровней реализации. Первый — это абстрактная модель, которая представляет

собой реализацию ПБ со всем ее содержимым в соответствии с необходимой для этого теорией и ассоциациями между элементами политики безопасности. Например, ассоциация между элементами Пользователь и Роль базовой политики РРД [6] имеет следующий вид: пользователь может быть авторизован на одну или большее число ролей, и на некоторую роль может быть авторизовано как несколько пользователей, так и ни одного. Для удобства программирования можно построить UML-диаграммы [10] в соответствии с теорией.

На практике для этого требования подходят механизм наследования и наличие абстрактных классов объектно-ориентированного программирования (ООП).

Вторым уровнем должно стать доопределение данной модели в соответствии с защищаемой компьютерной системой. Иначе говоря, необходимо унаследовать классы, описывающие объекты защищаемой компьютерной системы, от соответствующих классов абстрактной модели. Тем самым достигается свойство универсальности реализации политики безопасности. В случае написания политики для другой компьютерной системы абстрактная модель сохранит свою функциональность, и отпадет необходимость переписывать весь код политики, что является ценным качеством программного продукта в связи со всё ужесточающейся борьбой между нападением на компьютерные системы и ее защитой.

Следующим пожеланием к реализации ПБ является возможность гибкого конфигурирования данной политики. Наилучшим для этого способом является вынесение всех важных константных значений в некоторый конфигурационный файл. На данное время общепризнанным стандартом формирования таких файлов является формат XML [11]. Для многих языков существуют библиотеки, представляющие собой XML-парсеры, то есть синтаксические анализаторы конфигурационных файлов. Они позволяют представить содержащуюся в файле информацию в виде, понятном обработчикам этой информации. Следовательно, необходимо таким образом определить расположение информации в тегах XML, чтобы оно содержало всю необходимую конфигурационную информацию и удовлетворяло спецификациям формата XML и чтобы существовала возможность удобного добавления/удаления записей из файла.

Существуют две принципиально различные модели разбора XML-документов — SAX (Simple Api for Xml) и DOM (Document Object Model). Кратко отметим основные особенности этих двух подходов.

DOM позволяет представить данные, заключенные в XML-файле, в виде иерархической (древовидной) объектной структуры. DOM создает дерево с узлами на основе информации, заключенной в файле. В дальнейшем для доступа к информации необходимо производить операции с этим деревом. Очевидно, что данная модель разбора XML-файла ресурсоемкая в смысле расхода памяти для хранения данного дерева.

SAX позволяет представить процесс разбора XML-файла в виде череды событий, генерируемых функцией разбора, использующей SAX. Эти события генерируются в процессе обхода данного файла разборщиком — например, при обнаружении открытия тега, его закрытия, а также некоторой информации, заключенной между этими тегами и т. п. В дальнейшем необходимо написать программы-обработчики данных событий.

Для разбора XML-файлов при реализации политики безопасности рекомендуется использовать SAX-подход, так как содержимое файла конфигурации, который будет подвергаться разбору, обычно хорошо укладывается в объектную модель (например, Конфигурация — Объект конфигурации).

3. Необходимые понятия из АОП

Точка выполнения программы (join point) — любая команда в любом алгоритме объектной системы.

Описание точек выполнения программы (pointcut) — конструкция языка программирования, задающая произвольное число точек выполнения программы. С помощью логических связок «и», «или», «не» можно составить композицию описаний точек выполнения программы, в свою очередь являющуюся описанием точек выполнения программы. Описание точек выполнения программы называется также *срезом точек выполнения программы.*

След описания точек выполнения программы (join point shadow) — это все точки выполнения программы, в данный момент подходящие под описание точек выполнения программы.

Предписание (advice) — алгоритм, который сопоставляется описанию точек выполнения программы и который выполняется всякий раз, когда выполнение программы достигает любой точки из следа описания.

Аспект (aspect) — модуль или совокупность модулей программирования, которые реализуют тот или иной аспект приложения.

Для взаимодействия политики безопасности и КС могут быть использованы два очень полезных свойства АОП — рефлексия и интроспекция.

Рефлексия — это способность программы менять собственную структуру или поведение.

Интроспекция — это способность программы определять свойства объектов во время её выполнения.

Метод интроспекции является весьма полезным способом реализации АОП, так как с его с помощью можно отличать выполнение аспекта от объекта к объекту, узнавать свойства текущего объекта.

4. Рекомендации по разработке и реализации соединительного модуля

На данном этапе подразумевается наличие готовой политики безопасности, существующей отдельно от главного процесса защищаемой КС. Для её присоединения к КС требуется:

1) определить моменты логики работы защищаемой КС, при которых необходимо использование функционала политики безопасности. В данном пункте предстоит серьезная работа по изучению исходного кода компьютерной системы и построение UML-диаграмм полезных классов;
2) распределить указанные моменты по группам для последующего совместного описания на языке АОП;
3) выделить точки выполнения защищаемой компьютерной системы, соответствующие п. 1 и 2;
4) описать срезы точек выполнения на аспектном расширении того языка, на котором написана защищаемая компьютерная система;
5) запрограммировать предписания на описанные срезы точек выполнения.

Процесс функционирования политики безопасности предусматривает непосредственный обмен информацией между КС и политикой, поэтому требуется передача данных от пользователя к модулю политики в обход основной функциональности программы, так как модификация исходных кодов не предполагается в силу принятия принципа разделения аспектов. Так, например, при внедрении ПБ в некоторый сервер,

который имеет возможность пользовательского входа, можно перехватывать данные пользователя при аутентификации в КС этим методом.

Для этого существует один из видов предписаний — around. Он позволяет своему программному тексту быть исполненным вместо некоторого среза точек выполнения. Поэтому очень важно правильно определить точки выполнения и сгруппировать их.

Еще одним полезным свойством данного предписания является его использование в моментах связи программной реализации защищаемой КС и политики безопасности. В случае успешного прохождения проверок политики безопасности ход выполнения может перейти дальше на прерванную команду. В случае же отрицательного ответа от политики возможно прерывание работы компьютерной системы и выброс исключения.

В качестве иллюстрации процесса внедрения политики безопасности выберем в качестве защищаемой КС файловый сервер Apache Ftp Server.

5. Необходимые сведения об Apache Ftp Server

Данный FTP-сервер разработан компанией Apache Software Foundation для свободного использования [12]. Компания Apache позиционирует свою разработку как приложение, обладающее, помимо прочих, следующими особенностями, важными для выполняемой работы:

— сервер полностью разработан на Java;
— виртуальная домашняя директория для каждого пользователя, запрос на возможность записи в директорию;
— возможность анонимного доступа;
— данные о пользователе могут храниться как в файле, так и в базе данных;
— возможность запретить соединения с определенных IP-адресов.

В результате анализа исходных кодов сервера выяснилось, что в процессе его работы возможна потеря и утечка информации вследствие неправильного вынесения решения о предоставлении пользователю доступа к файлам. Для предотвращения этого предлагается внедрить в сервер базовую политику РРД [6] методом АОП. Ниже показывается, как это делается в соответствии с высказанными выше рекомендациями.

6. Реализация ролевого разграничения доступа в FTP-сервере Apache методом аспектно-ориентированного программирования

Реализация РРД в FTP-сервере Apache методом АОП сводится к выполнению следующих мероприятий, рассматриваемых далее подробно:

1) реализовать основную функциональность политики РРД на языке Java, на котором написан FTP-сервер, — модули User.java, Role.java, Session.java, Privilege.java, Query.java, Action.java, TargetDescriptor.java, TargetsDescriptor.java — в соответствии с теоретическими результатами в [6] и оформить в виде пакета RBAC;
2) реализовать механизм ограничений, предусмотренный политикой РРД, на аспектно-ориетированном расширении AspectJ языка Java — RoleUser.aj, RoleSession.aj, RolePrivilege.aj, SessionChange.aj, Test.aj и часть функциональности на Java — Fragmentation.java, PrivFragmentation.java, RoleFragmentation.java;
3) дополнить данную реализацию классами, которые носят специфичные черты защищаемой КС FTP-сервер Apache — FtpTargetsDescriptor.java, FtpTargetDescriptor.java, FtpPriv.java ;
4) реализовать механизм инициализации объектов политики РРД, таких, как пользователь, роль, право доступа и т. д. — SaxHandler.java;

5) написать соединительный модуль на языке AspectJ для связи главного процесса сервера и политики безопасности — пакет RBAC-AJ.

6.1. Реализация пакета RBAC

Основными объектами, согласно теории, в базовой модели РРД являются:

1) пользователь (в текущей релизации политики — класс User);
2) роль пользователя (Role);
3) сессия пользователя (Session);
4) право доступа (Privilege).

Для наглядности описываемых классов, связей и ассоциаций между ними можно построить диаграмму классов реализации РРД (рис. 1).

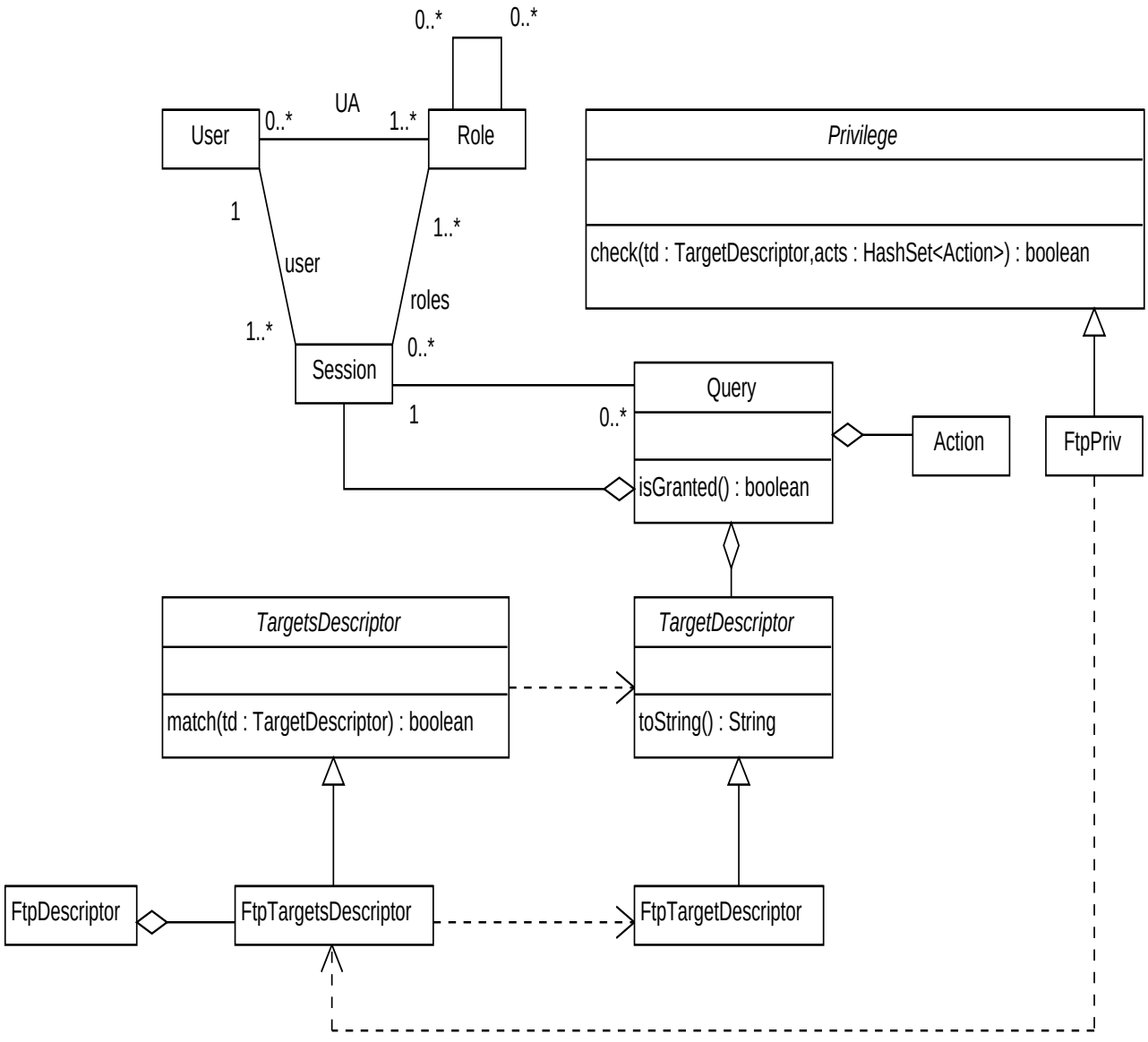

Рис. 1. Диаграмма классов реализации РРД

Для последующей поддержки любой КС, в работу которой может быть введена политика РРД, некоторые классы создаются абстрактными, что предоставляет возможность унаследовать от них класс объектов, имеющих непосредственное отношение к той КС, в работу которой внедряется данная реализация политики безопасности.

Заметим, что класс `Privilege` задуман абстрактным, так как для каждой компьютерной системы, в которую внедряется РРД, понятие «право доступа» определяется по-своему. Важной в данном классе является функция `check`. Эта функция проверяет, соответствует ли право доступа `this` тому, что передано аргументом. В данной реализации от этого класса был унаследован класс `FtpPriv`, который представляет собой право доступа на объект в хранилище FTP-сервера.

Само по себе право доступа реализуется как два объекта — набор из описателей объектов доступа пользователей — `TargetsDescriptor` и набор возможных действий над данными объектами, заданными описателями. Для FTP-сервера эти описатели реализованы в классе `FtpTargetsDescriptor` и представляют собой набор пар:

— регулярное выражение для обобщения некоторого количества имен файлов в хранилище;
— логическое значение, в итоге эмулирующее операцию разности множеств.

Для единичного файла введен класс `FtpTargetDescriptor`, который описывается именем. Для проверки, удовлетворяет ли данное имя конкретного файла регулярному выражению, реализована объявленная абстрактной в классе `TargetsDescriptor` функция `match`. Реализацию можно найти в классе `FtpTargetsDescriptor`.

Заметим, что после входа пользователя в систему для работы с сервером вся его деятельность описывается рядом запросов, которые и реализованы в виде экземпляров класса `Query`. Запрос представляет собой таблицу из двух строк и переменного числа колонок. В каждой колонке находится часть запроса — описатель файла (первая строка), к которому запрашивается некоторое множество действий (вторая строка).

Действие — также отдельный класс `Action`, экземпляры которого характеризуются типом — «read», «write» и т. д.

Согласно диаграмме и ассоциациям между классами, класс `User` представляет собой имя и множество ролей, которые ему разрешены для авторизации.

Согласно общей теории РРД, существует иерархия ролей, что и отражено на диаграмме ассоциацией класса `Role` на себя. В описываемой реализации каждая роль хранит набор потомков в виде экземпляра класса `ArrayList<Role> children`.

Для реализации ассоциации с `Privilege` в экземплярах класса Role хранится множество прав доступа, разрешенных данной роли.

В классе `Session` хранятся имя пользователя и список ролей, на которые он будет авторизован в данной сессии.

Во всех классах реализованы set/get методы.

6.2. Реализация ограничений на языке AspectJ и Java

В любой КС для того, чтобы механизмы политики безопасности функционировали не напрасно, необходимо введение некоторых ограничений на создаваемые и существующие объекты. Например, целесообразно ограничить количество пользователей, обладающих административными правами, до минимума, так как в противном случае о безопасности не может идти речи. Существуют и ограничения, которые необходимо реализовать в соответствии с теоретической моделью РРД:

1) количественное ограничение, связанное с максимальным числом пользователей, которые могут быть авторизованы на некоторую роль;

2) количественное ограничение, связанное с максимальным числом ролей пользователей, которые могут использовать некоторое право доступа;
3) количественное ограничение, связанное с максимальным числом сессий пользователей, которые могут быть авторизованы на некоторую роль;
4) статическое ограничение взаимного исключения ролей;
5) статическое ограничение взаимного исключения прав доступа;
6) динамическое ограничение взаимного исключения ролей;
7) статическое ограничение необходимости обладания ролью;
8) статическое ограничение необходимости обладания правом доступа;
9) динамическое ограничение необходимости обладания ролью.

Реализацию данных ограничений разобьем для удобства на три группы:

— реализация количественных ограничений;
— реализация ограничений взаимного исключения;
— реализация ограничений необходимости обладания.

Реализация количественных ограничений

Общий подход к реализации количественных ограничений одинаков — создать аспект с хранилищем элементов конкретно каждого ограничения, в котором количество элементов будет ограничиваться некоторой константой, полученной из конфигурационного файла.

Проиллюстрируем данный подход на примере первого количественного ограничения, связанного с числом пользователей, которые могут быть авторизованы на некоторую роль (см. листинг 1).

```
 1 import java.util.ArrayList;
 2 import java.util.HashSet;
 3 public aspect RoleUser {
 4  private ArrayList<User> Role.usersPerRole=
 5                            new ArrayList<User>();
 6  public int Role.getNumUsersPerRole()
 7  {  return usersPerRole.size(); }
 8  public boolean Role.isFullOfUsers()
 9  {if(Config.getHandler().getMaxUsersPerRole().get(getName())==
      null) return true;
10   return usersPerRole.size()==Config.getHandler().
11        getMaxUsersPerRole().get(getName());
12  }
13  public ArrayList<User> Role.getUsersPerRole()
14  {  return usersPerRole;   }
15  public void Role.addUser(User u)
16  {  usersPerRole.add(u); }
17  public boolean Role.hasNoUsers()
18  {  return usersPerRole.size()==0; }
19  public void Role.delUser(User u)
20  {  usersPerRole.remove(u); }
21 }
```

Листинг 1. Файл RoleUser.aj

В данном примере используется возможность AspectJ модифицировать статические свойства класса.

В строке 11 листинга 1 создается требуемая ограничением константа, считываемая из файла конфигурации. Далее в строке 5 создается хранилище — список пользователей, которые уже авторизованы на роль. Далее определяются функции для работы с хранилищем, а также проверки, заполнено хранилище или нет, и т. д.

Следующим этапом является написание предписания для применения данного ограничения. Для начала определяется срез точек выполнения, в которых необходимо выполнение предписания. В данном случае это будут вызовы функции из класса `User` — addRole(Role r). Необходимо заметить, что вызов этой функции может происходить как из контекста пользователя (когда объектом this является экземпляр класса User), так и в случае, когда вызывается эта функция явным указанием экземпляра класса User с использованием символа точки.

На языке AspectJ данные два среза будут выглядеть следующим образом:

```
1    public pointcut addingRole(User u,Role r):
2      (call(void addRole(Role)) && this(u)&& args(r));
3    public pointcut addingRole1(User u,Role r):
4      (call(void addRole(Role)) && target(u)&& args(r));
```

Листинг 2. Фрагмент файла Test.aj

Строка 2 в листинге 2 соответствует первому случаю, а строка 4 — второму.

Теперь объединим данные строчки в одну

«`public pointcut addingUserRole(Role r):(call(void addRole(Role)))&& args(r);`»

для того, чтобы не делать лишних предписаний. В дальнейшем будем разграничивать эти два случая при помощи свойства рефлексии — будем получать this и target прямо в момент точки выполнения следующим образом — см. строки 2–8 листинга 3.

Предписание будет выглядеть следующим образом:

```
1  before(Role r):addingUserRole(r)
2  { Object o=thisJoinPoint.getThis();
3   Object o1=thisJoinPoint.getTarget();
4   if((o instanceof User) || (o1 instanceof User)) {
5     User u;
6     RoleFragmentation f=Config.getHandler().
      getRoleFragmentation();
7     if (o instanceof User)
8        u=(User)o; else u=(User)o1;
9     try
10    { if (r.isFullOfUsers()||u.getRoles().contains(r))
11      throw new NUsersPerRoleConstraintCheckException();
12    }
13    catch (NUsersPerRoleConstraintCheckException ex)
14    { System.out.println(ex.toString()); }
15  }
16 }
```

Листинг 3. Фрагмент файла Test.aj

Для наглядности в случае неудовлетворения требованиям ограничения выбрасывается исключение — экземпляр собственного класса исключений, порожденный от стандартного класса `Exception`. Возможно в дальнейшем выдавать от сервера некоторый ответ со своим кодом.

Заполнить хранилище необходимо после того, как выполнены все ограничения. Для этого используется ключевое слово after (листинг 4):

```
1  after(Role r):addingUserRole(r)
2   {Object o=thisJoinPoint.getThis();
3    Object o1=thisJoinPoint.getTarget();
4     if ((o instanceof User)||(o1 instanceof User))
5     { User u;
6       if (o instanceof User) u=(User)o; else u=(User)o1;
7       r.addUser(u);
8     }
9   }
```

Листинг 4. Фрагмент файла Test.aj

Остальные количественные ограничения релизуются аналогично.

Реализация ограничений взаимного исключения

Общий подход к реализации ограничений взаимного исключения — создать класс, реализующий разбиение объектов на классы, и в этом классе написать функцию проверки того, можно ли добавить очередной объект в хранилище согласно разбиению.

Затем аналогично предыдущему — описание среза точек выполнения и написание предписания.

Рассмотрим случай статического взаимного исключения ролей. Для этого случая был написан абстрактный класс `Fragmentation<T>`. Его реализация показана на листинге 5. Он представляет собой реализацию разбиения множества экземпляров класса `T` на классы.

В основе — список в строке 4, элементы которого — классы разбиения. Далее реализованы основные операции с экземпляром данного класса. Основная работа возлагается на реализацию в потомках абстрактной функции проверки, какому классу принадлежит некоторый элемент.

```
1  import java.util.ArrayList;
2  import java.util.HashSet;
3  public abstract class Fragmentation<T> {
4    private ArrayList<HashSet<T>> frag;
5    public Fragmentation()
6    {   frag=new ArrayList<HashSet<T>>(); }
7    public Fragmentation(HashSet<T> r)
8    { frag=new ArrayList<HashSet<T>>();
9      frag.add(r);
10   }
11   public Fragmentation(ArrayList<HashSet<T>> rs)
12   {   frag=rs;  }
13   public ArrayList<HashSet<T>> getFrag()
14   {   return frag;  }
15   public void setFrag(ArrayList<HashSet<T>> rf)
16   {   frag=rf;  }
17   public HashSet<T> getBlock(Integer n)
18   { System.out.println("n="+n.toString());
19     return frag.get(n);
20   }
```

```
21    ...
22    public abstract Integer belongsTo(T element);
23  }
```

Листинг 5. Фрагмент файла Fragmentation.java

От класса `Fragmentation` наследуются два класса, необходимые для реализации ограничений `RoleFragmentation` и `PrivFragmentation`, которые отвечают за разбиение ролей и прав доступа на классы.

Рассмотрим `RoleFragmentation` (листинг 6). Реализация другого класса аналогична.

```
1  import java.util.ArrayList;
2  import java.util.HashSet;
3  public class RoleFragmentation extends Fragmentation<Role>{
4  public Integer belongsTo(Role r)
5    { for (Integer i=0;i<getFrag().size();i++)
6      { for (Role s:getFrag().get(i))
7          { if (s.getName().equals(r.getName()))
8            return i;
9          }
10     }
11     return -1;
12   }
13   public boolean checkAbility(User u,Role r)
14   { ArrayList<Role> userRoles=u.getRoles();
15     HashSet<Integer> blocks=new HashSet<Integer>();
16     Integer nB;
17     for (Role role:userRoles)
18     { nB=belongsTo(role);
19       if (nB!=-1)blocks.add(nB);
20     }
21     return (!blocks.contains(belongsTo(r)));
22   }
23   public boolean checkAbilityS(Session s,Role r)
24   { ArrayList<Role> sessionRoles=s.getAuthRoles();
25     HashSet<Integer> blocks=new HashSet<Integer>();
26     Integer nB;
27     for (Role role:sessionRoles)
28     { nB=belongsTo(role);
29       if (nB!=-1)blocks.add(nB);
30     }
31     return (!blocks.contains(belongsTo(r)));
32   }
33   public String toString()
34   { return getFrag().toString();  }
35   public void printBlockRoles(Integer n)
36   { if (getFrag().get(n)==null) System.out.println("No Such
      Block");
37     for (Role t:getFrag().get(n))
38     { System.out.println(t.getName()); }
39   }
40   public void print()
```

```
41    { for (Integer i=0;i<getFrag().size();i++)
42      { for (Role r:getFrag().get(i))
43        { System.out.println(r.getName());  }
44          System.out.println("");
45      }
46    }
47  }
```

Листинг 6. Файл RoleFragmentation.java

В соответствии с правилами наследования необходимо реализовать абстрактую функцию `belongsTo`, что и сделано в листинге 6 в строках 4–12. Чтобы проверить, действительно ли пользователь может быть авторизован на роли `(HashSet<Role> req)`, необходимо иметь функцию для проверки этого условия. Выглядит она следующим образом (листинг 7):

```
1  public static boolean fitFrag(HashSet<Role> req)
2  { HashSet<Role> ch;
3    RoleFragmentation rf=Config.getHandler().
      getRoleFragmentation();
4    HashSet<Integer> elements=new HashSet<Integer>();
5    Integer curblock;
6    for(Role r:req)
7    { ch=r.getAllChildren();
8      for (Role s:ch)
9      {curblock=rf.belongsTo(s);
10      System.out.println(curblock);
11      if ((curblock==-1)||(elements.contains(curblock)))
12       { return false; }
13       else
14       { elements.add(curblock); }
15     }
16       elements.clear();
17   }
18     return true;
19 }
```

Листинг 7. Файл RoleFragmentation.java

Поскольку ограничения статического и динамического взаимных исключений ролей практически идентичны для реализации, то пишутся функции проверки, можно ли пользователю `u` в список разрешенных ролей для авторизации (статическое ограничение) добавить роль `r` — строки 13–22 листинга 6, а также функция проверки, можно ли добавить сессии `s` в список ролей, на которые она авторизована, роль `r` (динамическое ограничение), что показано в строках 23–32 листинга 6. Так как статическое ограничение взаимного исключения ролей означает, что все роли, на которые может быть авторизован пользователь, находятся в разных блоках разбиения, то динамическое ограничение, в случае задания и статического, будет означать, что все запрашиваемые пользователем роли должны содержаться среди иерархически предшествующих разрешенным ролям. В случае отсутствия статического ограничения необходимо использовать функцию `CheckAbilityS`. Теперь необходимо выделить срез точек выполнения.

Во-первых, это набор `addingUserRole` из реализации количественных ограничений. Во-вторых, аналогичный набор для динамического ограничения (листинг 8):

```
1 public pointcut addingSessionRole(Session s, Role r):
2        call(void addAuthRole(Role)) && args(r);
```

Листинг 8. Фрагмент файла Test.aj

Предписания аналогичны предыдущим, с той лишь разницей, что в этот раз используются написанные функции проверок `checkAbility(User u, Role r)` и `checkAbilityS(Session s, Role r)`.

Реализация ограничений необходимости обладания

Основная идея реализации статических ограничений необходимости обладания состоит в том, чтобы создать для каждого объекта множество объектов, обладание которыми необходимо для обладания самим объектом.

Рассмотрим реализацию на примере ограничения необходимости обладания ролью.

Класс `Config` реализует шаблон *синглтон*, что означает, что экземпляр данного класса единственный. В процессе разбора файла конфигурации заполняется данное хранилище. В дальнейшем оно запрашивается у этого единственного экземпляра. Но так как это разбиение запрашивается еще в процессе разбора конфигурации, то запрос будем производить у самого обработчика. Для этого используется статическая функция `getHandler()` в классе `Config`.

Для эффективного хранения информации из конфигурационного файла используется словарь с эффективным поиском по ключу — `HashMap`. Например, в листинге 9 в строке 1 хранится (имя роли; максимальное количество пользователей на роль):

```
1 private HashMap<String,Integer> MaxUsersPerRoleContainer;
2 private HashMap<String,Integer> MaxRolesPerPrivilegeContainer;
3 private HashMap<String,Integer> MaxSessionsPerRoleContainer;
4 private HashMap<String,ArrayList<Role>>
     NeededRolesPerRoleContainer;
5 private HashMap<String,HashSet<Privilege>>
     NeededPrivsPerPrivContainer;
```

Листинг 9. Фрагмент файла Test.aj

Проверки реализованы функциями в аспекте (листинг 10):

```
1  public boolean checkAddingPrivilegeAbilityN(Role r,
2                                              Privilege p)
3  {System.out.println("checking privilege "+p.getName());
4   System.out.println(Config.getHandler().
      getNeededPrivsPerPrivilege());
5   if (Config.getHandler().getNeededPrivsPerPrivilege().
6      get(p.getName())==null)
7          return true;
8   return r.getPrivs().containsAll(Config.getHandler().
9      getNeededPrivsPerPrivilege().get(p.getName()));
10 }
11 public boolean checkAddingRoleAbilityN(User u, Role r)
12 {System.out.println("Checking role "+r.getName());
13  System.out.println(Config.getHandler().
```

```
14                getNeededRolesPerRole());
15   if (Config.getHandler().getNeededRolesPerRole().
16                get(r.getName())==null)
17      return true;
18   return u.getRoles().containsAll(Config.getHandler().
19             getNeededRolesPerRole().get(r.getName()));
20 }
21 public boolean checkAddingRoleToSessionAbilityN(Session s,
22                                                  Role r)
23 {if (s.getNeededRoles(r)==null) return true;
24   return s.getAuthRoles().containsAll(s.getNeededRoles(r));
25 }
```

Листинг 10. Фрагмент файла Test.aj

Срез точек выполнения для данного ограничения — `addingUserRole`. Предписание выглядит вызовом одной функции из листинга 10. Например (листинг 11):

```
1 try
2 {if (checkAddingRoleAbilityN(u, r)==false) throw new
     NeededRolesConstraintCheckException();
3 }
4 catch (NeededRolesConstraintCheckException ex)
5 { ex.printStackTrace();
6   System.out.println(ex.toString());
7 }
```

Листинг 11. Фрагмент файла Test.aj из before(Role r):addingUserRole(r)

7. Внедрение политики РРД в Apache Ftp Server

Для внедрения политики РРД в работу Ftp-сервера создадим пакет `RBAC-AJ`, который помещается в проект `ftpserver-core-1.0.x` и связывает его и пакет `RBAC`.

Для иллюстрации возможных ситуаций в работе сервера будем полагать, что у всех пользователей одна и та же виртуальная домашняя директория — '/'.

В главном файле `rbac.aj` располагаются, в том числе, предписания на выполнение в точках перехвата введённых пользователем команд проверки введённых данных на соответствие политике безопасности РРД.

В частности, необходим перехват команд `USER, PASS, CWD, MKD, RMD, STOU, RNFR, LIST`, а также всех команд, описанных в [13]. В дальнейшем предлагается сопоставить команды сервера и действия над объектами хранилища для определения необходимых прав доступа пользователя к данным объектам.

Создадим вспомогательный класс `SessionManager`, который будет хранить сопоставления уникального идентификатора сессии UUID и объекта сессии (строка 8 листинга 12), а также UUID и ролей, которые запрашивает пользователь (строка 10 листинга 12). Данный класс реализует шаблон *синглтон* — строки 11–13 листинга 12.

```
1 package rbac;
2 import java.util.HashMap;
3 import java.util.HashSet;
4 import java.util.UUID;
5 import org.apache.ftpserver.impl.FtpIoSession;
6 public class SessionManager {
```

```
 7 private static HashMap<UUID,my.Session> sessionsContainer=
 8  new HashMap<UUID,my.Session>();
 9 private static HashMap<UUID,HashSet<my.Role>>
10  sessionRolesContainer=new HashMap<UUID,HashSet<my.Role>>();
11 public static my.Session getInstance(FtpIoSession fs){
12  return sessionsContainer.get(fs.getSessionId());
13  }
14 ...
15 }
```

Листинг 12. Из проекта ftpserver-core-1.0.x (пакет src/java/main/rbac, файл SessionManager.java)

7.1. Инициализация РРД

Вследствие предоставления языком AspectJ возможностей по написанию предписаний типа `around`, имя пользователя и список ролей, на которые претендует пользователь, запрашиваются вместе с командой `USER` сервера, а затем в предписании извлекается список ролей, а серверу передаётся корректный запрос команды `USER`.

Формат команды `USER`: `USER имя_пользователя роль1[роль2 ... рольN]`.

Таким образом достигается совместимость с любыми ftp-клиентами, поскольку не вводятся дополнительные команды для протокола FTP.

Формировать сессию РРД будем в случае успешной аутентификации пользователя в рамках ftp-сервера, что проиллюстрировано в строках 4–10 листинга 13.

```
 1 void around(Command c,FtpIoSession s,FtpServerContext ctx,
     FtpRequest r):usercommand(c,s,ctx,r) && if(c instanceof
     PASS)
 2 { proceed(c,s,ctx,r);
 3   if (s.getUser()!=null){
 4   my.Session currentSession=new Session();
 5   currentSession.setUser(Config.getDefaultInstance()
 6   .getUserContainer().get(s.getUser().getName()));
 7   currentSession.addAuthRoles(SessionManager.getRoles(s));
 8   SessionManager.addSession(s, currentSession);
 9   SessionManager.delRoles(s);
10  }
11 }
```

Листинг 13. Перехват PASS

Перед стартом сервера необходимо разобрать файл конфигурации, что производится построением экземпляра класса `Config`: `Config.getDefaultInstance();`.

В соответствии с теоретической моделью РРД, так как отношение иерархии ролей не обязательно является деревом (и даже полурешёткой), то и в данной реализации у каждой роли могут быть одна или несколько родительских ролей, что влечет за собой несколько тегов `<parent>` у роли.

7.2. Формат файла конфигурации

В качестве приемлемого способа инициализации параметров политики РРД и гибкого конфигурирования выберем хранение конфигурации в файле формата XML [11]. В соответствии со стандартом формата XML все содержимое конфигурационного файла заключено в тег `<configuration>` и соответствующий закрывающий тег

</configuration>. У данного тега есть три атрибута, олицетворяющие собой настройки по умолчанию для количественных ограничений.

Структура файла конфигурации разбита на следующие блоки:

1) определение прав доступа;
2) разбиение прав доступа на блоки;
3) определение ролей;
4) разбиение множества ролей на блоки;
5) определение пользователей.

Определение права доступа осуществляется между открывающим и закрывающим тегами <privilegedef>; атрибут name задаёт имя права доступа, которое в дальнейшем может быть использовано для ссылки на него.

В соответствии с методикой реализации РРД у права доступа есть список регулярных выражений (описывается с помощью тега <targetsd>), описывающих набор объектов, к которому оно применяется, флаг инвертирования (атрибут inverted тега <targetsd>) и набор действий — список открывающих и закрывающих тегов <action>.

При определении права доступа можно установить количественное ограничение числа ролей на него — <maxRolesPerPrivilege>.

Для каждого права доступа могут быть заданы другие права доступа, на которые роль также должна быть авторизована. Они должны быть перечислены по одному в тегах <neededPriv> и </neededPriv>.

Разбиение множества прав доступа заключено между открывающим и закрывающим тегами privfragmentation, в которых каждый отдельный класс разбиения заключен между тегами <privblock> и </privblock>. Каждое право в блоке идентифицируется по его имени, заданном при определении, и заключается в теги <privpart> и </privpart>.

Определение роли осуществляется между открывающим и закрывающим тегами roledef. В это определение могут входить список прав доступа (в тегах <priv> и </priv>), а также родительские роли в тегах parent по одной, количественные ограничения внутри тегов maxUsersPerRole и maxSessionsPerRole и необходимости обладания ролью — neededRole.

Разбиение множества ролей на классы аналогично разбиению множества прав доступа, за исключением замены в именах тегов priv на role.

Определение пользователей происходит внутри тегов <userdef> и </userdef> и имеет атрибут name, который и вводится пользователем в команде USER и есть в файле пользователей сервера. Далее можно определить роли, на которые может быть авторизован пользователь — <role>rolename</role>.

7.3. Функционирование политики безопасности

Рассмотрим общую схему функционирования политики безопасности.

Пользователь обращается на FTP-сервер с запросом предоставить ему доступ к хранилищу файлов под именем — первым аргументом команды USER — с ролями, список которых он указывает сразу после имени пользователя в команде USER.

В момент поступления команды от пользователя задействуется функциональность политики безопасности путём передачи в специальном виде информации о запросе пользователя, и затем анализируется ответ от модуля политики. Если ответ на запрос пользователя положительный, то запрашиваемый доступ предоставляется, иначе — запрещается.

Для успешного функционирования вышеописанной процедуры проверки на возможность предоставления доступа создаётся экземпляр класса `Query` и вызвается метод `isGranted()`. Эта последовательность действий производится в предписаниях-перехватчиках команд пользователя, реализация которых похожа друг на друга структурой:

— сохранить строку — аргумент команды, если таковой имеется;
— преобразовать полученную строку к полному пути относительно виртуальной директории сервера;
— интерпретировать название команды, полученной от пользователя, в последовательность требуемых для ее успешного выполнения прав доступа, которым должна обладать хотя бы одна роль, на которую авторизован данный пользователь;
— проверить наличие у пользователя необходимых привилегий и вынести решение — продолжать выполнение команды сервером или сгенерировать соответствующее исключение.

Для начала необходимо определить срез точек выполнения — моменты перехвата вводимых команд (листинг 14). Заметим, что вызовы команд однотипны — в интерфейсе под именем названия команды существует единственная функция — `void execute(...)`.

```
1  pointcut usercommand(Command c,FtpIoSession s,
     FtpServerContext ctx,FtpRequest r):call(void execute
2    (FtpIoSession,FtpServerContext,FtpRequest))
3     && target(c) && !adviceexecution() && args(s,ctx,r);
```

Листинг 14. Фрагмент файла Rbac.aj — срез моментов перехвата команд

Рассмотрим по порядку возможную реализацию перехватчиков.

Работа с аргументом команды

У интерфейса `FtpRequest` существуют функции по обработке аргументов запроса пользователя. Основные из них перечислены в листинге 15.

```
1  /* Get the client request string.
2  * @return The full request line, e.g. "MKDIR newdir"
3  */
4  String getRequestLine();
5  /* Returns the ftp request command.
6  * @return The command part of the request line, e.g. "MKDIR"
7  */
8  String getCommand();
9  /* Get the ftp request argument.
10 * @return The argument part of the request line, e.g. "newdir"
11 */
12 String getArgument();
13 /* Check if request contains an argument
14 * @return true if an argument is available
15 */
16 boolean hasArgument();
```

Листинг 15. Из пакета org.apache.ftpserver.ftplet — FtpRequest.java

С их помощью требуемая работа очевидна.

Преобразование полученной строки к полному пути относительно виртуальной директории сервера

Для решения этой задачи необходимо воспользоваться строчкой 6 листинга 16, которая позволяет получить полный путь текущей рабочей директории в сессии пользователя. Сделав некоторые поправки на разницу между именем файла и именем директории, получим искомый результат на листинге 16.

```
1  public String getFullName(FtpIoSession s,String name)
2  { String nD=name;
3    if ((nD.length()>0)&&(nD.charAt(0)!='/'))
4    { String wd="";
5      try{wd=s.getFileSystemView().getWorkingDirectory().
6          getAbsolutePath();
7        }
8      catch(FtpException ex)
9      { ex.printStackTrace(); }
10     if (wd.charAt(wd.length()-1)=='/')
11     { nD=wd+name;   }
12     else
13     { nD=wd+"/"+name; }
14   }
15   return nD;
16 }
```

Листинг 16. Фрагмент файла Rbac.aj- getFullName

В последующем потребуется функция определения родительской директории объекта (листинг 17).

```
1  public static String upper(String name)
2    { Integer index;
3      index=name.lastIndexOf("/");
4      if ((index==-1)||(index==0)) return "/";
5      return name.substring(0,index);
6    }
```

Листинг 17. Фрагмент файла Rbac.aj-upper

Интерпретировать название команды FTP-сервера в последовательность требуемых прав доступа

Для решения данной задачи необходимы сведения о выполняемых командой действиях. Соответственно этим сведениям определяются следующие наборы привилегий, где объект — это аргумент команды:

- APPE — в случае существования объекта право записи в объект, иначе — право записи в родительскую директорию;
- CWD — в случае присутствия аргумента команды право чтения объекта, иначе — право чтения корневой виртуальной директории;
- DELE — право записи в родительскую директорию;
- LIST — право чтения объекта;
- MKD — право записи в родительскую директорию объекта;
- RETR — право чтения объекта;
- RMD — право записи в родительскую директорию объекта;

— RNFR — право записи в родительскую директорию объекта;
— RNTO — право записи в родительскую директорию объекта;
— STOR — в случае существования объекта право записи в объект, иначе — право записи в родительскую директорию;
— STOU — право записи в родительскую директорию объекта.

Проверка требуемых прав доступа и вынесение решения

Проиллюстрируем все шаги на примере команды RETR (листинг 18).

```
1  void around(Command c,FtpIoSession s,FtpServerContext ctx,
       FtpRequest r):
2            usercommand(c,s,ctx,r) && if(c instanceof RETR)
3  { String nD = r.getArgument();
4    nD=getFullName(s, nD);
5    Query q=new Query();
6    FtpTargetDescriptor ftd=new FtpTargetDescriptor(nD);
7    HashSet<Action> hs=new HashSet<Action>();
8    q.addColumn(ftd, hs);
9    q.setSession(SessionManager.getInstance(s));
10   if (q.isGranted())
11           proceed(c,s,ctx,r);
12   else
13           s.write(LocalizedFtpReply.translate(s, r, ctx,
14           FtpReply.REPLY_550_REQUESTED_ACTION_NOT_TAKEN, "RETR",
15           nD));
16 }
```

Листинг 18. Фрагмент файла Rbac.aj- around RETR

В листинге 18 используется написанная ранее в классе `Query` функция `isGranted`, которая и принимает требумое решение. В строках 6 и 9 происходит конструирование запроса с заполнением объекта, к которому требуется доступ и действие чтения, что соответствует команде RETR.

В дальнейшем, если решение положительное (строка 11), то выполнение команды продолжится, иначе (строка 13) — генерируется ответ сервера с кодом 550. При использовании графических ftp-клиентов пользователю будет выдано развернутое сообщение об ошибке.

Заключение

Аспектно-ориентированное программирование — одна из важнейших парадигм программрования, которую можно успешно использовать для написания и внедрения в компьютерные системы политик безопасности. Она вводит требуемый для этого уровень модульности и гибкости внедрения. При помощи заключенных в ней механизмов существует возможность взаимодействовать с пользователем, встраивать политики безопасности и другие модули без модификации исходных кодов компьютерной системы.

В статье сформулированы основные требования и рекомендации к реализации и внедрению политик безопасности в защищаемые КС методом АОП. По этим рекомендациям осуществлено внедрение базовой политики ролевого разграничения доступа в FTP-сервер Apache, не обладающий механизмом разграничения доступа пользователей к файлам друг друга. Технология этого внедрения продемонстрирована в разд. 6

и 7, где представлены соответственно реализация основной функциональности РРД — пакет RBAC, содержащий реализации на языке Java основных объектов политики РРД, таких, как Пользователь (User.java), Роль (Role.java) и т. д., и реализация пакета RBAC-AJ — соединительного модуля между сервером и пакетом RBAC.

ЛИТЕРАТУРА

1. *Таненбаум Э.* Современные операционные системы. 2-е изд. СПб.: Питер, 2009. 314 с.
2. *Dijkstra E. W.* Selected Writings on Computing: A Personal Perspective. London; New York: Springer Verlag, 1982. С. 60–66.
3. *Elrad T., Aksit M. M., Kiczales G., et al.* Discussing Aspects of AOP // Comm. ACM. 2001. V. 44. No. 10. P. 33–38.
4. `www.ddj.com/architect/184414752` — Through the Looking Glass. 2001.
5. *Стефанцов Д. А.* Реализация политик безопасности в компьютерных системах с помощью аспектно-ориентированного программирования // Прикладная дискретная математика. 2008. № 1. С. 94–100.
6. *Девянин П. Н.* Модели безопасности компьютерных систем. М.: Академия, 2005. 144 с.
7. `http://java.sun.com/docs/books/jls/` — The Java books. 2010.
8. `http://eclipse.org/aspectj` — AspectJ Home Page. 2010.
9. `http://aophp.net` — Aspect Oriented PHP. 2010.
10. `http://www.uml.org` — Unified Modelling Language. 2010.
11. `http://tools.ietf.org/html/rfc3076` — RFC 3076 - Canonical Xml Version 1.0. 2010.
12. `http://mina.apache.org/ftpserver` — Apache Ftp Server. 2010.
13. `http://tools.ietf.org/html/rfc959` — RFC 959 - Ftp. 2010.

ПРИКЛАДНАЯ ТЕОРИЯ АВТОМАТОВ

DOI 10.17223/20710410/7/5

УДК 519.17

КАРКАС АВТОМАТА

В. Н. Салий

Саратовский государственный университет им. Н. Г. Чернышевского, г. Саратов, Россия

E-mail: saliiVN@info.sgu.ru

Каркасом автомата (без выходов) называется упорядоченное множество, которое образуют слои автомата (т. е. его сильно связные подмножества) вместе с отношением обратной достижимости. Установлены некоторые свойства каркаса автомата, связанные с основными алгебраическими конструкциями, такими, как подавтоматы, гомоморфизмы и конгруэнции.

Ключевые слова: *автомат, каркас автомата, подавтомат, гомоморфизм, конгруэнция, упорядоченное множество.*

В [1] было введено понятие каркаса автомата. Это упорядоченное множество, которое образуют слои автомата вместе с отношением обратной достижимости. Оно сыграло весьма существенную роль в описании автоматов, у которых каждая конгруэнция является ядром подходящего эндоморфизма. В предлагаемой работе устанавливаются некоторые свойства каркаса, связанные с основными алгебраическими конструкциями для автоматов, такими, как подавтомат, гомоморфизм, конгруэнция.

Автомат — это тройка $\mathbf{A} = (S, X, \delta)$, где S и X — конечные непустые множества, соответственно множество состояний и множество входных сигналов, а $\delta : S \times X \to S$ — отображение, называемое функцией переходов. Запись $\delta(s, x) = t$ для $s, t \in S$ и $x \in X$ означает, что автомат $\mathbf{A}$, находящийся в состоянии s, под действием входного сигнала x переходит в состояние t.

Подмножество $S' \subseteq S$ называется устойчивым в автомате $\mathbf{A}$, если $\delta(s, x) \in S'$ для любых $s \in S'$ и $x \in X$. Если S' устойчиво в $\mathbf{A}$, то, ограничивая функцию переходов δ на $S' \times X$, получают автомат $\mathbf{A}' = (S', X, \delta)$ — подавтомат автомата $\mathbf{A}$, соответствующий S'. Совокупность $\mathrm{Sub}\mathbf{A}$ всех подавтоматов автомата $\mathbf{A}$ (сюда включается и нулевой подавтомат $\mathbf{0} = (\varnothing, X, \varnothing)$) упорядочивается отношением $\mathbf{A_1} \leqslant \mathbf{A_2} \Longleftrightarrow S_1 \subseteq S_2$, где $\mathbf{A}_i = (S_i, X, \delta)$, $i = 1, 2$. Упорядоченное множество $(\mathrm{Sub}\mathbf{A}, \leqslant)$ является дистрибутивной решеткой. Решеточные операции в ней задаются равенствами $\mathbf{A_1} \wedge \mathbf{A_2} = (S_1 \cap S_2, X, \delta)$ и $\mathbf{A_1} \vee \mathbf{A_2} = (S_1 \cup S_2, X, \delta)$.

Пусть $\mathbf{A} = (S, X, \delta)$ и $\mathbf{B} = (T, X, \delta)$ — сравнимые автоматы, т. е. автоматы с одним и тем же множеством входных сигналов (функции переходов всех автоматов будем обозначать символом δ). Отображение $\varphi : S \to T$ по определению является гомоморфизмом автомата $\mathbf{A}$ в автомат $\mathbf{B}$, если $\varphi(\delta(s, x)) = \delta(\varphi(s), x)$ для любых $s \in S, x \in X$. Взаимно однозначные гомоморфизмы автоматов называются вложениями, а их биективные гомоморфизмы — изоморфизмами. Эндоморфизмы автомата — это его гомоморфизмы в себя, автоморфизмы — изоморфизмы на себя.

Отношение эквивалентности $\theta \subseteq S \times S$ называется конгруэнцией автомата $\mathbf{A} = (S, X, \delta)$, если оно согласовано с функцией переходов в том смысле, что

$(\forall s,t\in S)(\forall x\in X)((s,t)\in\theta\Longrightarrow(\delta(s,x),\delta(t,x))\in\theta)$. Каждая конгруэнция θ автомата $\mathbf{A}$ определяет его фактор-автомат $\mathbf{A}/\theta=(S/\theta,X,\delta)$, где $\delta(\theta(s),x):=\theta(\delta(s,x))$ для любых $s\in S,x\in X$.

Пусть X^* — множество всех конечных слов над алфавитом X. Продолжим функцию переходов δ на множество $S\times X^*$, полагая $\delta(s,e)=s$, где $e\in X^*$ — пустое слово, и $\delta(s,px)=\delta(\delta(s,p),x)$ для любых $s\in S$, $x\in X$, $p\in X^*$. Говорят, что состояние t достижимо в автомате $\mathbf{A}$ из состояния s, если найдется входное слово $p\in X^*$, такое, что $\delta(s,p)=t$. Записывая это в виде $(s,t)\in\tau$, вводим отношение достижимости τ в автомате $\mathbf{A}$. Понятно, что τ рефлексивно и транзитивно, т. е. является квазипорядком на S. Пусть $s\in S$ — произвольное состояние автомата $\mathbf{A}$. Подмножество $\tau(s)$, объединяющее все состояния, достижимые из s, устойчиво в $\mathbf{A}$ и, следовательно, определяет подавтомат $\mathbf{A}(s)=(\tau(s),X,\delta)$. Это — главный подавтомат, порожденный состоянием s.

Симметричная часть $\sigma=\tau\cap\tau^{-1}$ отношения достижимости называется отношением взаимной достижимости в $\mathbf{A}$. Очевидно, что σ будет эквивалентностью на множестве состояний S. Классы этой эквивалентности называют слоями автомата $\mathbf{A}$.

Каркасом автомата $\mathbf{A}$ назовем упорядоченное множество $F(\mathbf{A})=(S/\sigma,\tau^{-1})$. Его элементами являются слои автомата $\mathbf{A}$, а порядком — отношение, обратное достижимости, перенесенное на слои: $(\sigma(t),\sigma(s))\in\tau^{-1}$ равносильно тому, что $(s,t)\in\tau$.

Теорема 1. Каждое конечное упорядоченное множество изоморфно каркасу подходящего автомата с двумя входными сигналами.

Доказательство. Пусть $\mathbf{P}=(P,\leqslant)$ — конечное упорядоченное множество. Построим автомат $\mathbf{A}=(S,X,\delta)$, такой, что $|X|=2$ и $F(\mathbf{A})\cong\mathbf{P}$. (Здесь и далее знаком $\cong$ обозначается изоморфизм соответствующих структур.)

Каждому минимальному элементу $a\in P$ сопоставим символ $s_0(a)$. Если элемент a не минимален и имеет k нижних соседей, то сопоставим a символы $s_i(a),0\leqslant i\leqslant k-1$. Проделав эту процедуру для всех элементов из P, получим множество S состояний автомата $\mathbf{A}$. (Элемент a упорядоченного множества $(P,\leqslant)$ называется нижним соседом элемента b, если $a<b$ и $\neg(\exists x\in P)(a<x<b)$.)

Далее, положим $X=\{0,1\}$. Определим функцию переходов $\delta:S\times X\to S$ следующим образом.

Если a является минимальным элементом в упорядоченном множестве $\mathbf{P}$, то $\delta(s_0(a),0)=\delta(s_0(a),1)=s_0(a)$.

Если a — не минимальный элемент и у него k нижних соседей, то состояния $s_0(a),s_1(a),\dots,s_{k-1}(a)$ по входному сигналу 0 образуют контур в указанном порядке их прохождения (последнее состояние переходит в первое).

Если a — не минимальный элемент и $a_0,a_1,\dots,a_{k-1}$ — его нижние соседи, то $\delta(s_i(a),1)=s_0(a_i),0\leqslant i\leqslant k-1$.

Покажем, что $\mathbf{P}\cong F(\mathbf{A})$.

В автомате $\mathbf{A}$ слоями будут в точности подмножества вида $\{s_0(a)\}$, где a — минимальный в $\mathbf{P}$ элемент, и подмножества вида $\{s_0(a),s_1(a),\dots,s_{k-1}(a)\}$, где a имеет $k\geqslant 1$ нижних соседей.

Определим отображение $\varphi:P\to F(\mathbf{A})$, полагая $\varphi(a)=\sigma(s_0(a))$. Очевидно, что φ является взаимно однозначным соответствием между множествами P и $F(\mathbf{A})$.

Пусть $a<b$ в $\mathbf{P}$. Возьмем в $\mathbf{P}$ какую-нибудь неуплотняемую возрастающую цепь $a=x_1<x_2<\dots<x_l=b$. Так как x_i является нижним соседом для $x_{i+1},1\leqslant i\leqslant l-1$, то $\delta(s_0(x_{i+1}),1)=s_0(x_i)$, и значит, $\delta(s_0(b),1^{l-1})=s_0(a)$, откуда $\varphi(a)=\sigma(s_0(a))<$ $<\sigma(s_0(b))=\varphi(b)$ в $F(\mathbf{A})$.

С другой стороны, если $\varphi(a) < \varphi(b)$, т. е. $\sigma(s_0(a)) < \sigma(s_0(b))$ в $F(\mathbf{A})$, то состояние $s_0(a)$ достижимо в $\mathbf{A}$ из состояния $s_0(b)$. Входным словом минимальной длины, переводящим $s_0(b)$ в $s_0(a)$, является слово вида $1^{l-1}, l > 1$. Подавая его на вход автомата $\mathbf{A}$, получим последовательность состояний $s_0(b) = s_0(x_l), s_0(x_{l-1}), \ldots, s_0(x_1) = s_0(a)$. При этом x_i будет нижним соседом в $\mathbf{P}$ для $x_{i+1}, 1 \leqslant i \leqslant l-1$. Таким образом, в $\mathbf{P}$ получаем убывающую цепь от b к a, откуда $a < b$.

Итак, отображение φ биективно, изотонно и обратно изотонно, т. е. является изоморфизмом упорядоченного множества $\mathbf{P}$ на упорядоченное множество $F(\mathbf{A})$. ■

Теорема 2. Конечное упорядоченное множество тогда и только тогда изоморфно каркасу автономного автомата, когда у каждого его элемента имеется не более чем один нижний сосед.

Доказательство. Граф переходов автономного автомата представляет собой объединение функциональных графов, т. е. орграфов, в которых степень исхода каждой вершины равна 1. Слоями такого автомата будут одноэлементные множества, соответствующие состояниям, не входящим в контуры, а также сами контуры. В диаграмме каркаса автономного автомата, если под высотой элемента (слоя) понимать удаленность соответствующего состояния от контура, контурам соответствуют минимальные элементы, у них нет нижних соседей, у каждого неминимального элемента имеется в точности один нижний сосед.

С другой стороны, если диаграмма упорядоченного множества удовлетворяет условию теоремы, то эту диаграмму можно преобразовать в граф переходов автономного автомата путем ориентации ее ребер от больших элементов к меньшим и присоединения петель во всех минимальных элементах. Диаграмма каркаса построенного автомата совпадет с диаграммой исходного упорядоченного множества. ■

Теорема 3. Если $\mathbf{A}$ и $\mathbf{B}$ — произвольные автоматы, то $\mathrm{Sub}\mathbf{A} \cong \mathrm{Sub}\mathbf{B}$ тогда и только тогда, когда $F(\mathbf{A}) \cong F(\mathbf{B})$.

Доказательство. Пусть L — конечная дистрибутивная решетка. Ненулевой элемент $a \in L$ называется неразложимым, если для любых $x, y \in L$ равенство $a = x \vee y$ влечет одно из равенств $a = x$ или $a = y$. Это означает, что у a имеется в L единственный нижний сосед. Например, неразложимыми элементами решетки подавтоматов являются в точности главные подавтоматы (см. [2], с. 174, лемма 2.5).

Известно (см. [2], с. 174, следствие 2.20), что конечная дистрибутивная решетка с точностью до изоморфизма определяется своим упорядоченным подмножеством, которое образуют ее неразложимые элементы.

Пусть $\mathrm{PSub}\mathbf{A}$ обозначает упорядоченное множество всех главных подавтоматов автомата $\mathbf{A}$. Покажем, что $\mathrm{PSub}\mathbf{A} \cong F(\mathbf{A})$.

Рассмотрим отображение $\varphi : \mathrm{PSub}\mathbf{A} \to F(\mathbf{A})$, $\mathbf{A}(s) \mapsto \sigma(s)$. Очевидно, что φ сюръективно. Если $\varphi(\mathbf{A}(s)) = \varphi(\mathbf{A}(t))$, то $\sigma(s) = \sigma(t)$, откуда $\tau(s) = \tau(t)$, и значит, $\mathbf{A}(s) = \mathbf{A}(t)$, что означает инъективность отображения φ. Далее, так как $\mathbf{A}(s) \leqslant \mathbf{A}(t) \iff \tau(s) \subseteq \tau(t) \iff (t, s) \in \tau \iff \sigma(s) \leqslant \sigma(t)$ для любых $s, t \in S$, то получаем, что биекция φ изотонна и обратно изотонна, т. е. является изоморфизмом упорядоченных множеств $\mathrm{PSub}\mathbf{A}$ и $F(\mathbf{A})$.

Доказательство завершается следующей формулой: $\mathrm{Sub}\mathbf{A} \cong \mathrm{Sub}\mathbf{B} \iff \mathrm{PSub}\mathbf{A} \cong \mathrm{PSub}\mathbf{B} \iff F(\mathbf{A}) \cong F(\mathbf{B})$. ■

Теорема 4. Если φ — вложение автомата $\mathbf{A}$ в автомат $\mathbf{B}$ и $F(\mathbf{A}) \cong F(\mathbf{B})$, то φ — изоморфизм $\mathbf{A}$ на $\mathbf{B}$.

Доказательство. Нужно показать, что φ — сюръективное отображение.

В каждом слое автомата $\mathbf{A}$ выберем одно состояние и обозначим полученное множество через S_0. В множестве состояний автомата $\mathbf{B}$ образуем подмножество $T_0 = \{t|(\exists s \in S_0)(\varphi(s) = t\}$. Покажем, что все элементы этого множества лежат в разных слоях автомата $\mathbf{B}$. Пусть это не так и $\sigma(t) = \sigma(t')$, где $t = \varphi(s), t' = \varphi(s')$ и $s, s' \in S_0$. Тогда существует слово $p \in X^*$, такое, что $\delta(t, p) = t'$. Отсюда имеем $\varphi(s') = t' = \delta(t, p) = \delta(\varphi(s), p) = \varphi(\delta(s, p))$. В силу инъективности φ получаем $s' = \delta(s, p)$, и значит, $(s, s') \in \tau$. Аналогично $(s', s) \in \tau$, откуда $(s, s') \in \sigma$, т. е. $\sigma(s) = \sigma(s')$, что невозможно по построению множества S_0. Поскольку $|T_0| = |S_0| = |F(\mathbf{A})| = |F(\mathbf{B})|$, то T_0 является селективным множеством для эквивалентности σ в автомате $\mathbf{B}$.

Пусть теперь t — произвольное состояние автомата $\mathbf{B}$. Существует $t' \in T_0$, такое, что $\sigma(t) = \sigma(t')$. Значит, $t = \delta(t', p)$ для подходящего $p \in X^*$. Так как $t' = \varphi(s')$ для некоторого $s' \in S_0$, то $t = \delta(t', p) = \delta(\varphi(s'), p) = \varphi(\delta(s', p))$, т. е. $t \in \mathrm{pr}_2\varphi$. Следовательно, φ сюръективно. Итак, φ — инъективный и сюръективный гомоморфизм, т. е. изоморфизм автомата $\mathbf{A}$ на автомат $\mathbf{B}$. ■

Следствие 1. Если $\mathbf{A}'$ — подавтомат автомата $\mathbf{A}$ и $F(\mathbf{A}') \cong F(\mathbf{A})$, то $\mathbf{A}' = \mathbf{A}$.

Доказательство. Тождественное отображение из множества состояний автомата $\mathbf{A}'$ в множество состояний автомата $\mathbf{A}$ является автоматным вложением. Согласно теореме 4, оно будет изоморфизмом. ■

Следствие 2. Если φ — эндоморфизм автомата $\mathbf{A}$ и $F(\varphi(\mathbf{A})) \cong F(\mathbf{A})$, то φ — автоморфизм.

Доказательство. Поскольку эндоморфный образ $\varphi(\mathbf{A})$ является подавтоматом автомата $\mathbf{A}$, то из следствия 1 получаем требуемое утверждение. ■

Теорема 5. Если θ — конгруэнция автомата $\mathbf{A}$, то $F(\mathbf{A}/\theta) \cong F(\mathbf{A})$ тогда и только тогда, когда $\theta \subseteq \sigma$.

Доказательство. Пусть θ — произвольная конгруэнция автомата $\mathbf{A} = (S, X, \delta)$.

I. Предположим, что $\theta \subseteq \sigma$, и докажем изоморфность упорядоченных множеств (каркасов) $F(\mathbf{A}/\theta)$ и $F(\mathbf{A})$.

Прежде всего, заметим, что естественное отображение $nat\theta : S \to S/\theta$, $s \mapsto \theta(s)$, согласовано с отношениями достижимости в автоматах $\mathbf{A}$ и $\mathbf{A}/\theta$. В самом деле,

$$(s, t) \in \tau \Longrightarrow (\exists p \in X^*)(\delta(s, p) = t) \Longrightarrow$$
$$\Longrightarrow (\exists p \in X^*)(\delta(\theta(s), p) = \theta(\delta(s, p)) = \theta(t)) \Longrightarrow (\theta(s), \theta(t)) \in \tau.$$

Отсюда следует, что $(s, t) \in \sigma$ влечет $(\theta(s), \theta(t)) \in \sigma$.

Рассмотрим теперь отображение $\varphi : F(\mathbf{A}/\theta) \to F(\mathbf{A})$, полагая $\varphi(\sigma(\theta(s))) = \sigma(s)$.

В силу сделанного выше замечания, φ определено корректно, т. е. не зависит от выбора конкретных представителей в классах $\theta(s)$ и $\sigma(s)$. Действительно, если $\theta(t) = \theta(s)$, то $(s, t) \in \theta \subseteq \sigma$, откуда $\varphi(\sigma(\theta(t))) = \varphi(\sigma(\theta(s))) = \sigma(s) = \sigma(t)$.

Очевидно, что φ сюръективно. Покажем, что оно и инъективно. В самом деле,

$$\varphi(\sigma(\theta(s))) = \varphi(\sigma(\theta(t))) \Longrightarrow \sigma(s) = \sigma(t) \Longrightarrow (s, t) \in \sigma \Longrightarrow$$
$$\Longrightarrow (\theta(s), \theta(t)) \in \sigma \Longrightarrow \sigma(\theta(s)) = \sigma(\theta(t)).$$

Итак, φ — биекция.

Теперь доказываем ее изотонность:

$$\sigma(\theta(s)) \leqslant \sigma(\theta(t)) \Longrightarrow (\theta(t), \theta(s)) \in \tau \Longrightarrow (\exists p \in X^*)(\delta(\theta(t), p) = \theta(s) \Longrightarrow$$
$$\Longrightarrow (\exists p \in X^*)(\theta(\delta(t, p)) = \theta(s)) \Longrightarrow (\exists p \in X^*)((\delta(t, p), s) \in \theta) \Longrightarrow$$
$$\Longrightarrow (\exists p \in X^*)((\delta(t, p), s) \in \sigma) \Longrightarrow (\exists p, q \in X^*)(\delta(t, pq) = s) \Longrightarrow (t, s) \in \tau \Longrightarrow$$
$$\Longrightarrow \sigma(s) \leqslant \sigma(t) \Longrightarrow \varphi(\sigma(\theta(s))) \leqslant \varphi(\sigma(\theta(t)));$$

и обратную изотонность:

$$\varphi(\sigma(\theta(s))) \leqslant \varphi(\sigma(\theta(t))) \Longrightarrow \sigma(s) \leqslant \sigma(t) \Longrightarrow (t, s) \in \tau \Longrightarrow$$
$$\Longrightarrow ((\theta(t), \theta(s)) \in \tau \Longrightarrow \sigma(\theta(s)) \leqslant \sigma(\theta(t)).$$

Таким образом, φ — изоморфизм каркаса $F(\mathbf{A}/\theta)$ на каркас $F(\mathbf{A})$.

II. Предположим, что $F(\mathbf{A})/\theta) \cong F(\mathbf{A})$, и докажем, что $\theta \subseteq \sigma$.

От противного. Пусть θ не содержится в σ. Тогда существует пара $(s, t) \in \theta$, где s и t лежат в разных слоях автомата $\mathbf{A}$. Здесь могут представиться два случая.

1. Предположим, что слои $\sigma(s)$ и $\sigma(t)$ сравнимы в каркасе $F(\mathbf{A})$, например, $\sigma(s) < \sigma(t)$.

Из всех таких пар слоев выберем одну с тем свойством, что в ней $\sigma(s)$ имеет наименьшую возможную высоту. Тогда каждое состояние из слоя $\sigma(t)$ находится в одном θ-классе с некоторым состоянием из $\sigma(s)$. Действительно, если $t' \in \sigma(t)$, то существует слово $p \in X^*$, такое, что $\delta(t, p) = t'$. Так как $(s, t) \in \theta$, то $(\delta(s, p), t') \in \theta$. По выбору слоя $\sigma(s)$ имеем $\delta(s, p) \in \sigma(s)$. Точно так же любое состояние из каждого промежуточного слоя $\sigma(u)$, т. е. когда имеют место неравенства $\sigma(s) < \sigma(u) < \sigma(t)$, находится в одном θ-классе с подходящим состоянием из $\sigma(s)$. Отсюда следует, что при факторизации автомата $\mathbf{A}$ по конгруэнции θ все слои из интервала $[\sigma(s), \sigma(t)]$ отождествятся, и значит, в каркасе $F(\mathbf{A}/\theta)$ будет меньше элементов, чем в изоморфном ему каркасе $F(\mathbf{A})$, что невозможно.

2. Предположим, что слои $\sigma(s)$ и $\sigma(t)$ несравнимы в $F(\mathbf{A})$.

Допустим, что найдется слово $p \in X^*$, такое, что $s' = \delta(s, p) \in \sigma(s)$, но $t' = \delta(t, p) \notin \notin \sigma(t)$. Возьмем слово $q \in X^*$ со свойством $\delta(s', q) = s$. Тогда по модулю θ имеем $t \equiv s = \delta(s', q) \equiv \delta(t', q) \notin \sigma(t)$, и мы получаем пару состояний $(t, \delta(t', q)) \in \theta$, где $\sigma(\delta(t', q)) < \sigma(t)$, что приводит к рассмотренному случаю 1.

Пусть, напротив, оказалось, что любое слово $p \in X^*$, переводящее s в одно из состояний слоя $\sigma(s)$, переводит t в одно из состояний слоя $\sigma(t)$. Тогда каждому $s' \in \sigma(s)$ соответствует $t' \in \sigma(t)$, такое, что $(s', t') \in \theta$. Следовательно, при факторизации автомата $\mathbf{A}$ по конгруэнции θ слой $\sigma(s)$ отождествится со слоем $\sigma(t)$, и значит, в каркасе $F(\mathbf{A}/\theta)$ будет меньше элементов, чем в $F(\mathbf{A})$, что невозможно.

Полученные противоречия приводят к выводу о том, что $\theta \subseteq \sigma$. ■

ЛИТЕРАТУРА

1. *Салий В. Н.* Автоматы, у которых все конгруэнции — внутренние // Изв. вузов. Математика. 2009. № 9. С. 36–45.
2. *Богомолов А. М., Салий В. Н.* Алгебраические основы теории дискретных систем. М.: Наука, 1997.

DOI 10.17223/20710410/7/6

УДК 518.6+681.3

АНАЛИЗ НЕЛИНЕЙНЫХ АВТОМАТОВ С ЛАГОМ 2 НАД КОНЕЧНЫМ КОЛЬЦОМ

В. В. Скобелев, В. Г. Скобелев

Институт прикладной математики и механики НАН Украины, г. Донецк, Украина

E-mail: skbv@iamm.ac.donetsk.ua

Для обратимых нелинейных одномерных автоматов с лагом 2 над кольцом $\boldsymbol{Z}_{p^k} = (\mathbb{Z}_{p^k}, \oplus, \circ)$ исследована структура автоматного графа, охарактеризованы множества эквивалентных состояний, решены задачи параметрической идентификации и идентификации начального состояния, охарактеризованы множества неподвижных точек отображений, реализуемых инициальными автоматами.

Ключевые слова: *нелинейные автоматы, конечные кольца, симметричные поточные шифры, системы уравнений над конечными кольцами.*

Введение

В [1, 2] показано, что как с позиции теории автоматов, так и с позиции современной криптографии представляет интерес исследование нелинейных обратимых автоматов над кольцом $\boldsymbol{Z}_{p^k} = (\mathbb{Z}_{p^k}, \oplus, \circ)$, где p — простое число, а операции в кольце определены равенствами $a \oplus b = a + b \mod p^k$ и $a \circ b = a \cdot b \mod p^k$.

Замечание 1. С позиции криптографии интерес исследования обратимых автоматов над конечным кольцом обосновывается тем, что любой такой автомат может рассматриваться в качестве математической модели симметричного поточного шифра.

При фиксации параметров $a, b, c, d, e \in \mathbb{Z}_{p^k}$ система уравнений

$$\begin{cases} q_{t+2} = a \oplus b \circ q_{t+1}^2 \oplus c \circ q_t \oplus d \circ x_{t+1}, \\ y_{t+1} = e \circ q_{t+2}, \end{cases} \quad (t \in \mathbb{Z}_+), \tag{1}$$

определяет автомат Мура [3] M над кольцом $\boldsymbol{Z}_{p^k}$, для которого переменная x_{t+1} — входной символ в момент $t+1$, переменная y_{t+1} — выходной символ в момент $t+1$, а переменная $\mathbf{q}_t = (q_{t+1}, q_t)$ — состояние автомата M в момент t, $t \in \mathbb{Z}_+$. При этом фиксация в автомате M начального состояния $\mathbf{q}_0 = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ приводит к инициальному автомату Мура $(M, \mathbf{q}_0)$, осуществляющему отображение входной полугруппы $\mathbb{Z}_{p^k}^+$ в себя.

Замечание 2. В терминологии, принятой в [4], автомат M, определяемый системой уравнений (1), представляет собой автомат с задержкой.

Целью настоящей работы является исследование множества M всех таких автоматов M, определенных системой уравнений (1), что параметры a, b, c, d, e удовлетворяют условиям: $a, c \in \mathbb{Z}_{p^k}$, $b \in \mathbb{Z}_{p^k} \backslash \{0\}$, $d, e \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, где $\mathbb{Z}_{p^k}^{\mathrm{inv}}$ — множество всех обратимых элементов кольца $\boldsymbol{Z}_{p^k}$.

С алгебраической точки зрения инициальный автомат $(M, \mathbf{q}_0)$ $(M \in \mathsf{M},\ \mathbf{q}_0 \in \mathbb{Z}_{p^k}^2)$ на каждом такте своего функционирования осуществляет фиксированное преобразование (т. е. сюръективное отображение на себя) множества $\mathbb{Z}_{p^k}$, представляющее собой

фиксированное линейное преобразование $u = e \circ v$ линейной комбинации $v = \alpha \oplus \beta$ фиксированной квадратичной формы $\alpha = b \circ q_{t+1}^2 \oplus c \circ q_t$ текущего состояния автомата M и фиксированного аффинного преобразования $\beta = a \oplus d \circ x_{t+1}$ множества $\mathbb{Z}_{p^k}$.

Замечание 3. Первое уравнение системы уравнений (1) представляет собой аналог над кольцом $\boldsymbol{Z}_{p^k}$ ряда модельных хаотических отображений, в частности отображения Эно [5]. Отсюда вытекает, что рассмотренный в [1, 2, 6] автомат Эно представляет собой специальный случай автомата (1).

Автоматом, обратным автомату $M \in \mathsf{M}$, является автомат Мили

$$\begin{cases} q_{t+2} = e^{-1} \circ x_{t+1}, \\ y_{t+1} = d^{-1} \circ (e^{-1} \circ x_{t+1} \ominus a \ominus b \circ q_{t+1}^2 \ominus c \circ q_t), \end{cases} \quad (t \in \mathbb{Z}_+), \tag{2}$$

где $\ominus$ — операция, обратная операции $\oplus$.

Упорядоченная пара

$$((M, \mathbf{q}_0), (M^{\mathrm{inv}}, \mathbf{q}_0)) \ (M \in \mathsf{M},\ \mathbf{q}_0 \in \mathbb{Z}_{p^k}^2) \tag{3}$$

определяет симметричный поточный шифр, секретным ключом которого является упорядоченный набор $(a, b, c, d, e, \mathbf{q}_0) \in \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k} \backslash \{0\}) \times \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k}^{\mathrm{inv}})^2 \times \mathbb{Z}_{p^k}^2$, причем параметры a, b, c, d, e — секретный ключ средней длительности, а начальное состояние $\mathbf{q}_0 = (q_1, q_0)$ — секретный сеансовый ключ. Таким образом, для симметричного поточного шифра (3):

1) длина в битах секретного ключа равна $7\lceil k \log p \rceil$, из которых $5\lceil k \log p \rceil$ бит представляют секретный ключ средней длительности, а $2\lceil k \log p \rceil$ бит — секретный сеансовый ключ;

2) мощность множества секретных ключей равна $p^{6k}(p^k - 1)(1 - p^{-1})^2$, мощность множества секретных ключей средней длительности равна $p^{4k}(p^k - 1)(1 - p^{-1})^2$, а при каждом фиксированном секретном ключе средней длительности мощность множества секретных сеансовых ключей равна p^{2k}.

Замечание 4. Симметричный поточный шифр (3) обладает тем свойством, что в процессе *шифрование — расшифрование* автоматы M и M^{inv} движутся в пространстве состояний по одной и той же траектории в одном и том же направлении.

1. Структура автоматного графа исследуемой модели

В соответствии с [4, 7] считаем, что автоматный граф G_M автомата Мура $M \in \mathsf{M}$ — это размеченный орграф с петлями [8], содержащий p^{2k} вершин. Вершины графа G_M отмечены парами $(\mathbf{q}, y)$, где $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$; $y = e \circ q_1$ — выходной символ, вырабатываемый автоматом M при переходе в состояние $\mathbf{q}$ (т. е. имеется взаимно-однозначное соответствие между вершинами графа G_M и состояниями автомата M). Из вершины с отметкой $(\mathbf{q}, y)$ $(\mathbf{q} = (q_1, q_0))$ в вершину с отметкой $(\widetilde{\mathbf{q}}, \widetilde{y})$ $(\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0))$ идет дуга тогда и только тогда, когда $\widetilde{q}_0 = q_1$ и существует такой входной символ $x \in \mathbb{Z}_{p^k}$, что $\widetilde{q}_1 = a \oplus b \circ q_1^2 \oplus c \circ q_0 \oplus d \circ x$. Эта дуга отмечена множеством всех входных символов, при которых автомат M переходит из состояния $\mathbf{q}$ в состояние $\widetilde{\mathbf{q}}$.

Замечание 5. Так как $M \in \mathsf{M}$ — обратимый автомат Мура, то каждая дуга автоматного графа G_M отмечена единственным входным символом $x \in \mathbb{Z}_{p^k}$.

Охарактеризуем структуру автоматного графа G_M $(M \in \mathsf{M})$.

Теорема 1. Любой автомат $M \in \mathsf{M}$ является сильно связным автоматом, причем диаметр его автоматного графа G_M равен 2.

Доказательство. Пусть $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ и $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ — любые состояния автомата $M \in \mathsf{M}$.

Входной символ

$$x_1 = d^{-1} \circ (\widetilde{q}_0 \ominus a \ominus b \circ q_1^2 \ominus c \circ q_0) \tag{4}$$

переводит состояние $\mathbf{q}$ автомата $M \in \mathsf{M}$ в состояние $\mathbf{q}_1 = (\widetilde{q}_0, q_1) \in \mathbb{Z}_{p^k}^2$, а входной символ

$$x_2 = d^{-1} \circ (\widetilde{q}_1 \ominus a \ominus b \circ \widetilde{q}_0^2 \ominus c \circ q_1) \tag{5}$$

переводит состояние $\mathbf{q}_1$ автомата $M \in \mathsf{M}$ в состояние $\mathbf{q}_2 = \widetilde{\mathbf{q}}$. Следовательно, входное слово x_1x_2, определяемое равенствами (4) и (5), переводит любое состояние $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ автомата $M \in \mathsf{M}$ в любое состояние $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$, т. е. $M \in \mathsf{M}$ — сильно связный автомат, что и требовалось доказать.

Так как входное слово x_1x_2, определяемое равенствами (4) и (5), переводит любое состояние $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ автомата $M \in \mathsf{M}$ в любое состояние $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$, то в автоматном графе G_M ($M \in \mathsf{M}$) расстояние между любыми двумя вершинами не превосходит 2.

Пусть $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ и $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ — такие два состояния автомата $M \in \mathsf{M}$, что $q_1 \neq \widetilde{q}_0$.

Входной символ $x_1 \in \mathbb{Z}_{p^k}$ переводит состояние $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ автомата M в состояние $\mathbf{q}_1 = (q_2, q_1) \in \mathbb{Z}_{p^k}^2$, где

$$q_2 = a \oplus b \circ q_1^2 \oplus c \circ q_0 \oplus d \circ x_1.$$

Так как $q_1 \neq \widetilde{q}_0$, то $\mathbf{q}_1 \neq \widetilde{\mathbf{q}}$ для любого входного символа $x_1 \in \mathbb{Z}_{p^k}$. Следовательно, входное слово $x_1x_2 \in \mathbb{Z}_{p^k}^2$, определяемое равенствами (4) и (5), является минимальным входным словом, переводящим состояние $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ автомата M в состояние $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ ($q_1 \neq \widetilde{q}_0$). Это означает, что в автоматном графе G_M ($M \in \mathsf{M}$) расстояние от вершины с отметкой $(\mathbf{q}, y)$ ($\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2, y = e \circ q_1$) до вершины с отметкой $(\widetilde{\mathbf{q}}, \widetilde{y})$ ($\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2, \widetilde{y} = e \circ \widetilde{q}_1$) равно 2, если $q_1 \neq \widetilde{q}_0$.

Отсюда вытекает, что диаметр автоматного графа G_M ($M \in \mathsf{M}$) равен 2, что и требовалось доказать. ■

Следствие 1. В автоматном графе G_M ($M \in \mathsf{M}$) существует петля в вершине с отметкой $(\mathbf{q}, y)$ ($\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2, y = e \circ q_1$) тогда и только тогда, когда $q_1 = q_0$. Эта петля отмечена входным символом

$$x = d^{-1} \circ (q_0 \ominus a \ominus b \circ q_0^2 \ominus c \circ q_0). \tag{6}$$

Доказательство. Автомат $M \in \mathsf{M}$ из состояния $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ под действием входного символа $x \in \mathbb{Z}_{p^k}$ переходит в состояние $\mathbf{q}_1 = (q_2, q_1)$, где

$$q_2 = a \oplus b \circ q_1^2 \oplus c \circ q_0 \oplus d \circ x. \tag{7}$$

Равенство $\mathbf{q} = \mathbf{q}_1$ истинно тогда и только тогда, когда $q_1 = q_0$ и $q_2 = q_1$ $(= q_0)$.

Положив $q_1 = q_0$ и $q_2 = q_0$ в равенстве (7), получим, что единственный входной символ $x \in \mathbb{Z}_{p^k}$, отмечающий петлю в вершине отметкой $((q_0, q_0), e \circ q_0)$ ($q_0 \in \mathbb{Z}_{p^k}$), определяется равенством (6). ■

Из доказательства теоремы 1 и из следствия 1 непосредственно вытекает, что истинны следующие два следствия.

Следствие 2. В автоматном графе G_M ($M \in \mathsf{M}$) множество вершин, находящихся на расстоянии 1 от вершины с отметкой $(\mathbf{q}, y)$ ($\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2,\ y = e \circ q_1$), состоит из $p^k - 1$ вершин, отметки которых имеют вид $((q_2, q_1), e \circ q_2)$ ($q_2 \in \mathbb{Z}_{p^k} \backslash \{q_0\}$), если $q_1 = q_0$, и из p^k вершин, отметки которых имеют вид $((q_2, q_1), e \circ q_2)$ ($q_2 \in \mathbb{Z}_{p^k}$), если $q_1 \neq q_0$.

Следствие 3. В автоматном графе G_M ($M \in \mathsf{M}$) множество вершин, от которых вершина с отметкой $(\mathbf{q}, y)$ ($\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2,\ y = e \circ q_1$) находится на расстоянии 1, состоит из $p^k - 1$ вершин, отметки которых имеют вид $((q_0, \widetilde{q}_0), e \circ q_0)$ ($\widetilde{q}_0 \in \mathbb{Z}_{p^k} \backslash \{q_0\}$), если $q_1 = q_0$, и из p^k вершин, отметки которых имеют вид $((q_0, \widetilde{q}_0), e \circ q_0)$ ($\widetilde{q}_0 \in \mathbb{Z}_{p^k}$), если $q_1 \neq q_0$.

Теорема 2. Для любых двух фиксированных вершин автоматного графа G_M ($M \in \mathsf{M}$) либо исходящие из них дуги, отмеченные любым входным символом $x \in \mathbb{Z}_{p^k}$, ведут в различные вершины графа G_M, либо исходящие из них дуги, отмеченные любым входным символом $x \in \mathbb{Z}_{p^k}$, ведут в одну и ту же вершину графа G_M.

Доказательство. Предположим, что в автоматном графе G_M ($M \in \mathsf{M}$) дуги, отмеченные входным символом $\widetilde{x}$ ($\widetilde{x} \in \mathbb{Z}_{p^k}$), ведут из двух фиксированных вершин с отметками соответственно $(\mathbf{q}^{(1)}, y^{(1)})$ ($\mathbf{q}^{(1)} = (q_1^{(1)}, q_0^{(1)}) \in \mathbb{Z}_{p^k}^2,\ y^{(1)} = e \circ q_1^{(1)}$) и $(\mathbf{q}^{(2)}, y^{(2)})$ ($\mathbf{q}^{(2)} = (q_1^{(2)}, q_0^{(2)}) \in \mathbb{Z}_{p^k}^2,\ y^{(2)} = e \circ q_1^{(2)}$) в одну и ту же вершину с отметкой $(\widetilde{\mathbf{q}}, \widetilde{y})$ ($\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2,\ \widetilde{y} = e \circ \widetilde{q}_1$). Тогда $q_2^{(1)} = q_2^{(1)}$ ($= \widetilde{q}_1$), т. е.

$$a \oplus b \circ (q_1^{(1)})^2 \oplus c \circ q_0^{(1)} \oplus d \circ \widetilde{x} = a \oplus b \circ (q_1^{(2)})^2 \oplus c \circ q_0^{(2)} \oplus d \circ \widetilde{x},$$

и $q_1^{(1)} = q_1^{(2)}$ ($= \widetilde{q}_0$), т. е. $\mathbf{q}^{(1)} = (\widetilde{q}_0, q_0^{(1)})$ и $\mathbf{q}^{(2)} = (\widetilde{q}_0, q_0^{(2)})$.

Следовательно,

$$a \oplus b \circ \widetilde{q}_0^2 \oplus c \circ q_0^{(1)} \oplus d \circ \widetilde{x} = a \oplus b \circ \widetilde{q}_0^2 \oplus c \circ q_0^{(2)} \oplus d \circ \widetilde{x} \Leftrightarrow$$

$$\Leftrightarrow a \oplus b \circ \widetilde{q}_0^2 \oplus c \circ q_0^{(1)} = a \oplus b \circ \widetilde{q}_0^2 \oplus c \circ q_0^{(2)} \Leftrightarrow$$

$$\Leftrightarrow (\forall x \in \mathbb{Z}_{p^k})(a \oplus b \circ \widetilde{q}_0^2 \oplus c \circ q_0^{(1)} \oplus d \circ x = a \oplus b \circ \widetilde{q}_0^2 \oplus c \circ q_0^{(2)} \oplus d \circ x).$$

Последнее утверждение означает, что дуги, отмеченные любым входным символом $x \in \mathbb{Z}_{p^k}$, ведут из вершин с отметками $(\mathbf{q}^{(1)}, y^{(1)})$ и $(\mathbf{q}^{(2)}, y^{(2)})$ в вершину с отметкой $(\mathbf{q}, y)$ ($\mathbf{q} = (q_1, \widetilde{q}_0),\ y = e \circ q_1$), где

$$q_1 = a \oplus b \circ \widetilde{q}_0^2 \oplus c \circ q_0^{(1)} \oplus d \circ x\ (= a \oplus b \circ \widetilde{q}_0^2 \oplus c \circ q_0^{(2)} \oplus d \circ x)),$$

что и требовалось доказать. ■

2. Критерий эквивалентности состояний исследуемой модели

Два различных состояния автомата называются близнецами [9], если по любому входному символу они переходят в одно и то же состояние. Из теоремы 2 вытекает, что истинно следующее следствие.

Следствие 4. В любом автомате $M \in \mathsf{M}$ любые два состояния $\mathbf{q}, \widetilde{\mathbf{q}} \in \mathbb{Z}_{p^k}^2$ ($\mathbf{q} \neq \widetilde{\mathbf{q}}$) либо являются близнецами, либо по любому входному символу $x \in \mathbb{Z}_{p^k}$ переходят в различные состояния автомата M.

Так как $M \in \mathsf{M}$ — автомат Мура, то эквивалентны любые состояния-близнецы $\mathbf{q}, \widetilde{\mathbf{q}} \in \mathbb{Z}_{p^k}^2$ автомата $M \in \mathsf{M}$. Следующая теорема показывает, что понятие «состояния-близнецы» дает возможность полностью охарактеризовать эквивалентные состояния автомата $M \in \mathsf{M}$.

Теорема 3. Состояния $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ и $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ $(\mathbf{q} \neq \widetilde{\mathbf{q}})$ автомата $M \in \mathsf{M}$ эквивалентны тогда и только тогда, когда либо $q_1 = \widetilde{q}_1$, а состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ — близнецы, либо когда $q_1 \neq \widetilde{q}_1$, а состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ по любому входному символу переходят в состояния-близнецы.

Доказательство. Пусть $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ и $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ $(\mathbf{q} \neq \widetilde{\mathbf{q}})$ — эквивалентные состояния автомата $M \in \mathsf{M}$.

Под действием входного символа $x_1 \in \mathbb{Z}_{p^k}$ состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ автомата M переходят соответственно в состояния $\mathbf{q}_1 = (q_2, q_1)$ и $\widetilde{\mathbf{q}}_1 = (\widetilde{q}_2, \widetilde{q}_1)$, где

$$q_2 = a \oplus b \circ q_1^2 \oplus c \circ q_0 \oplus d \circ x_1, \tag{8}$$

$$\widetilde{q}_2 = a \oplus b \circ \widetilde{q}_1^2 \oplus c \circ \widetilde{q}_0 \oplus d \circ x_1. \tag{9}$$

Так как $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ — эквивалентные состояния автомата M, то $y_1 = \widetilde{y}_1$ для любого входного символа $x_1 \in \mathbb{Z}_{p^k}$. Следовательно,

$$y_1 = \widetilde{y}_1 \Leftrightarrow e \circ \widetilde{q}_2 = e \circ q_2 \Leftrightarrow \widetilde{q}_2 = q_2 \tag{10}$$

для любого входного символа $x_1 \in \mathbb{Z}_{p^k}$.

Из (10) вытекает, что для любого входного символа $x_1 \in \mathbb{Z}_{p^k}$ истинно равенство $\mathbf{q}_1 = \widetilde{\mathbf{q}}_1$ тогда и только тогда, когда $\widetilde{q}_1 = q_1$ и, кроме того, для любого входного символа $x_1 \in \mathbb{Z}_{p^k}$ истинно равенство $\widetilde{q}_2 = q_2$.

Таким образом, $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ — такие эквивалентные состояния автомата $M \in \mathsf{M}$, что $\widetilde{q}_1 = q_1$ тогда и только тогда, когда $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ — близнецы, что и требовалось доказать.

Пусть $\widetilde{q}_1 \neq q_1$. Тогда $\mathbf{q}_1$ и $\widetilde{\mathbf{q}}_1$ — такие эквивалентные состояния автомата $M \in \mathsf{M}$, что $\mathbf{q}_1 \neq \widetilde{\mathbf{q}}_1$. Из равенства $\widetilde{q}_2 = q_2$ вытекает, что под действием входного символа $x_2 \in \mathbb{Z}_{p^k}$ состояния $\mathbf{q}_1$ и $\widetilde{\mathbf{q}}_1$ автомата M переходят соответственно в состояния $\mathbf{q}_2 = (q_3, q_2)$ и $\widetilde{\mathbf{q}}_2 = (\widetilde{q}_3, q_2)$, где

$$q_3 = a \oplus b \circ q_2^2 \oplus c \circ q_1 \oplus d \circ x_2, \tag{11}$$

$$\widetilde{q}_3 = a \oplus b \circ q_2^2 \oplus c \circ \widetilde{q}_1 \oplus d \circ x_2. \tag{12}$$

Так как $\mathbf{q}_1$ и $\widetilde{\mathbf{q}}_1$ — эквивалентные состояния автомата M, то $y_2 = \widetilde{y}_2$ для любого входного символа $x_2 \in \mathbb{Z}_{p^k}$. Следовательно,

$$y_2 = \widetilde{y}_2 \Leftrightarrow e \circ \widetilde{q}_3 = e \circ q_3 \Leftrightarrow \widetilde{q}_3 = q_3 \tag{13}$$

для любого входного символа $x_2 \in \mathbb{Z}_{p^k}$.

Из равенства $\widetilde{q}_2 = q_2$ и из (13) вытекает, что $\mathbf{q}_2 = \widetilde{\mathbf{q}}_2$ для любого входного символа $x_2 \in \mathbb{Z}_{p^k}$, т. е. что состояния $\mathbf{q}_1$ и $\widetilde{\mathbf{q}}_1$ — близнецы, что и требовалось доказать. ■

Из доказательства теоремы 3 вытекает, что истинно следующее следствие.

Следствие 5. Если $\mathbf{q}, \widetilde{\mathbf{q}} \in \mathbb{Z}_{p^k}^2$ $(\mathbf{q} \neq \widetilde{\mathbf{q}})$ — неэквивалентные состояния автомата $M \in \mathsf{M}$, то существует входное слово $x_1 x_2 \in \mathbb{Z}_{p^k}^2$, различающее состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$.

Следствие 6. Состояния $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ и $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ $(\mathbf{q} \neq \widetilde{\mathbf{q}})$ автомата $M \in \mathsf{M}$ эквивалентны тогда и только тогда, когда либо

$$\begin{cases} \widetilde{q}_1 = q_1, \\ c \circ (\widetilde{q}_0 \ominus q_0) = 0, \end{cases} \tag{14}$$

либо

$$\begin{cases} \widetilde{q}_1 \neq q_1, \\ b \circ (\widetilde{q}_1^2 \ominus q_1^2) \oplus c \circ (\widetilde{q}_0 \ominus q_0) = 0, \\ c \circ (\widetilde{q}_1 \ominus q_1) = 0. \end{cases} \tag{15}$$

Доказательство. Пусть эквивалентные состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ — близнецы. Тогда $\widetilde{q}_1 = q_1$, а из (10) вытекает, что $\widetilde{q}_2 = q_2$.

Подставив $\widetilde{q}_1 = q_1$ и $\widetilde{q}_2 = q_2$ в равенства (8) и (9) и вычитая равенство (9) из равенства (8), получим

$$c \circ (\widetilde{q}_0 \ominus q_0) = 0,$$

что и требовалось доказать.

Пусть эквивалентные состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ не являются близнецами. Тогда $\widetilde{q}_1 \neq q_1$ и, в силу (10), $\widetilde{q}_2 = q_2$.

Подставив $\widetilde{q}_2 = q_2$ в равенства (8) и (9) и вычитая равенство (9) из равенства (8), получим

$$b \circ (\widetilde{q}_1^2 \ominus q_1^2) \oplus c \circ (\widetilde{q}_0 \ominus q_0) = 0.$$

Из (13) вытекает, что $\widetilde{q}_3 = q_3$. Подставив $\widetilde{q}_3 = q_3$ в равенства (11) и (12) и вычитая равенство (12) из равенства (11), получим

$$c \circ (\widetilde{q}_1 \ominus q_1) = 0,$$

что и требовалось доказать. ■

3. Вспомогательные результаты

Положив в равенствах (14)

$$\widetilde{q}_0 \ominus q_0 = u, \tag{16}$$

а в равенствах (15)

$$\begin{cases} \widetilde{q}_1 \ominus q_1 = u, \\ \widetilde{q}_1 \oplus q_1 = v, \\ \widetilde{q}_0 \ominus q_0 = w, \end{cases} \tag{17}$$

получим, что для того чтобы охарактеризовать структуру множества состояний, эквивалентных фиксированному состоянию $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ автомата $M \in \mathsf{M}$, достаточно охарактеризовать структуру множества S_1 решений u уравнения

$$c \circ u = 0 \tag{18}$$

и структуру множества S_2 решений (u, v, w) системы уравнений

$$\begin{cases} b \circ u \circ v \oplus c \circ w = 0, \\ c \circ u = 0. \end{cases} \tag{19}$$

В соответствии с [1] определим p-тип $\mathtt{t}_p(z)$ элемента $z \in \mathbb{Z}_{p^k}$ равенством

$$\mathtt{t}_p(z) = \begin{cases} k, & \text{если } z = 0; \\ 0, & \text{если } z \in \mathbb{Z}_{p^k}^{\mathrm{inv}}; \\ r \ (1 \leqslant r \leqslant k-1), & \text{если } z \equiv 0 \pmod{p^r} \text{ и } z \not\equiv 0 \pmod{p^{r+1}}. \end{cases} \tag{20}$$

Замечание 6. Из (20) вытекает, что множество всех элементов кольца $\boldsymbol{Z}_{p^k}$, p-тип которых равен r $(0 \leqslant r \leqslant k-1)$, представляет собой множество всех тех элементов $z \in \mathbb{Z}_{p^k}$, которые единственным образом представимы в виде $z = \alpha \circ p^r$, где $\alpha \in \mathbb{Z}_{p^{k-r}}^{\mathrm{inv}}$.

Отсюда, в частности, вытекает, что для любых $u, v \in \mathbb{Z}_{p^k}$

$$\mathtt{t}_p(u \circ v) = \min\{k, \mathtt{t}_p(u) + \mathtt{t}_p(v)\},$$
$$\mathtt{t}_p(u \oplus v) \geqslant \min\{\mathtt{t}_p(u), \mathtt{t}_p(v)\},$$

причем

$$\mathtt{t}_p(u \oplus v) = 0,$$

если $\mathtt{t}_p(u) > 0$ и $\mathtt{t}_p(v) = 0$.

Так как $M \in \mathsf{M}$, то

$$\mathtt{t}_p(b) \in \{0, 1, \ldots, k-1\},$$
$$\mathtt{t}_p(c) \in \{0, 1, \ldots, k\}.$$

Исследуем структуру множеств S_1 и S_2 при этих предположениях.

Случай 1. Пусть $\mathtt{t}_p(c) = 0$, т. е. $c \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$.

Для уравнения (18) получим

$$c \circ u = 0 \Leftrightarrow u = 0 \Leftrightarrow S_1 = \{0\}. \tag{21}$$

Для системы уравнений (19) получим

$$\begin{cases} b \circ u \circ v \oplus c \circ w = 0, \\ c \circ u = 0, \end{cases} \quad \Leftrightarrow \begin{cases} u = 0, \\ v \in \mathbb{Z}_{p^k}, \\ c \circ w = 0, \end{cases} \quad \Leftrightarrow$$
$$\Leftrightarrow \begin{cases} u = 0, \\ v \in \mathbb{Z}_{p^k}, \\ w = 0, \end{cases} \quad \Leftrightarrow S_2 = \{(0, v, 0) | v \in \mathbb{Z}_{p^k}\}. \tag{22}$$

Случай 2. Пусть $\mathtt{t}_p(c) = k$, т. е. $c = 0$.

Для уравнения (18) получим

$$0 \circ u = 0 \Leftrightarrow S_1 = \mathbb{Z}_{p^k}. \tag{23}$$

Для системы уравнений (19) получим

$$\begin{cases} b \circ u \circ v \oplus 0 \circ w = 0, \\ 0 \circ u = 0, \end{cases} \quad \Leftrightarrow \begin{cases} u \in \mathbb{Z}_{p^k}, \\ b \circ u \circ v = 0, \\ w \in \mathbb{Z}_{p^k}, \end{cases} \quad \Leftrightarrow$$

$$\Leftrightarrow S_2 = \{(u, v, w) \in \mathbb{Z}_{p^k}^3 | \mathtt{t}_p(v) \geqslant k - \mathtt{t}_p(b) - \mathtt{t}_p(u)\}. \tag{24}$$

Случай 3. Пусть $\mathtt{t}_p(c) \in \{1, \ldots, k-1\}$, т. е. $c = \alpha \circ p^{\mathtt{t}_p(c)}$, где $\alpha \in \mathbb{Z}_{p^{k-\mathtt{t}_p(c)}}^{\mathrm{inv}}$.

Для уравнения (18) получим

$$\alpha \circ p^{\mathtt{t}_p(c)} \circ u = 0 \Leftrightarrow p^{\mathtt{t}_p(c)} \circ u = 0 \Leftrightarrow$$

$$\Leftrightarrow S_1 = \{0\} \cup \bigcup_{l_1=k-\mathtt{t}_p(c)}^{k-1} \{\delta_1 \circ p^{l_1} | \delta_1 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_1}}\}. \tag{25}$$

Рассмотрим систему уравнений (19). Из (25) вытекает, что

$$S_2 = S_2^{'} \cup S_2^{''}, \tag{26}$$

где $S_2^{'}$ — множество решений системы уравнений (19) при $u = 0$, а $S_2^{''}$ — множество решений системы уравнений (19) при условии, что

$$u \in \bigcup_{l_1=k-\mathtt{t}_p(c)}^{k-1} \{\delta_1 \circ p^{l_1} | \delta_1 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_1}}\}.$$

Отметим, что

$$S_2^{'} \cap S_2^{''} = \varnothing.$$

Найдем множество $S_2^{'}$:

$$\begin{cases} b \circ u \circ v \oplus \alpha \circ p^{\mathtt{t}_p(c)} \circ w = 0, \\ \alpha \circ p^{\mathtt{t}_p(c)} \circ u = 0, \\ u = 0, \end{cases} \Leftrightarrow \begin{cases} u = 0, \\ v \in \mathbb{Z}_{p^k}, \\ \alpha \circ p^{\mathtt{t}_p(c)} \circ w = 0, \end{cases} \Leftrightarrow$$

$$\Leftrightarrow S_2^{'} = \{(0, v, w) | v, w \in \mathbb{Z}_{p^k},\ \mathtt{t}_p(w) \geqslant k - \mathtt{t}_p(c)\}. \tag{27}$$

Найдем множество $S_2^{''}$.

Положим

$$S_2^{''}(\delta_1 \circ p^{l_1}) = \{(u, v, w) \in S_2 | u = \delta_1 \circ p^{l_1}\ (l_1 \in \{k - \mathtt{t}_p(c), \ldots, k-1\};\ \delta_1 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_1}})\}.$$

Так как множества $S_2^{''}(\delta_1 \circ p^{l_1})$ $(l_1 \in \{k - \mathtt{t}_p(c), \ldots, k-1\};\ \delta_1 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_1}})$ попарно не пересекаются, причем

$$S_2^{''} = \bigcup_{l_1=k-\mathtt{t}_p(c)}^{k-1} \bigcup_{\delta_1 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_1}}} S_2^{''}(\delta_1 \circ p^{l_1}), \tag{28}$$

то для того, чтобы найти множество $S_2^{''}$, необходимо и достаточно найти множества $S_2^{''}(\delta_1 \circ p^{l_1})$ $(l_1 \in \{k - \mathtt{t}_p(c), \ldots, k-1\};\ \delta_1 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_1}})$.

Найдем эти множества.

Так как

$$\begin{cases} b \circ u \circ v \oplus \alpha \circ p^{\mathtt{t}_p(c)} \circ w = 0, \\ u = \delta_1 \circ p^{l_1}, \end{cases} \Leftrightarrow \begin{cases} u = \delta_1 \circ p^{l_1}, \\ b \circ \delta_1 \circ p^{l_1} \circ v \oplus \alpha \circ p^{\mathtt{t}_p(c)} \circ w = 0 \end{cases} \tag{29}$$

и $\mathtt{t}_p(b) \in \{0, 1, \ldots, k-1\}$, то $b = \beta \circ p^{\mathtt{t}_p(b)}$, где $\beta \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-\mathtt{t}_p(b)}}$.

Следовательно, система уравнений (29) имеет вид

$$\begin{cases} u = \delta_1 \circ p^{l_1}, \\ (\beta \circ \delta_1) \circ p^{\mathtt{t}_p(b)+l_1} \circ v \oplus \alpha \circ p^{\mathtt{t}_p(c)} \circ w = 0. \end{cases} \tag{30}$$

Возможны следующие три случая.

Случай 3.1. Пусть $l_1 < \mathtt{t}_p(c) - \mathtt{t}_p(b)$.

Замечание 7. Так как $l_1 \geqslant k - \mathtt{t}_p(c)$, то случай 3.1 имеет место тогда и только тогда, когда $\mathtt{t}_p(c) > 0{,}5 \cdot (k + \mathtt{t}_p(b))$.

Представим систему уравнений (30) в следующей эквивалентной форме:

$$\begin{cases} u = \delta_1 \circ p^{l_1}, \\ p^{\mathtt{t}_p(b)+l_1} \circ (v \oplus \alpha \circ (\beta \circ \delta_1)^{-1} \circ p^{\mathtt{t}_p(c)-\mathtt{t}_p(b)-l_1} \circ w) = 0. \end{cases} \tag{31}$$

Из второго уравнения системы (31) вытекает, что либо

$$v \oplus \alpha \circ (\beta \circ \delta_1)^{-1} \circ p^{\mathtt{t}_p(c)-\mathtt{t}_p(b)-l_1} \circ w = 0,$$

либо

$$v \oplus \alpha \circ (\beta \circ \delta_1)^{-1} \circ p^{\mathtt{t}_p(c)-\mathtt{t}_p(b)-l_1} \circ w = \delta_2 \circ p^{l_2} \ \ (k - \mathtt{t}_p(b) - l_1 \leqslant l_2 \leqslant k-1, \ \delta_2 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_2}}).$$

Таким образом, если $l_1 < \mathtt{t}_p(c) - \mathtt{t}_p(b)$, то

$$S_2''(\delta_1 \circ p^{l_1}) = \{(\delta_1 \circ p^{l_1}, \ominus\alpha \circ (\beta \circ \delta_1)^{-1} \circ p^{\mathtt{t}_p(c)-\mathtt{t}_p(b)-l_1} \circ w, w) | w \in \mathbb{Z}_{p^k}\} \cup$$

$$\cup \bigcup_{l_2=k-\mathtt{t}_p(b)-l_1}^{k-1} \bigcup_{w \in \mathbb{Z}_{p^k}} \{(\delta_1 \circ p^{l_1}, \delta_2 \circ p^{l_2} \ominus \alpha \circ (\beta \circ \delta_1)^{-1} \circ p^{\mathtt{t}_p(c)-\mathtt{t}_p(b)-l_1} \circ w, w) | \delta_2 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_2}}\}. \tag{32}$$

Случай 3.2. Пусть $l_1 = \mathtt{t}_p(c) - \mathtt{t}_p(b)$.

Замечание 8. Так как $l_1 \geqslant k - \mathtt{t}_p(c)$, то случай 3.2 имеет место тогда и только тогда, когда $\mathtt{t}_p(c) \geqslant 0{,}5(k + \mathtt{t}_p(b))$.

Представим систему уравнений (30) в следующей эквивалентной форме:

$$\begin{cases} u = \delta_1 \circ p^{l_1}, \\ p^{\mathtt{t}_p(c)} \circ (v \oplus \alpha \circ (\beta \circ \delta_1)^{-1} \circ w) = 0. \end{cases} \tag{33}$$

Из второго уравнения системы (33) вытекает, что либо

$$v \oplus \alpha \circ (\beta \circ \delta_1)^{-1} \circ w = 0,$$

либо

$$v \oplus \alpha \circ (\beta \circ \delta_1)^{-1} \circ w = \delta_3 \circ p^{l_3} \ \ (k - \mathtt{t}_p(c) \leqslant l_3 \leqslant k-1, \ \delta_3 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_3}}).$$

Таким образом, если $l_1 = \mathtt{t}_p(c) - \mathtt{t}_p(b)$, то

$$\begin{gathered} S_2''(\delta_1 \circ p^{l_1}) = \{(\delta_1 \circ p^{l_1}, \ominus\alpha \circ (\beta \circ \delta_1)^{-1} \circ w, w) | w \in \mathbb{Z}_{p^k}\} \cup \\ \cup \bigcup_{l_3=k-\mathtt{t}_p(c)}^{k-1} \bigcup_{w \in \mathbb{Z}_{p^k}} \{(\delta_1 \circ p^{l_1}, \delta_3 \circ p^{l_3} \ominus \alpha \circ (\beta \circ \delta_1)^{-1} \circ w, w) | \delta_3 \in \mathbb{Z}^{\mathrm{inv}}_{p^{k-l_3}}\}. \end{gathered} \tag{34}$$

Случай 3.3. Пусть $\mathtt{t}_p(c) - \mathtt{t}_p(b) < l_1 \leqslant k-1$. Представим систему уравнений (30) в следующей эквивалентной форме:

$$\begin{cases} u = \delta_1 \circ p^{l_1}, \\ p^{\mathtt{t}_p(c)} \circ (\beta \circ \delta_1 \circ \alpha^{-1} \circ p^{\mathtt{t}_p(b)-\mathtt{t}_p(c)+l_1} \circ v \oplus w) = 0. \end{cases} \tag{35}$$

Из второго уравнения системы (35) вытекает, что либо

$$\beta \circ \delta_1 \circ \alpha^{-1} \circ p^{\mathtt{t}_p(b)-\mathtt{t}_p(c)+l_1} \circ v \oplus w = 0,$$

либо

$$\beta \circ \delta_1 \circ \alpha^{-1} \circ p^{\mathtt{t}_p(b)-\mathtt{t}_p(c)+l_1} \circ v \oplus w = \delta_4 \circ p^{l_4} \ \ (k - \mathtt{t}_p(c) \leqslant l_4 \leqslant k-1,\ \delta_4 \in \mathbb{Z}_{p^{k-l_4}}^{\mathrm{inv}}).$$

Таким образом, если $\mathtt{t}_p(c) - \mathtt{t}_p(b) < l_1 \leqslant k-1$, то

$$S_2''(\delta_1 \circ p^{l_1}) = \{(\delta_1 \circ p^{l_1}, v, \ominus\beta \circ \delta_1 \circ \alpha^{-1} \circ p^{\mathtt{t}_p(b)-\mathtt{t}_p(c)+l_1} \circ v) | v \in \mathbb{Z}_{p^k}\} \cup$$
$$\cup \bigcup_{l_4=k-\mathtt{t}_p(c)}^{k-1} \bigcup_{v \in \mathbb{Z}_{p^k}} \{(\delta_1 \circ p^{l_1}, v, \delta_4 \circ p^{l_4} \ominus \beta \circ \delta_1 \circ \alpha^{-1} \circ p^{\mathtt{t}_p(b)-\mathtt{t}_p(c)+l_1} \circ v) | \delta_4 \in \mathbb{Z}_{p^{k-l_4}}^{\mathrm{inv}}\}. \qquad (36)$$

4. Структура множеств эквивалентных состояний исследуемой модели

В соответствии с [10] автомат $M \in \mathsf{M}$ назовем приведенным, если все его состояния попарно неэквивалентные.

Теорема 4. Автомат $M \in \mathsf{M}$ является приведенным автоматом тогда и только тогда, когда $c \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$.

Доказательство. Из следствия 6 вытекает, что состояния $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ и $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ $(\mathbf{q} \neq \widetilde{\mathbf{q}})$ автомата $M \in \mathsf{M}$ являются близнецами тогда и только тогда, когда они удовлетворяют равенствам (14).

Пусть $c \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$. Из (21) вытекает, что уравнение (18) имеет единственное решение $u = 0$. Подставив это значение u в (16), получим, что $\widetilde{q}_0 = q_0$. Отсюда вытекает, что состояния $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ и $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ автомата $M \in \mathsf{M}$ удовлетворяют равенствам (14) тогда и только тогда, когда $\mathbf{q} = \widetilde{\mathbf{q}}$. Следовательно, никакие два различных состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ автомата $M \in \mathsf{M}$ не являются близнецами.

Отсюда (в силу теоремы 3) вытекает, что никакие два различных состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ автомата $M \in \mathsf{M}$ не являются эквивалентными состояниями, т. е. $M \in \mathsf{M}$ — приведенный автомат, что и требовалось доказать.

Пусть $c \notin \mathbb{Z}_{p^k}^{\mathrm{inv}}$. Из (23) и (25) вытекает, что уравнение (18) имеет решение $u_0 \neq 0$. Подставив это решение в (16), получим $\widetilde{q}_0 = q_0 \oplus u_0$. Так как u_0 — решение уравнения (18) и $u_0 \neq 0$, то $\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ и $\widetilde{\mathbf{q}} = ((q_1, q_0 \oplus u_0) \in \mathbb{Z}_{p^k}^2$ — различные состояния автомата $M \in \mathsf{M}$, удовлетворяющие равенствам (14), т. е. состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ являются близнецами.

Отсюда (в силу теоремы 3) вытекает, что $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ $(\mathbf{q} \neq \widetilde{\mathbf{q}})$ — эквивалентные состояния автомата $M \in \mathsf{M}$, т. е. автомат $M \in \mathsf{M}$ не является приведенным автоматом, что и требовалось доказать. ■

Обозначим через M_{NR} множество всех автоматов $M \in \mathsf{M}$, не являющихся приведенными автоматами.

Замечание 9. Так как $|\mathsf{M}| = p^{4k}(p^k - 1)(1 - p^{-1})^2$ и $|\mathsf{M}_{NR}| = p^{4k}(p^k - 1)(1 - p^{-1})^3$, то $|\mathsf{M}_{NR}| \cdot |\mathsf{M}|^{-1} = 1 - p^{-1}$. Значит, $|\mathsf{M}_{NR}| \cdot |\mathsf{M}|^{-1} \xrightarrow[p\to\infty]{} 1$, т. е. при $p \to \infty$ почти все автоматы $M \in \mathsf{M}$ не являются приведенными автоматами.

Обозначим через $C_M(\mathbf{q})$ $(\mathbf{q} = (q_1, q_0) \in \mathbb{Z}_{p^k}^2,\ M \in \mathsf{M}_{NR})$ множество всех состояний автомата M, эквивалентных состоянию $\mathbf{q}$.

Из теоремы 3 вытекает, что множество $C_M(\mathbf{q})$ может быть представлено в виде

$$C_M(\mathbf{q}) = \{\mathbf{q}\} \cup C_M^{(1)}(\mathbf{q}) \cup C_M^{(2)}(\mathbf{q}), \qquad (37)$$

где $C_M^{(1)}(\mathbf{q})$ — множество всех таких состояний $\widetilde{\mathbf{q}} = (q_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ $(q_0 \neq \widetilde{q}_0)$, что состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ — близнецы, а $C_M^{(2)}(\mathbf{q})$ — множество всех таких состояний $\widetilde{\mathbf{q}} = (\widetilde{q}_1, \widetilde{q}_0) \in \mathbb{Z}_{p^k}^2$ $(q_1 \neq \widetilde{q}_1)$, что состояния $\mathbf{q}$ и $\widetilde{\mathbf{q}}$ по любому входному символу переходят в состояния-близнецы.

Из (14), (16), (23) и (25) вытекает, что

$$C_M^{(1)}(\mathbf{q}) = \{\widetilde{\mathbf{q}} = (q_1, q_0 \oplus u) | u \in S_1 \backslash \{0\}\}, \tag{38}$$

а из (15), (17), (24)и (26) вытекает, что

$$C_M^{(2)}(\mathbf{q}) = \begin{cases} \{(\widetilde{q}_1, \widetilde{q}_0) | \widetilde{q}_1 \neq q_1 \ \& \ (\widetilde{q}_1 \ominus q_1, \widetilde{q}_1 \oplus q_1, \widetilde{q}_0 \ominus q_0) \in S_2\}, & \text{если } \mathtt{t}_p(c) = k; \\ \{(\widetilde{q}_1, \widetilde{q}_0) | \widetilde{q}_1 \neq q_1 \ \& \ (\widetilde{q}_1 \ominus q_1, \widetilde{q}_1 \oplus q_1, \widetilde{q}_0 \ominus q_0) \in S_2''\}, & \text{если } 1 \leqslant \mathtt{t}_p(c) \leqslant k-1. \end{cases}$$

Замечание 10. Из (23), (25) и (38) вытекает, что

$$|C_M^{(1)}(\mathbf{q})| = p^{\mathtt{t}_p(c)} - 1.$$

В то же время из (24), (28), (32), (34) и (36) вытекает, что значение числа $|C_M^{(2)}(\mathbf{q})|$ представляется достаточно громоздкими суммами, вид которых существенно зависит от элемента $q_1 \in \mathbb{Z}_{p^k}$ кольца, а также от значений чисел $\mathtt{t}_p(b)$ и $\mathtt{t}_p(c)$. В настоящее время авторам не удалось получить простые оценки числа $|C_M^{(2)}(\mathbf{q})|$.

5. Задачи идентификации исследуемой модели

Рассмотрим задачу идентификации начального состояния $\mathbf{q}_0 = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ автомата $M \in \mathsf{M}$.

Теорема 5. Для любого автомата $M \in \mathsf{M}$, если экспериментатору известен набор значений параметров $(a, b, c, d, e) \in \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k} \backslash \{0\}) \times \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k}^{\text{inv}})^2$, то:

1) если $c \in \mathbb{Z}_{p^k}^{\text{inv}}$, то для любого входного слова $x_1 x_2 \in \mathbb{Z}_{p^k}^2$ начальное состояние $\mathbf{q}_0 = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ может быть вычислено по формулам

$$\begin{aligned} q_1 &= c^{-1} \circ (e^{-1} \circ y_2 \ominus a \ominus b \circ e^{-2} \circ y_1^2 \ominus d \circ x_2), \\ q_0 &= c^{-1} \circ (e^{-1} \circ y_1 \ominus a \ominus b \circ q_1^2 \ominus d \circ x_1); \end{aligned} \tag{39}$$

2) если $c \in \mathbb{Z}_{p^k} \backslash \mathbb{Z}_{p^k}^{\text{inv}}$, то множество $C_M(\mathbf{q}_0)$ $(\mathbf{q}_0 = (q_1, q_0) \in \mathbb{Z}_{p^k}^2)$ всех состояний автомата M, эквивалентных начальному состоянию $\mathbf{q}_0$, может быть вычислено по формуле

$$C_M(\mathbf{q}_0) = \bigcap_{x_1 x_2 \in \mathbb{Z}_{p^k}^2} S_{x_1 x_2}, \tag{40}$$

где $S_{x_1 x_2}$ $(x_1 x_2 \in \mathbb{Z}_{p^k}^2)$ — множество всех решений (q_1, q_0) системы уравнений

$$\begin{cases} c \circ q_1 = e^{-1} \circ y_2 \ominus a \ominus b \circ e^{-2} \circ y_1^2 \ominus d \circ x_2, \\ c \circ q_0 = e^{-1} \circ y_1 \ominus a \ominus b \circ q_1^2 \ominus d \circ x_1. \end{cases} \tag{41}$$

Доказательство. Из системы уравнений (1) вытекает, что

$$y_{t+1} = e \circ (a \oplus b \circ q_{t+1}^2 \oplus c \circ q_t \oplus d \circ x_{t+1}) \ (t \in \mathbb{Z}_+). \tag{42}$$

Полагая $t = 0$ и 1 в (42), получим, что для любого входного слова $x_1x_2 \in \mathbb{Z}_{p^k}^2$ истинны равенства

$$\begin{cases} y_1 = a \circ e \oplus b \circ e \circ q_1^2 \oplus c \circ e \circ q_0 \oplus d \circ e \circ x_1, \\ y_2 = a \circ e \oplus b \circ e \circ q_2^2 \oplus c \circ e \circ q_1 \oplus d \circ e \circ x_2, \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y_1 = a \circ e \oplus b \circ e \circ q_1^2 \oplus c \circ e \circ q_0 \oplus d \circ e \circ x_1, \\ y_2 = a \circ e \oplus b \circ e^{-1} \circ y_1^2 \oplus c \circ e \circ q_1 \oplus d \circ e \circ x_2, \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} c \circ q_1 = e^{-1} \circ y_2 \ominus a \ominus b \circ e^{-2} \circ y_1^2 \ominus d \circ x_2, \\ c \circ q_0 = e^{-1} \circ y_1 \ominus a \ominus b \circ q_1^2 \ominus d \circ x_1. \end{cases} \tag{43}$$

Если $c \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то из (43) вытекает, что истинны равенства (39), что и требовалось доказать.

Если $c \in \mathbb{Z}_{p^k} \backslash \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то из (43) и следствия 5 вытекает, что истинны равенства (40), что и требовалось доказать. ■

Из теоремы 5 вытекает, что истинно следующее следствие.

Следствие 7. Для любого автомата $M \in \mathsf{M}$, если экспериментатору известен набор значений параметров $(a, b, c, d, e) \in \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k} \backslash \{0\}) \times \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k}^{\mathrm{inv}})^2$, то

1) если $c \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то идентификация начального состояния автомата M осуществима посредством простого эксперимента длины 2;

2) если $c \in \mathbb{Z}_{p^k} \backslash \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то идентификация множества всех состояний, эквивалентных начальному состоянию автомата M, осуществима посредством кратного эксперимента кратности p^{2k} и высоты 2.

Замечание 11. Если $c = 0$, то система уравнений (41) принимает вид

$$\begin{cases} e^{-1} \circ y_2 \ominus a \ominus b \circ e^{-2} \circ y_1^2 \ominus d \circ x_2 = 0, \\ e^{-1} \circ y_1 \ominus a \ominus b \circ q_1^2 \ominus d \circ x_1 = 0, \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} e^{-1} \circ y_2 \ominus a \ominus b \circ e^{-2} \circ y_1^2 \ominus d \circ x_2 = 0, \\ b \circ q_1^2 = e^{-1} \circ y_1 \ominus a \ominus d \circ x_1, \end{cases}$$

где равенство

$$e^{-1} \circ y_2 \ominus a \ominus b \circ e^{-2} \circ y_1^2 \ominus d \circ x_2 = 0$$

не содержит переменных и является истинным равенством в кольце $\boldsymbol{Z}_{p^k}$. Следовательно, если $c = 0$, то для любого входного слова $x_1x_2 \in \mathbb{Z}_{p^k}^2$ множество $S_{x_1x_2}$ всех решений системы уравнений (41) состоит из всех таких пар (q_1, q_0), что $q_0 \in \mathbb{Z}_{p^k}$, а q_1 является решением квадратного уравнения

$$b \circ q_1^2 = e^{-1} \circ y_1 \ominus a \ominus d \circ x_1.$$

Рассмотрим задачу параметрической идентификации автомата $M \in \mathsf{M}$ в предположении, что экспериментатор может проводить с автоматом M кратный эксперимент любой кратности.

Возможны следующие два случая.

Случай 1. Начальное состояние $\mathbf{q}_0 = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ автомата M известно экспериментатору.

Возможны следующие два случая.

Случай 1.1. Пусть $\mathbf{q}_0 = (q_1, q_0) = (0, 0)$. Полагая $t = 0, 1, \ldots, l$ ($l \geqslant 6$) в (42), получим, что для любого входного слова $x_1 \ldots x_l \in \mathbb{Z}_{p^k}^l$ истинны равенства

$$\begin{aligned} a \circ e \oplus d \circ e \circ x_1 &= y_1, \\ a \circ e \oplus b \circ e^{-1} \circ y_1^2 \oplus d \circ e \circ x_2 &= y_2, \\ a \circ e \oplus b \circ e^{-1} \circ y_2^2 \oplus c \circ y_1 \oplus d \circ e \circ x_3 &= y_3, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a \circ e \oplus b \circ e^{-1} \circ y_{l-1}^2 \oplus c \circ y_{l-2} \oplus d \circ e \circ x_l &= y_l. \end{aligned} \tag{44}$$

Матрица A совместной системы уравнений (44) имеет следующий вид:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ l \end{array} \begin{array}{c} \begin{array}{cccc} a{\circ}e & b{\circ}e^{-1} & c & d{\circ}e \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & x_1 \\ 1 & y_1^2 & 0 & x_2 \\ 1 & y_2^2 & y_1 & x_3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & y_{l-1}^2 & y_{l-2} & x_l \end{pmatrix} \end{array}.$$

Найдем такое входное слово $x_1 \ldots x_l \in \mathbb{Z}_{p^k}^l$ заранее неизвестной длины $l \geqslant 4$, что матрица A содержит обратимую подматрицу B 4-го порядка. Четыре уравнения системы уравнений (44), определяемые подматрицей B, образуют систему линейных уравнений, имеющую единственное решение:

$$\begin{cases} a \circ e = \alpha_1, \\ b{\circ}e^{-1} = \alpha_2, \\ c = \alpha_3, \\ d{\circ}e = \alpha_4. \end{cases} \tag{45}$$

Из (44) вытекает, что даже при отсутствии информации об истинном значении набора параметров $(a, b, c, d, e) \in \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k} \backslash \{0\}) \times \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k}^{\mathrm{inv}})^2$ решение (45) дает возможность моделировать работу автомата M на любом входном слове $x_1 \ldots x_l \in \mathbb{Z}_{p^k}^l$, $l \in \mathbb{N}$, так как

$$\begin{aligned} y_1 &= \alpha_1 \oplus \alpha_4 \circ x_1, \\ y_2 &= \alpha_1 \oplus \alpha_2 \circ y_1^2 \oplus \alpha_4 \circ x_2, \\ y_3 &= \alpha_1 \oplus \alpha_2 \circ y_2^2 \oplus \alpha_3 \circ y_1 \oplus \alpha_4 \circ x_3, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_l &= \alpha_1 \oplus \alpha_2 \circ y_{l-1}^2 \oplus \alpha_3 \circ y_{l-2} \oplus \alpha_4 \circ x_l. \end{aligned} \tag{46}$$

А так как $\alpha_4 = d{\circ}e \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то решение (45) дает возможность по любому выходному слову $y_1 \ldots y_l \in \mathbb{Z}_{p^k}^l$ однозначно вычислить поданное на автомат M входное слово $x_1 \ldots x_l \in \mathbb{Z}_{p^k}^l$.

Случай 1.2. Пусть $\mathbf{q}_0 = (q_1, q_0) \neq (0,0)$. Полагая $t = 0, 1, \ldots, l$ $(l \geqslant 6)$ в (42), получим, что для любого входного слова $x_1 \ldots x_l \in \mathbb{Z}_{p^k}^l$ истинны равенства

$$\begin{gathered}
a \circ e \oplus b \circ e \circ q_1^2 \oplus c \circ e \circ q_0 \oplus d \circ e \circ x_1 = y_1,\\
a \circ e \oplus b \circ e^{-1} \circ y_1^2 \oplus c \circ e \circ q_1 \oplus d \circ e \circ x_2 = y_2,\\
a \circ e \oplus b \circ e^{-1} \circ y_2^2 \oplus c \circ y_1 \oplus d \circ e \circ x_3 = y_3,\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
a \circ e \oplus b \circ e^{-1} \circ y_{l-1}^2 \oplus c \circ y_{l-2} \oplus d \circ e \circ x_l = y_l.
\end{gathered} \tag{47}$$

Матрица C совместной системы уравнений (47) имеет следующий вид:

$$C = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ l \end{array} \begin{array}{c} \begin{array}{cccccc} a{\circ}e & b{\circ}e & b{\circ}e^{-1} & c{\circ}e & c & d{\circ}e \end{array} \\ \begin{pmatrix} 1 & q_1^2 & 0 & q_0 & 0 & x_1 \\ 1 & 0 & y_1^2 & q_1 & 0 & x_2 \\ 1 & 0 & y_2^2 & 0 & y_1 & x_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & y_{l-1}^2 & 0 & y_{l-2} & x_l \end{pmatrix} \end{array}.$$

Найдем такое входное слово $x_1 \ldots x_l \in \mathbb{Z}_{p^k}^l$ заранее неизвестной длины $l \geqslant 6$, что матрица C содержит обратимую подматрицу D 6-го порядка. Шесть уравнений системы уравнений (47), определяемые подматрицей D, образуют систему линейных уравнений, имеющую единственное решение:

$$\begin{cases}
a \circ e = \beta_1,\\
b{\circ}e = \beta_2,\\
b{\circ}e^{-1} = \beta_3,\\
c{\circ}e = \beta_4,\\
c = \beta_5,\\
d{\circ}e = \beta_6.
\end{cases} \tag{48}$$

Из (47) вытекает, что даже при отсутствии информации об истинном значении набора параметров $(a, b, c, d, e) \in \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k} \backslash \{0\}) \times \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k}^{\mathrm{inv}})^2$ решение (48) дает возможность моделировать работу автомата M на любом входном слове $x_1 \ldots x_l \in \mathbb{Z}_{p^k}^l$ $(l \in \mathbb{N})$, так как

$$\begin{gathered}
y_1 = \beta_1 \oplus \beta_2 \circ q_1^2 \oplus \beta_4 \circ q_0 \oplus \beta_6 \circ x_1,\\
y_2 = \beta_1 \oplus \beta_3 \circ y_1^2 \oplus \beta_4 \circ q_1 \oplus \beta_6 \circ x_2,\\
y_3 = \beta_1 \oplus \beta_3 \circ y_2^2 \oplus \beta_5 \circ y_1 \oplus \beta_6 \circ x_3,\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
y_l = \beta_1 \oplus \beta_3 \circ y_{l-1}^2 \oplus \beta_5 \circ y_{l-2} \oplus \beta_6 \circ x_l.
\end{gathered} \tag{49}$$

А так как $\beta_6 = d{\circ}e \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то решение (48) дает возможность по любому выходному слову $y_1 \ldots y_l \in \mathbb{Z}_{p^k}^l$ однозначно вычислить поданное на автомат M входное слово $x_1 \ldots x_l \in \mathbb{Z}_{p^k}^l$.

Замечание 12. Ясно, что равенства (46) представляют собой специальный случай равенств (49). Действительно, если в (49)положить $q_1 = q_0 = 0$ и (в силу равенств (45) и (48)) $\beta_1 = \alpha_1$, $\beta_3 = \alpha_2$, $\beta_5 = \alpha_3$ и $\beta_6 = \alpha_4$, то получим равенства (46).

Случай 2. Начальное состояние $\mathbf{q}_0 = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$ автомата M не известно экспериментатору.

Отбросив в системе (47) первые два уравнения, получим, что для любого входного слова $x_1 \dots x_l \in \mathbb{Z}_{p^k}^l$ $(l \geqslant 6)$ истинны равенства

$$\begin{gathered} a \circ e \oplus b \circ e^{-1} \circ y_2^2 \oplus c \circ y_1 \oplus d \circ e \circ x_3 = y_3, \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a \circ e \oplus b \circ e^{-1} \circ y_{l-1}^2 \oplus c \circ y_{l-2} \oplus d \circ e \circ x_l = y_l. \end{gathered} \tag{50}$$

Матрица E совместной системы уравнений (50) имеет следующий вид:

$$E = \begin{array}{c} \\ 1 \\ \\ l-2 \end{array} \begin{array}{c} \begin{array}{cccc} a \circ e & b \circ e^{-1} & c & d \circ e \end{array} \\ \begin{pmatrix} 1 & y_2^2 & y_1 & x_3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & y_{l-1}^2 & y_{l-2} & x_l \end{pmatrix} \end{array}.$$

Найдем такое входное слово $x_1 \dots x_l \in \mathbb{Z}_{p^k}^l$ заранее неизвестной длины $l \geqslant 6$, что матрица E содержит обратимую подматрицу F 4-го порядка. Четыре уравнения системы уравнений (50), определяемые подматрицей F, образуют систему линейных уравнений, имеющую единственное решение:

$$\begin{cases} a \circ e = \gamma_1, \\ b \circ e^{-1} = \gamma_2, \\ c = \gamma_3, \\ d \circ e = \gamma_4. \end{cases} \tag{51}$$

Из (50) вытекает, что даже при отсутствии информации об истинном значении набора параметров $(a, b, c, d, e) \in \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k} \backslash \{0\}) \times \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k}^{\mathrm{inv}})^2$ решение (51) дает возможность при известных значениях y_1 и y_2 моделировать работу автомата M на финальном отрезке $x_3 \dots x_l$ любого входного слова $x_1 \dots x_l \in \mathbb{Z}_{p^k}^l$ $(l \in \mathbb{N})$ $(l \geqslant 3)$, так как

$$\begin{gathered} y_3 = \gamma_1 \oplus \gamma_2 \circ y_2^2 \oplus \gamma_3 \circ y_1 \oplus \gamma_4 \circ x_3, \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ y_l = \gamma_1 \oplus \gamma_2 \circ y_{l-1}^2 \oplus \gamma_3 \circ y_{l-2} \oplus \gamma_4 \circ x_l. \end{gathered}$$

А так как $\gamma_4 = d \circ e \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то решение (51) дает возможность по любому выходному слову $y_1 \dots y_l \in \mathbb{Z}_{p^k}^l$ однозначно вычислить финальный отрезок $x_3 \dots x_l \in \mathbb{Z}_{p^k}^{l-2}$ поданного на автомат M входного слова $x_1 \dots x_l \in \mathbb{Z}_{p^k}^l$.

6. Неподвижные точки исследуемой модели

Неподвижной точкой словарной функции $f : X^+ \to X^+$ называется любое такое слово $u \in X^+$, что $f(u) = u$.

Обозначим через $S_{\mathrm{fxd}}(M, \mathbf{q}_0)$ $(M \in \mathsf{M},\ \mathbf{q}_0 = (q_1, q_0) \in \mathbb{Z}_{p^k}^2)$ множество всех неподвижных точек словарной функции $f_{(M, \mathbf{q}_0)} : \mathbb{Z}_{p^k}^+ \to \mathbb{Z}_{p^k}^+$, реализуемой инициальным автоматом $(M, \mathbf{q}_0)$.

Ясно, что

$$S_{\mathrm{fxd}}(M, \mathbf{q}_0) = \bigcup_{l=1}^{\infty} S_{\mathrm{fxd}}^{(l)}(M, \mathbf{q}_0),$$

где $S^{(l)}_{\mathrm{fxd}}(M,\mathbf{q}_0)$ — множество всех неподвижных точек длины l.

Так как множества $S^{(l)}_{\mathrm{fxd}}(M,\mathbf{q}_0)$ $(l\in\mathbb{N})$ попарно не пересекаются, то для исследования структуры множества $S_{\mathrm{fxd}}(M,\mathbf{q}_0)$ достаточно охарактеризовать общий член последовательности $\{S^{(l)}_{\mathrm{fxd}}(M,\mathbf{q}_0)\}_{l\in\mathbb{N}}$.

Теорема 6. Для любого автомата $M\in\mathsf{M}$ при любом начальном состоянии $\mathbf{q}_0=(q_1,q_0)\in\mathbb{Z}^2_{p^k}$ множество $S^{(l)}_{\mathrm{fxd}}(M,\mathbf{q}_0)$ $(l\in\mathbb{N})$ состоит из всех таких входных слов $x_1\dots x_l\in\mathbb{Z}^l_{p^k}$, что $(x_1,\dots,x_l)$ — решение системы уравнений

$$\begin{cases}(1\ominus\beta_6)\circ x_1=\beta_1\oplus\beta_2\circ q_1^2\oplus\beta_4\circ q_0,\\ (1\ominus\beta_6)\circ x_2\ominus\beta_3\circ x_1^2=\beta_1\oplus\beta_4\circ q_1,\\ (1\ominus\beta_6)\circ x_3\ominus\beta_3\circ x_2^2\ominus\beta_5\circ x_1=\beta_1,\\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\ (1\ominus\beta_6)\circ x_l\ominus\beta_3\circ x_{l-1}^2\ominus\beta_5\circ x_{l-2}=\beta_1,\end{cases}\tag{52}$$

где набор значений $(\beta_1,\dots,\beta_6)$ определен равенствами (48).

Доказательство. Реакцией инициального автомата $(M,\mathbf{q}_0)$ на входное слово $x_1\dots x_l\in S^{(l)}_{\mathrm{fxd}}(M,\mathbf{q}_0)$ является такое выходное слово $y_1\dots y_l\in\mathbb{Z}^l_{p^k}$, что

$$y_1\dots y_l=x_1\dots x_l.$$

Положив $y_i=x_i$ $(i=1,\dots,l)$ в равенствах (49), получим

$$\begin{cases}x_1=\beta_1\oplus\beta_2\circ q_1^2\oplus\beta_4\circ q_0\oplus\beta_6\circ x_1,\\ x_2=\beta_1\oplus\beta_3\circ x_1^2\oplus\beta_4\circ q_1\oplus\beta_6\circ x_2,\\ x_3=\beta_1\oplus\beta_3\circ x_2^2\oplus\beta_5\circ x_1\oplus\beta_6\circ x_3,\\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\ x_l=\beta_1\oplus\beta_3\circ x_{l-1}^2\oplus\beta_5\circ x_{l-2}\oplus\beta_6\circ x_l,\end{cases}\quad\Leftrightarrow$$

$$\Leftrightarrow\begin{cases}(1\ominus\beta_6)\circ x_1=\beta_1\oplus\beta_2\circ q_1^2\oplus\beta_4\circ q_0,\\ (1\ominus\beta_6)\circ x_2\ominus\beta_3\circ x_1^2=\beta_1\oplus\beta_4\circ q_1,\\ (1\ominus\beta_6)\circ x_3\ominus\beta_3\circ x_2^2\ominus\beta_5\circ x_1=\beta_1,\\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\ (1\ominus\beta_6)\circ x_l\ominus\beta_3\circ x_{l-1}^2\ominus\beta_5\circ x_{l-2}=\beta_1,\end{cases}$$

что и требовалось доказать. ■

Следствие 8. Для любого автомата $M\in\mathsf{M}$ при любом начальном состоянии $\mathbf{q}_0=(q_1,q_0)\in\mathbb{Z}^2_{p^k}$, если $1\ominus\beta_6\in\mathbb{Z}^{\mathrm{inv}}_{p^k}$, то $|S^{(l)}_{\mathrm{fxd}}(M,\mathbf{q}_0)|=1$ для всех $l\in\mathbb{N}$.

Доказательство. Если $1\ominus\beta_6\in\mathbb{Z}^{\mathrm{inv}}_{p^k}$, то для всех $l\in\mathbb{N}$ система уравнений (52) имеет единственное решение $(x_1,\dots,x_l)$, где

$$\begin{cases}x_1=(1\ominus\beta_6)^{-1}\circ(\beta_1\oplus\beta_2\circ q_1^2\oplus\beta_4\circ q_0),\\ x_2=(1\ominus\beta_6)^{-1}\circ(\beta_3\circ x_1^2\oplus\beta_1\oplus\beta_4\circ q_1),\\ x_3=(1\ominus\beta_6)^{-1}\circ(\beta_3\circ x_2^2\oplus\beta_5\circ x_1\oplus\beta_1),\\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\ x_l=(1\ominus\beta_6)^{-1}\circ\oplus\beta_3\circ x_{l-1}^2\oplus\beta_5\circ x_{l-2}\oplus\beta_1),\end{cases}$$

что и требовалось доказать. ■

Следствие 9. Для любого автомата $M \in \mathsf{M}$ при любом начальном состоянии $\mathbf{q}_0 = (q_1, q_0) \in \mathbb{Z}_{p^k}^2$, если $a \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, $\mathtt{t}_p(b)>0$, $\mathtt{t}_p(c)>0$ и $\mathtt{t}_p(1 \ominus d \circ e)>0$, то $S_{\mathrm{fxd}}(M, \mathbf{q}_0) = \varnothing$.

Доказательство. Рассмотрим первое уравнение системы уравнений (52):

$$(1 \ominus \beta_6) \circ x_1 = \beta_1 \oplus \beta_2 \circ q_1^2 \oplus \beta_4 \circ q_0. \tag{53}$$

Так как $\mathtt{t}_p(1 \ominus d \circ e) > 0$, $\mathtt{t}_p(b) > 0$ и $\mathtt{t}_p(c) > 0$, то

$$\begin{gathered}\mathtt{t}_p(1 \ominus \beta_6) = \mathtt{t}_p(1 \ominus d \circ e) > 0,\\ \mathtt{t}_p(\beta_2) = \mathtt{t}_p(b \circ e) > 0,\\ \mathtt{t}_p(\beta_4) = \mathtt{t}_p(c \circ e) > 0.\end{gathered}$$

А так как $a, e \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то

$$\mathtt{t}_p(\beta_1) = \mathtt{t}_p(a \circ e) = 0.$$

Следовательно, для любых $x_1, q_1, q_0 \in \mathbb{Z}_{p^k}$

$$\mathtt{t}_p((1 \ominus \beta_6) \circ x_1) > 0, \tag{54}$$

$$\mathtt{t}_p(\beta_1 \oplus \beta_2 \circ q_1^2 \oplus \beta_4 \circ q_0) = 0. \tag{55}$$

Из (54) и (55) вытекает, что уравнение (53) не имеет решений, т. е.

$$S_{\mathrm{fxd}}^{(1)}(M, \mathbf{q}_0) = \varnothing.$$

А так как

$$S_{\mathrm{fxd}}^{(l+1)}(M, \mathbf{q}_0) \subseteq \{ux | u \in S_{\mathrm{fxd}}^{(l)}(M, \mathbf{q}_0),\ x \in \mathbb{Z}_{p^k}\}\ (l \in \mathbb{N})),$$

то

$$S_{\mathrm{fxd}}^{(l)}(M, \mathbf{q}_0) = \varnothing$$

для всех $l \in \mathbb{N}$.

Отсюда вытекает, что $S_{\mathrm{fxd}}(M, \mathbf{q}_0) = \varnothing$, что и требовалось доказать. ■

Заключение

В работе исследован класс M обратимых нелинейных одномерных автоматов Мура, определенных системой уравнений с лагом 2 над кольцом $\boldsymbol{Z}_{p^k}$, которые на каждом такте своего функционирования осуществляют линейное преобразование линейной комбинации квадратичной формы текущего состояния автомата и аффинного преобразования входного символа. Для исследуемой модели охарактеризованы структура автоматного графа и множества эквивалентных состояний, решены задачи параметрической идентификации и идентификации начального состояния, охарактеризованы множества неподвижных точек отображений, реализуемых инициальными автоматами.

Полученные результаты следующим образом характеризуют симметричный поточный шифр (3).

В п. 6 (следствие 9) выделены значения ключей (a, b, c, d, e) средней длительности, при которых отображение $f_{(M,\mathbf{q}_0)} : \mathbb{Z}_{p^k}^+ \to \mathbb{Z}_{p^k}^+$ не имеет неподвижных точек. Систематический анализ вариации таких отображений при вариации секретного сеансового ключа $\mathbf{q}_0 \in \mathbb{Z}_{p^k}^2$ является одним из возможных направлений дальнейших исследований.

Если $c \in \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то любые сеансовые ключи неэквивалентны (теорема 4), т. е. не возникает задача выбора нового секретного сеансового ключа, неэквивалентного предыдущему секретному сеансовому ключу. Однако, если криптоаналитик располагает секретным ключом средней длительности (a, b, c, d, e), то он без труда идентифицирует секретный сеансовый ключ (теорема 5 и следствие 7). Если же $c \notin \mathbb{Z}_{p^k}^{\mathrm{inv}}$, то, даже располагая секретным ключом средней длительности (a, b, c, d, e), для идентификации секретного сеансового ключа (с точностью до множества эквивалентных состояний) криптоаналитик вынужден решать совокупность систем квадратных уравнений над кольцом $\boldsymbol{Z}_{p^k}$, полученных посредством кратного эксперимента кратности p^{2k} и высоты 2 (теорема 5 и следствие 7). Однако при этом возникает задача выбора нового секретного сеансового ключа, неэквивалентного предыдущему секретному сеансовому ключу. Эта задача решается посредством кратного эксперимента кратности не более p^{2k} и высоты не более 2 (теорема 5).

В п. 5 показано, что вычислительная стойкость симметричного поточного шифра (3) полностью определяется сложностью поиска решения (51) (а не сложностью идентификации набора параметров (a, b, c, d, e)). Поэтому вторым из возможных направлений исследований является анализ с позиции криптографии подклассов автоматов, принадлежащих классу M и определяемых четверкой $(a \circ e, b \circ e^{-1}, c, d \circ e)$ $((a, b, c, d, e) \in \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k} \backslash \{0\}) \times \mathbb{Z}_{p^k} \times (\mathbb{Z}_{p^k}^{\mathrm{inv}})^2)$ элементов кольца $\boldsymbol{Z}_{p^k} = (\mathbb{Z}_{p^k}, \oplus, \circ)$.

Третьим возможным направлением исследований является анализ изменения вычислительной стойкости симметричного поточного шифра при сохранении функции переходов и замене линейной функции выхода на нелинейную функцию выхода, а также при переходе к автомату Мили.

ЛИТЕРАТУРА

1. *Скобелев В. В., Скобелев В. Г.* Анализ шифрсистем. Донецк: ИПММ НАН Украины, 2009. 479 с.
2. *Скобелев В. Г.* Комбинаторно-алгебраические модели в криптографии // Прикладная дискретная математика. Приложение. 2009. № 2. С. 74–114.
3. *Глушков В. М.* Синтез цифровых автоматов. М.: Физматлит, 1962. 476 с.
4. *Трахтенброт Б. А., Барздинь Я. М.* Конечные автоматы (поведение и синтез). М.: Наука, 1970. 400 с.
5. *Кузнецов С. П.* Динамический хаос. М.: Физматлит, 2001. 296 с.
6. *Скобелев В. Г., Тубольцева О. В.* Шифр на основе отображения Эно // Вестник Томского госуниверситета. Приложение. 2005. № 14. С. 74–78.
7. *Гилл А.* Введение в теорию конечных автоматов. М.: Наука, 1966. 272 с.
8. *Bollobas B.* Modern graph theory. NY: Springer Verlag, 1998. 394 p.
9. *Коршунов А. Д.* О перечислении конечных автоматов // Проблемы кибернетики. Вып. 34. М.: Наука, 1978. С. 5–82.
10. *Кудрявцев В. Б., Алешин С. В., Подколзин А. С.* Введение в теорию конечных автоматов. М.: Наука, 1985. 320 с.

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/7/7

УДК 519.7

АЛГОРИТМЫ РАБОТЫ С ROBDD КАК С БАЗАМИ БУЛЕВЫХ ОГРАНИЧЕНИЙ

А. С. Игнатьев, А. А. Семенов

Институт динамики систем и теории управления СО РАН, г. Иркутск, Россия

E-mail: aign@icc.ru, biclop@rambler.ru

Исследуются алгоритмические свойства сокращенных упорядоченных диаграмм решений (ROBDD) при их рассмотрении в роли баз булевых ограничений в гибридном (SAT+ROBDD)-выводе. Приведены ROBDD-аналоги основных алгоритмических процедур, используемых в DPLL-выводе (подстановки, правило единичного дизъюнкта, CL-процедура, механизмы отсроченных вычислений). Описан новый алгоритм изменения порядка в ROBDD. Для всех алгоритмов приводятся оценки их трудоемкости.

Ключевые слова: *логические уравнения, двоичные диаграммы решений, гибридный вывод.*

Введение

В последние годы заметен рост интереса к символьным алгоритмам, эффективным на практически важных классах логических уравнений. Особенно впечатляет прогресс в разработке таких алгоритмов для решения задачи ВЫПОЛНИМОСТЬ (SAT-задачи, см. [1]). Этим вопросам посвящены многочисленные конференции и специализированные издания (одной из центральных тем издающегося в Нидерландах журнала JSAT [2] является алгоритмика SAT-задач).

SAT-задачи находят широкое применение в различных разделах прикладной дискретной математики и кибернетики: синтез и верификация дискретных автоматов [3], верификация программных логик [4, 5], криптоанализ [6], задачи информационной биологии [7] и во многих других областях.

Наибольшую эффективность в решении SAT-задач (по результатам специализированных конкурсов [1]) демонстрируют методы, использующие в своей основе алгоритм DPLL (Devis, Putnam, Logemann, Loveland, см. [8]) и последующие его модернизации [9 – 11]. Главное отличие поздних версий от классического DPLL в том, что в них информация о ходе вывода хранится в форме булевых ограничений-дизъюнктов, запрещающих конфликтные присвоения.

Одной из центральных проблем в современных высокоскоростных SAT-решателях является проблема переполнения памяти генерируемыми в процессе поиска булевыми ограничениями. Вообще говоря, несложно построить семейства противоречивых конъюнктивных нормальных форм (КНФ), для опровержения которых перечисленным алгоритмам потребуется порождать экспоненциальное от размерности КНФ число конфликтных дизъюнктов, однако такие тесты искусственны и далеки по своей природе от реальных практических задач.

На практике проблема переполнения памяти решается при помощи процедур чистки баз ограничений, в результате которых признаваемые нерелевантными ограничения попросту отбрасываются. Однако все заключения о релевантности носят характер эвристик и, как следствие, отбрасывание ограничений в общем случае приводит к потере алгоритмом полноты ввиду невозможности гарантировать окончание его работы на произвольных КНФ. Эффект потери полноты наблюдался на некоторых криптографических тестах и приводил при крупноблочном распараллеливании соответствующих SAT-задач к «сверхлинейному» ускорению (см. [12]).

В работе [13] для решения проблемы переполнения памяти на задачах обращения дискретных функций был предложен гибридный подход, в котором нехронологический DPLL-вывод сочетается с выводом на двоичных диаграммах решений (BDD), а точнее, на сокращенных упорядоченных BDD, или ROBDD.

Настоящая статья посвящена описанию основных алгоритмов работы с ROBDD, рассматриваемыми в роли баз булевых ограничений, порожденных в процессе нехронологического DPLL-вывода. Для всех представленных в работе алгоритмов построены оценки их трудоемкости. Приведем краткий план статьи.

В п. 1 содержатся необходимые сведения об алгоритмах логического вывода на основе DPLL и их применении к задачам обращения полиномиально вычислимых дискретных функций. Здесь же кратко описываются двоичные диаграммы решений и основные алгоритмы манипулирования булевыми функциями с их помощью.

В п. 2 кратко описаны результаты работы [13] по основам гибридного (SAT+ROBDD)-подхода к обращению дискретных функций. Приведены (без доказательств) теорема о ядре DPLL-вывода и теорема, обосновывающая использование ROBDD в роли базы булевых ограничений в гибридном (SAT+ROBDD)-выводе на SAT-задачах, кодирующих проблемы обращения полиномиально вычислимых дискретных функций.

В п. 3 описаны основные алгоритмы работы с ROBDD как с базами булевых ограничений в гибридном (SAT+ROBDD)-выводе и даны оценки их трудоемкости. Приведен новый алгоритм изменения порядка в ROBDD на произвольный новый порядок. Описаны также ROBDD-аналоги основных алгоритмов, используемых в современных SAT-решателях, базирующихся на DPLL (подстановка значения переменной, правило единичного дизъюнкта, CL-процедура, механизмы отсроченных вычислений при работе с булевыми ограничениями).

1. Базовые понятия, конструкции и алгоритмы

1.1. SAT-задачи и их алгоритмика

К SAT-задачам относятся задачи поиска решений логических [14] (булевых) уравнений вида

$$C(x_1, \ldots, x_k) = 1,$$

где $C(x_1, \ldots, x_k)$ — формула исчисления высказываний (ИВ), имеющая вид КНФ; $x_1, \ldots, x_k$ — булевы переменные, образующие множество X. В общей постановке SAT-задачи NP-трудны, однако к ним сводится настолько обширное множество практически важных задач, что построение эвристических алгоритмов, эффективных на тех или иных классах SAT-задач, является одной из актуальных областей современной компьютерной алгебры.

Наиболее эффективные на сегодняшний день SAT-решатели используют в своей основе алгоритм DPLL и дальнейшие его модификации. В [15] описана архитектура современных скоростных SAT-решателей, базирующихся на DPLL. Далее кратко перечислены их основные алгоритмические компоненты.

1) Алгоритм DPLL и его составляющие: правило единичного дизъюнкта, стратегия распространения булевых ограничений (BCP) [8].
2) Графы анализа вывода, процедура «Clause Learning» (далее CL-процедура), смысл которой заключается в конъюнктивном приписывании к текущей КНФ новых ограничений-дизъюнктов, запрещающих приведшие к конфликтам присвоения [9].
3) Процедуры анализа конфликтов и построения конфликтных дизъюнктов [9, 16]. Их предназначение — синтез новых булевых ограничений-дизъюнктов.
4) Механизмы отсроченных вычислений (типа «watched literals», [17]). С их помощью сокращается время, затрачиваемое на подстановку в КНФ значений переменных (в некоторые дизъюнкты, не удовлетворяющие определенным признакам, подстановка не осуществляется).

Использование SAT-решателей с перечисленными компонентами оказалось оправданным даже на таких аргументированно трудных задачах, как задачи обращения некоторых криптографических функций (см. [6, 18]). При этом следует отметить выгодные свойства перечисленных алгоритмов при их крупноблочном распараллеливании (см. [18, 19]).

Здесь мы очень кратко опишем общую схему сведения проблем обращения полиномиально вычислимых дискретных функций к SAT-задачам (более подробное описание можно найти, например, в [20]).

Обозначим через $\Im$ класс, образованный натуральными семействами вычислимых за полиномиальное время дискретных функций вида $f : \{0,1\}^n \to \{0,1\}^*$. Проблема обращения $f \in \Im$ в произвольной точке $y \in \operatorname{range} f$ ставится следующим образом: зная y и алгоритм вычисления f, найти $x \in \{0,1\}^n$, такой, что $f(x) = y$. Используя фундаментальную идею С. Кука (см. [21]), можно свести данную задачу к задаче поиска выполняющего набора выполнимой КНФ $C(f)$ над множеством булевых переменных $\tilde{X} = \{x_1, \ldots, x_{q(n)}\}$, $q(\cdot)$ — некоторый полином. Именно этот факт лежит в основе многочисленных примеров применения SAT-решателей к задачам обращения дискретных функций. Про КНФ $C(f)$ (которую также будем обозначать через $C(x_1, \ldots, x_{q(n)})$) будем говорить, что она кодирует задачу обращения функции f в точке $y \in \operatorname{range} f$.

1.2. Двоичные диаграммы решений и алгоритмы манипулирования булевыми функциями на их основе

Двоичные диаграммы решений (BDD) были введены К. Ли в [22]. Фундаментальность этой структуры данных для дискретной математики была осознана после выхода работы Р. Брайанта [23], в которой он описал семейство алгоритмов «манипулирования» булевыми функциями при помощи BDD.

Двоичные диаграммы решений — это ориентированные ациклические помеченные графы, представляющие булевы функции. Вообще говоря, произвольную BDD можно рассматривать как графическую интерпретацию рекурсивного применения к некоторой булевой функции разложения Шеннона. Однако более наглядным является переход к BDD от двоичных деревьев решений (см. [24]).

Пусть $T(f)$ — некоторое двоичное дерево решений, представляющее всюду определенную булеву функцию $f : \{0,1\}^n \to \{0,1\}$. Для определенности можно считать, что f выражена формулой ИВ $L(f)$. Корень и внутренние вершины $T(f)$ помечаются переменными из множества $X = \{x_1, \ldots, x_n\}$, листья помечаются константами из $\{0,1\}$. Каждый путь в $T(f)$ из корня в лист определяет некоторую последовательную подстановку значений переменных из X в $L(f)$. Константа, приписанная листу, есть значение

функции f, полученное в результате этой подстановки. Из каждой вершины, помеченной переменной из X, выходят два ребра: пунктирное, что соответствует принятию переменной значения 0, и сплошное, что соответствует принятию ею значения 1. Традиционно используются термины «low-ребра» (для пунктирных) и «high-ребра» (для сплошных). Порядок выбора переменных из X при осуществлении последовательной подстановки будем называть порядком означивания переменных в соответствующем пути. Если в дереве $T(f)$ все пути подчинены общему порядку означивания переменных, то назовем это дерево упорядоченным. Если склеить все листья, помеченные «0», упорядоченного дерева в один и то же самое проделать с листьями, помеченными «1», то получится OBDD (упорядоченная двоичная диаграмма решений). Можно заметить, что каждая вершина в произвольной OBDD определяется тройкой «координат»: переменной из X, которой данная вершина помечена, low-ребенком и high-ребенком (то есть детьми данной вершины по соответствующим ребрам). Вершины, определяемые одинаковыми тройками координат в OBDD, можно склеить. Кроме этого, можно удалить (с сохранением структуры) из OBDD вершины, у которых low-ребенок совпадает с high-ребенком. В результате получим сокращенную упорядоченную диаграмму решений, или ROBDD. Каноническая теорема Р. Брайанта [23] утверждает, что произвольная всюду определенная булева функция f при фиксированном порядке означивания переменных единственным образом (с точностью до изоморфизма графов) представляется ROBDD $B(f)$.

Представление булевых функций в виде ROBDD имеет ряд привлекательных свойств. Во-первых, в силу канонической теоремы, произвольную ROBDD можно рассматривать как «сжатое» представление соответствующей булевой функции в специальном классе графов. Во вторых, используя ROBDD-представления, можно оперировать с булевыми функциями при помощи алгоритмов, большая часть которых была приведена в статье Р. Брайанта [23] (ниже перечислены основные).

1) Алгоритм *Apply* позволяет на основе ROBDD-представлений функций $B(f_1)$ и $B(f_2)$ построить ROBDD-представление функции $f_1 * f_2$, где $*$ — произвольная бинарная логическая связка. При совпадении порядка означивания переменных в $B(f_1)$ и $B(f_2)$ сложность *Apply* ограничена сверху величиной $O(|B(f_1)|\cdot|B(f_2)|)$ (здесь и далее через $|B|$ обозначается число вершин в ROBDD B).
2) Алгоритм *Restrict* по ROBDD, представляющей функцию f, выраженную формулой $L(f)$, строит ROBDD-представление функции $f|_{x=\alpha}$, выраженной формулой $L(f)|_{x=\alpha}$, для произвольных $x \in X$ и $\alpha \in \{0,1\}$. Сложность *Restrict* есть $O(|B(f)|)$.
3) Алгоритм *Satcount* позволяет по ROBDD $B(f)$, представляющей функцию f, подсчитать число векторов значений переменных из X, на которых f принимает значение 1. Сложность *Satcount* есть $O(|B(f)|)$.

Рассмотрим произвольную систему логических уравнений S следующего вида:

$$\begin{cases} U_1(x_1,\ldots,x_n) = 1, \\ \ldots \\ U_m(x_1,\ldots,x_n) = 1. \end{cases}$$

Под характеристической функцией системы S будем понимать булеву функцию $\chi_S : \{0,1\}^n \to \{0,1\}$, заданную формулой

$$U_1(x_1,\ldots,x_n)\cdot\ldots\cdot U_m(x_1,\ldots,x_n).$$

Несложно понять, что, имея ROBDD-представление функции χ_S, можно за линейное от числа вершин в данной ROBDD время предъявить решение системы S, либо констатировать ее несовместность.

Алгоритмы работы с ROBDD можно использовать для решения задач обращения полиномиально вычислимых дискретных функций напрямую, не переходя к логическим уравнениям (см. [25]). Однако проведенные вычислительные эксперименты показывают, что ROBDD-подход проигрывает по эффективности SAT-подходу на большинстве криптографических тестов. Кроме этого, можно показать (см., например, [26]), что задачи построения ROBDD-представлений булевых функций, заданных хорновскими КНФ (т. е. КНФ, состоящими из двухбуквенных дизъюнктов), не могут быть в общем случае решены за полиномиальное время в предположении, что $P \neq NP$. При этом известно, что SAT-задачи для таких КНФ решаются за полиномиальное время.

2. Гибридный (SAT+ROBDD)-вывод в применении к задачам обращения дискретных функций

Как было отмечено выше, использование ROBDD «в чистом виде» оправдано лишь на обращении сравнительно простых функций (криптоанализ простейших генераторов типа Геффе). Однако ROBDD привлекательны как структуры данных, наиболее экономным образом представляющие булевы функции в специальном классе графов. Этот факт приводит к идее использования ROBDD в качестве структур, представляющих булевы ограничения, накапливаемые в процессе нехронологического DPLL-вывода. Данная идея представляется перспективной именно в отношении задач обращения полиномиально вычислимых функций. И этому есть целый ряд причин.

Одна из главных причин состоит в том, что если рассматривать задачу обращения некоторой полиномиально вычислимой функции $f : \{0,1\}^n \to \{0,1\}^*$, $f \in \Im$, как SAT-задачу, то в соответствующей КНФ $C(f)$ можно выделить подмножество булевых переменных, от которых в некотором смысле «функционально зависят» все остальные переменные, фигурирующие в данной КНФ. Это множество, обозначаемое далее через X, образовано булевыми переменными, кодирующими входное слово из $\{0,1\}^n$. Как правило, число n существенно меньше общего числа переменных в $C(f)$. Однако можно показать, что для решения соответствующей SAT-задачи достаточно оперировать (в указанном ниже смысле) только с переменными множества X. Кратко остановимся на перечисленных моментах (результаты, представленные в данном пункте, подробно изложены в [13]).

Пусть $C = C(x_1, \ldots, x_k)$ — произвольная КНФ над множеством булевых переменных $\tilde{X} = \{x_1, \ldots, x_k\}$. Рассмотрим некоторое множество $X' = (x'_1, \ldots, x'_r)$, являющееся подмножеством $\tilde{X}$. Пусть $(\alpha_1, \ldots, \alpha_r)$ — произвольный вектор значений переменных из X'. Осуществим подстановку в КНФ C значения $x'_1 = \alpha_1$. Данная подстановка заключается в вычеркивании из C некоторых литералов и дизъюнктов. При этом отслеживаются возможности срабатывания правила единичного дизъюнкта с последующими подстановками в C соответствующих индуцированных значений. Если в результате не выведен конфликт или выполняющий C набор, то в КНФ $C|_{x'_1=\alpha_1}$ осуществляется подстановка $x'_2 = \alpha_2$. И так далее.

Описанная процедура определяет последовательную подстановку в C вектора $(\alpha_1, \ldots, \alpha_r)$ относительно порядка $x'_1 \prec \ldots \prec x'_r$. Очевидно, что последовательная подстановка в общем случае реализуется эффективно. Возможны различные исходы этой процедуры. Во-первых, может оказаться, что данная подстановка выводит конфликт. Во-вторых, что ее результатом является нахождение некоторого выполняющего C на-

бора. Наконец, возможен переход к некоторой КНФ, относительно которой нельзя сказать ничего. Если имеет место первая или вторая ситуация, то будем говорить, что данная подстановка индуцирует детерминированный DPLL-вывод соответственно конфликта или выполняющего набора.

Предположим, что относительно некоторой КНФ C над $\tilde{X}$ и некоторого множества $X' \subset \tilde{X}$ можно показать, что результатом последовательной подстановки любого вектора значений переменных из X' в C относительно некоторого порядка могут быть только первая или вторая ситуации. Несложно видеть, что в этом случае достаточно перебрать $2^{|X'|}$ всевозможных значений переменных из X', чтобы решить рассматриваемую SAT-задачу в отношении КНФ C.

Определение 1. Пусть $\tilde{X} = \{x_1, \ldots, x_k\}$ — множество булевых переменных и $X' \subseteq \tilde{X}$. Проекцией произвольного вектора $\alpha = (\alpha_1, \ldots, \alpha_k)$ значений переменных из $\tilde{X}$ на множество X' называется вектор, образованный теми компонентами α, которые являются значениями переменных из X'. Проекцию вектора α на множество X' обозначим через $\alpha_{X'}$.

Определение 2. Ядром DPLL-вывода для КНФ $C = C(x_1, \ldots, x_k)$ над множеством булевых переменных $\tilde{X} = \{x_1, \ldots, x_k\}$ называется такое множество $X^{\ker}(C) \subseteq \tilde{X}$ с введенным на нем порядком τ, что имеют место следующие свойства:

1) для любого вектора $\alpha = (\alpha_1, \ldots, \alpha_k)$, выполняющего C, последовательная подстановка в C вектора $\alpha_{X^{\ker}(C)}$ относительно τ индуцирует детерминированный DPLL-вывод α;
2) для любого вектора $\beta = (\beta_1, \ldots, \beta_k)$, такого, что $C|_\beta = 0$, последовательная подстановка в C вектора $\beta_{X^{\ker}(C)}$ относительно τ индуцирует детерминированный DPLL-вывод конфликта.

Ядро $X^{\ker}(C) = \tilde{X}$ называется тривиальным. Ядро наименьшей мощности называется минимальным и обозначается через $X^{\ker}_*(C)$.

Рассмотрим произвольную функцию $f : \{0,1\}^n \to \{0,1\}^*$, $f \in \Im$. Функция f — это дискретная функция, которая может быть выражена формулой от булевых переменных $x_1, \ldots, x_n$. Назовем множество $X = \{x_1, \ldots, x_n\}$ множеством переменных входа функции f. Рассматриваем задачу обращения f в произвольной точке $y \in \operatorname{range} f$. При помощи преобразований Цейтина (см. [27]) и техники, описанной, например, в [20], сводим за полиномиальное время данную проблему к задаче поиска выполняющего набора выполнимой КНФ $C(f)$ от булевых переменных, образующих множество $\tilde{X} = \{x_1, \ldots, x_{q(n)}\}$, $q(\cdot)$ — некоторый полином. Справедлива следующая теорема.

Теорема 1 [13]. Рассмотрим произвольную функцию $f : \{0,1\}^n \to \{0,1\}^*$ из класса $\Im$. Обозначим через X_τ множество переменных входа f с зафиксированным на нем порядком τ (вообще говоря, произвольным). Пусть $C(x_1, \ldots, x_{q(n)})$ — КНФ, кодирующая задачу обращения функции f в произвольной точке $y \in \operatorname{range} f$. Тогда $X^{\ker}_*(C(x_1, \ldots, x_{q(n)})) \subseteq X_\tau$.

Фактически данная теорема означает следующее. Будем рассматривать задачу обращения функции f в произвольной точке $y \in \operatorname{range} f$ как SAT-задачу в отношении КНФ $C(x_1, \ldots, x_{q(n)})$, решаемую при помощи любого алгоритма, основанного на DPLL. Тогда в качестве переменных уровней решения достаточно выбирать переменные входа рассматриваемой функции. Очень важен тот факт, что порядок выбора может быть произвольным. Это означает, что стратегия выбора переменных только из X гарантирует решение задачи обращения f в произвольной точке $y \in \operatorname{range} f$ алгоритмом DPLL, использующим CL-процедуру и рестарты [28].

Рассмотрим систему логических уравнений вида

$$\begin{cases} W_1(y_1,\ldots,y_s) & = 1, \\ \ldots & \\ W_k(y_1,\ldots,y_s) & = 1, \\ C(z_1,\ldots,z_t) & = 1 \end{cases} \tag{1}$$

над множеством булевых переменных $U = \{y_1,\ldots,y_s\} \cup \{z_1,\ldots,z_t\}$. Предположим, что $C(z_1,\ldots,z_t)$ — КНФ, для которой известно некоторое нетривиальное ядро DPLL-вывода $Z^{\ker}(C)$, причем $Z^{\ker}(C) \subseteq \{y_1,\ldots,y_s\}$. Через $B(W)$ обозначим ROBDD-представление характеристической функции системы

$$\begin{cases} W_1(y_1,\ldots,y_s) & = 1, \\ \ldots & \\ W_k(y_1,\ldots,y_s) & = 1. \end{cases} \tag{2}$$

Определение 3. Пусть $B(W)$ — ROBDD-представление булевой функции $W : \{0,1\}^s \to \{0,1\}$ от s булевых переменных $y_1,\ldots,y_s$. Назовем путь в $B(W)$ от корня к терминальной вершине «1» полным относительно множества $Y' \subseteq Y = \{y_1,\ldots,y_s\}$, если прохождение этого пути задает присвоение соответствующих значений всем переменным из Y'.

Справедлива следующая теорема.

Теорема 2 [29]. Прохождение любого пути в ROBDD $B(W)$ из корня в терминальную вершину «1», который полон относительно множества $Z^{\ker}(C)$, индуцирует подстановку в КНФ $C(z_1,\ldots,z_t)$ значений переменных из $Z^{\ker}(C)$. Результатом этой подстановки является либо вывод по правилу единичного дизъюнкта конфликта, либо вывод набора, выполняющего $C(z_1,\ldots,z_t)$. В последнем случае имеем некоторое решение исходной системы.

Пусть уравнение $C(x_1,\ldots,x_{q(n)}) = 1$ кодирует задачу обращения функции f из класса $\Im$ в некоторой точке $y \in \operatorname{range} f$. Через $X^{\ker}(C)$ обозначено некоторое нетривиальное ядро DPLL-вывода КНФ C (для рассматриваемой задачи в качестве $X^{\ker}(C)$ всегда можно взять X — множество переменных входа функции f). Организуем решение задачи поиска набора, выполняющего C, при помощи алгоритма DPLL, дополненного CL-процедурой (см. п. 1). При этом, руководствуясь теоремой 1, будем выбирать в качестве переменных уровней решения только те переменные, которые находятся в $X^{\ker}(C)$. Допустим, что осуществлено Q итераций такого выбора с последующим распространением булевых ограничений и реализацией CL-процедуры, но выполняющий набор при этом не найден. Обозначим через

$$D_1\left(x_1^1,\ldots,x_{r_1}^1\right),\ldots,D_Q\left(x_1^Q,\ldots,x_{r_Q}^Q\right)$$

конфликтные дизъюнкты, выведенные в данных итерациях (очевидно, что $\bigcup\limits_{i=1}^{Q}\{x_1^i,\ldots,x_{r_i}^i\} \subseteq X^{\ker}(C)$). Рассмотрим следующую систему логических уравнений:

$$\begin{cases} D_1(x_1^1,\ldots,x_{r_1}^1) & = 1, \\ \ldots & \\ D_Q(x_1^Q,\ldots,x_{r_Q}^Q) & = 1. \end{cases} \tag{3}$$

Определение 4. Пусть S — произвольная совместная система логических уравнений, $B(S)$ — ROBDD-представление ее характеристической функции и π — произвольный путь из корня $B(S)$ в терминальную вершину «1». Данный путь определяет некоторое множество решений S, обозначаемое через $A(\pi)$, $|A(\pi)| \geqslant 1$. Про любое $\alpha \in A(\pi)$ говорим также, что путь π содержит α.

Теорема 3 [13]. Обозначим через $\alpha = (\alpha_1, \ldots, \alpha_n)$ произвольное решение задачи обращения функции f из класса $\Im$ в некоторой точке $y \in \operatorname{range} f$. Пусть B — ROBDD-представление характеристической функции системы (3) в контексте рассматриваемой задачи. Тогда существует такой путь π из корня B в терминальную вершину «1» , что $\alpha \in A(\pi)$.

Отметим, что данная теорема определяет конкретный вид систем (1)–(2) в контексте задачи обращения функции f в точке $y \in \operatorname{range} f$. Роль $C(z_1, \ldots, z_t)$ в этом случае играет КНФ $C(f) = C(x_1, \ldots, x_{q(n)})$, а роль системы (2) — система (3), образованная уравнениями вида $D_i = 1, i \in \{1, \ldots, Q\}$, где D_i — конфликтные дизъюнкты, полученные в результате DPLL-вывода, примененного к КНФ $C(x_1, \ldots, x_{q(n)})$, с выбором переменных уровней решения из $X^{\ker}(C)$.

Теоремы 1–3 дают теоретическую базу для нового подхода к решению задач обращения полиномиально вычислимых дискретных функций, основная идея которого состоит в совместном использовании SAT и ROBDD именно в тех «частях» задачи обращения, где они могут дать ощутимый выигрыш. Как уже говорилось, вряд ли можно за счет использования только ROBDD обогнать SAT-подход на задачах обращения криптографических функций. Однако ROBDD могут использоваться для решения другой важной проблемы — потери полноты SAT-решателем в результате чистки баз ограничений. Именно ROBDD представляются оптимальными структурами данных для хранения массивов конфликтных дизъюнктов, накапливаемых SAT-решателем в процессе вывода (и это целиком подтверждается численными экспериментами). Особо отметим, что никакие из синтезированных в процессе вывода ограничений при этом не удаляются (в отличие от практики, принятой в большинстве современных SAT-решателей).

Все сказанное означает необходимость описания и изучения ROBDD-аналогов механизмов логического вывода, используемых в современных SAT-решателях, базирующихся на DPLL. Соответствующие процедуры будут применяться к базам булевых ограничений, представленным не в виде конъюнкций дизъюнктов, а в виде ROBDD.

3. Алгоритмы работы с ROBDD как с базами булевых ограничений

3.1. Алгоритмы логического вывода на ROBDD

Рассматривается проблема обращения функции $f : \{0,1\}^n \to \{0,1\}$ из класса $\Im$. Пусть B — ROBDD-представление характеристической функции системы логических уравнений (3). Дальнейшая цель состоит в описании процесса логического вывода на ROBDD B. Для этого потребуется определить аналоги таких компонент DPLL-вывода, как правило единичного дизъюнкта, CL-процедура, процедуры «отсроченной» работы с данными (head-tail literals и watched literals). Все приводимые далее результаты справедливы в отношении произвольных ROBDD.

Определение 5. Пусть B — ROBDD-представление произвольной булевой функции от булевых переменных $x_1, \ldots, x_n$. Каждой переменной $x_i, i \in \{1, \ldots, n\}$, и терминальным вершинам «0», «1» поставим в соответствие множества значений данной переменной, задаваемых всевозможными путями в B из корня в соответствующую терминальную вершину. Данные множества обозначим через $\Delta^0(x_i)$, $\Delta^1(x_i)$.

Предположим, что в некоторой ROBDD B выполнены следующие условия:

1) Для некоторой переменной $x_k \in X = \{x_1, \ldots, x_n\}$ любой путь π из корня B в терминальную вершину «1» обязательно проходит через некоторую вершину, помеченную переменной x_k.
2) Справедливо $|\Delta^1(x_k)| = 1$.

Результатом данной ситуации является заключение о том, что в любом наборе значений переменных из множества X, на котором значение функции f, представленной B, равно 1, переменная x_k может принимать только одно значение (соответствующее значение в $\Delta^1(x_k)$).

Определение 6. Определяемую условиями 1–2 ситуацию далее называем ROBDD-следствием соответствующего значения для переменной x_k.

Возникновение в B, представляющей базу булевых ограничений-дизъюнктов, ROBDD-следствия для некоторой переменной является аналогом правила единичного дизъюнкта в DPLL-выводе. Покажем, что справедлив следующий факт.

Лемма 1. Выполнимость 1–2 относительно некоторой переменной x_k означает, что выполнено в точности одно из следующих условий:

1) для каждой вершины, помеченной x_k, ее high-ребенком является терминальная вершина «0»;
2) для каждой вершины, помеченной x_k, ее low-ребенком является терминальная вершина «0».

Доказательство. Прежде всего отметим, что из любой нетерминальной вершины ROBDD B достижима как вершина «0», так и вершина «1».

Предположим теперь, что выполнены условия 1–2. Это означает, что для некоторой переменной x_k любой путь в ROBDD B содержит вершину, помеченную данной переменной, и пути в B задают x_k одно и то же значение (поскольку $|\Delta^1(x_k)| = 1$) — либо 0, либо 1. Не ограничивая общности, будем считать, что все пути задают x_k значение 1 ($\Delta^1(x_k) = \{1\}$). Если предположить, что ребенком некоторой $v(x_k)$ является терминальный «0», то это может быть только low-ребенок. Действительно, в противном случае из $v(x_k)$ по low-ребенку достижима терминальная вершина «1», и соответствующий путь задает x_k значение 0, что противоречит сделанному предположению.

Предположим, что найдется такая $v'(x_k)$, что ее low-ребенком не является терминальный «0», и обозначим через $v'(x_l)$ вершину, являющуюся low-ребенком $v'(x_k)$. Но из $v'(x_l)$ достижима терминальная вершина «1», поэтому существует путь из корня в «1», задающий x_k значение 0, что противоречит условиям 1–2. Итак, если выполнены условия 1–2 и любой путь в B из корня в «1» задает x_k значение 1, то low-ребенок любой вершины, помеченной x_k, — это терминальный «0». Аналогично, если выполнены 1–2 и любой путь из корня B в «1» задает x_k значение 0, то high-ребенок любой вершины, помеченной x_k, — это терминальный «0». Лемма 1 доказана. ■

Лемма 2. Процедура проверки возникновения ROBDD-следствий в ROBDD B требует детерминированного времени, ограниченного сверху величиной $O(n \cdot |B|)$.

Доказательство. Сам по себе данный факт не является трудным. Поэтому представляется целесообразным привести в процессе доказательства полное описание алгоритма отслеживания ROBDD-следствий. Осуществляется это при помощи представленной ниже процедуры *check_ impl*().

```
check_impl(u - текущая вершина,
           inf_vector - вектор выведенных значений переменных)
{
    if u == 1:
        // u - терминальная вершина
        // проверить «целостность пути» (проверка условия 1)
        check_path_consistency(u, inf_vector);
        return;

    if not already_in_cache():
        // если такой вершины еще нет в кэше,
        // производим рекурсивный спуск по исходной ROBDD,
        // вызывая check_impl() для потомков данной вершины
        check_impl(low(u),  inf_vector);
        check_impl(high(u), inf_vector);

        // проверить «целостность пути» (проверка условия 1)
        check_path_consistency(u, inf_vector);

        // проверить наличие нулевого ребенка (проверка условия 2)
        check_for_zero_child(u, inf_vector, 0);

        // добавить в кэш запись о том, что данная вершина пройдена
        put_to_cache(u);

    return;
}
```

Как видно из представленного выше псевдокода, процедура *check_ impl*() рекурсивна и осуществляет полный обход B. Для каждой вершины u проверяются условия 1–2 относительно переменной, соответствующей u. Это осуществляется посредством следующих двух процедур: *check_ path_ consistency*() и *check_ for_ zero_ child*().

Процедура *check_ path_ consistency*(), находясь в вершине u, помеченной некоторой переменной $x_k, k \in \{2, \ldots, n\}$, просматривает родителей данной вершины. Если среди них имеется вершина, помеченная переменной $x_j, j < k - 1$, то принимается решение о невозможности возникновения ROBDD-следствий в отношении переменных $x_{j+1}, \ldots, x_{k-1}$.

Процедура *check_ for_ zero_ child*() проверяет для вершины u, помеченной переменной x_k, выполняется ли в ее отношении утверждение леммы 1.

В представленном псевдокоде использована специальная процедура кэширования, назначение которой в хранении информации о пройденных вершинах (ее использование исключает повторное прохождение вершин).

Результатом работы *check_ impl*() является *inf_ vector* — вектор длины n с компонентами из множества $\{-1, 0, 1\}$. На начальном шаге все компоненты данного вектора нулевые. Компонента с номером $k, k \in \{1, \ldots, n\}$, принимает значение 1 или -1, если для переменной x_k по ROBDD-следствию выведено значение соответственно $x_k = 1$ или $x_k = 0$.

Оценим сложность процедуры *check_ impl*(). Заметим, что данная процедура один раз обходит ROBDD B, модифицируя при необходимости вектор *inf_ vector*, при этом работа с произвольной вершиной u может потребовать (в процедуре *check_ path_ consistency*()) просмотра всего текущего вектора *inf_ vector*. Применение процедуры *check_ for_ zero_ child*() в отношении произвольной вершины требует времени $O(1)$. Таким образом, сложность процедуры *check_ impl*() ограничена сверху величиной $O(n \cdot |B|)$. Лемма 2 доказана. ■

Для дальнейшей работы потребуется модифицированный алгоритм *Restrict*, который позволяет осуществлять в ROBDD B подстановку набора значений некоторых переменных из X. Как отмечает Р. Брайант [23, 30], данный алгоритм имеет ту же сложность, что и обычный *Restrict*, то есть $O(|B|)$. Установим теперь справедливость следующей теоремы.

Теорема 4. Пусть в ROBDD B подставляются значения переменных

$$x_{i_1} = \alpha_{i_1}, \ldots, x_{i_m} = \alpha_{i_m}, m \leqslant n, \alpha_{i_j} \in \{0, 1\}, j \in \{1, \ldots, m\}.$$

Сложность детерминированной процедуры, осуществляющей данную подстановку и проверяющей наличие всевозможных ROBDD-следствий, ограничена сверху величиной $O(n \cdot |B|)$.

Доказательство. Используя результаты леммы 2 и модифицированный *Restrict*, можно записать процесс подстановки набора значений переменных в ROBDD с проверкой возникновения ROBDD-следствий в виде следующей процедуры:

```
assign(oldbdd        - исходная ROBDD,
       newbdd        - новая ROBDD,
       assign_vector - вектор подставляемых значений переменных,
       inf_vector    - вектор выведенных значений переменных)
{
    // подстановка в ROBDD oldbdd значений переменных
    // из вектора assign_vector
    restrict_m(oldbdd, newbdd, assign_vector);

    // проверка наличия ROBDD-следствий в ROBDD newbdd
    check_impl(newbdd, inf_vector);
}
```

Учитывая, что сложность процедуры *check_ impl*() ограничена величиной $O(n \cdot |B|)$ (лемма 2), а сложность процедуры *restrict_ m*(), реализующей модифицированный *Restrict*, ограничена величиной $O(|B|)$, заключаем, что верхняя граница сложности для процедуры *assign*() имеет вид $O(n \cdot |B|)$. Теорема 4 доказана. ■

Следствие 1. Если результатом применения процедур *check_ impl*() или *assign*() к B является ROBDD-следствие $x_k = \alpha_k, \alpha_k \in \{0, 1\}$, для некоторой $x_k \in X$, то подстановка в B значения $x_k = \alpha_k$ не может привести к возникновению нового ROBDD-следствия, индуцированного данной подстановкой.

Доказательство. Пусть в результате одной из перечисленных процедур в B возникло ROBDD-следствие $x_k = \alpha_k$. Не ограничивая общности, полагаем, что $\alpha_k = 1$.

В силу леммы 1 сделанное предположение означает, что low-ребенком всех вершин, помеченных x_k, является терминальный «0». Подстановка $x_k = 1$ в B для произвольной вершины $u(x_k)$ означает передачу high-ребенка вершины $u(x_k)$ вершинам, являющимся родителями $u(x_k)$. Но low-ребенок $u(x_k)$, то есть терминальный «0», при этом не передается никаким вершинам. Таким образом, подстановка $x_k = 1$ в B не может привести к *возникновению* в B вершины, ребенком которой является терминальный «0» (что не исключает наличия таких вершин, находившихся в B до осуществления этой подстановки). Аналогичные рассуждения справедливы и в предположении, что $\alpha_k = 0$. Следствие доказано. ■

Данный факт демонстрирует очень привлекательное свойство ROBDD, рассматриваемой в роли базы булевых ограничений. Напомним, что подстановка значения некоторой переменной в КНФ может приводить к выводу по правилу единичного дизъюнкта (unit clause) ряда индуцированных присвоений, подстановка которых также не исключает дальнейших срабатываний unit clause, и т. д. В этом смысл стратегии распространения булевых ограничений (BCP). В общем случае полная реализация BCP может приводить к многократному обходу КНФ, что сопряжено с существенными вычислительными затратами. Полученное свойство ROBDD означает, что порождаемые произвольной подстановкой ROBDD-следствия сами по себе новых ROBDD-следствий породить не могут и, таким образом, вся информация, индуцируемая данной подстановкой, извлекается в результате однократного обхода ROBDD.

На рис. 1 слева показана реализация BCP-стратегии применительно к КНФ $(x_1 \vee x_2)(\overline{x}_2 \vee x_3)(\overline{x}_3 \vee x_4)$ в результате подстановки $x_1 = 0$; справа — результат подстановки $x_1 = 0$ в ROBDD, представляющую булеву функцию, которая выражается той же самой КНФ; требуется единственный обход ROBDD.

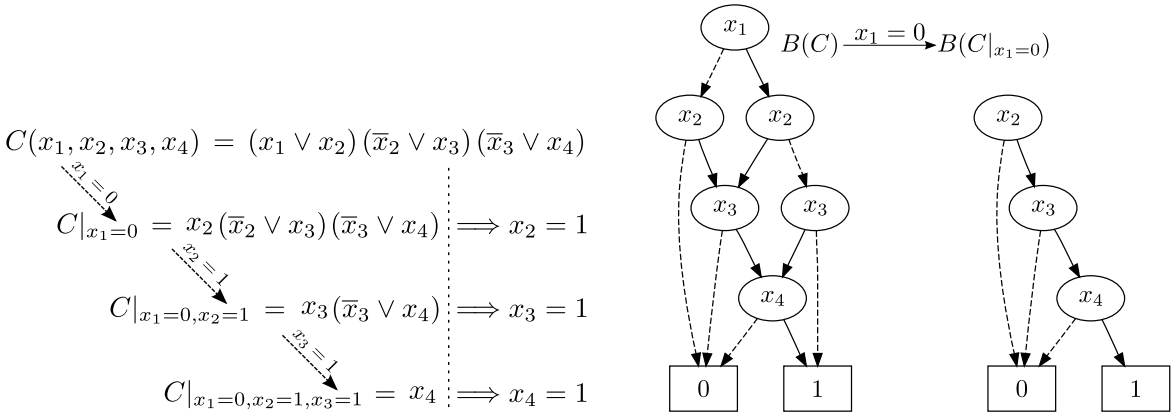

Рис. 1. Пример к следствию теоремы 4

Природа конфликтов в гибридном выводе более разнообразна, чем в DPLL-выводе. Во-первых, это обычные «DPLL-конфликты», возникающие в результате подстановок в КНФ-часть. Во-вторых, это «гибридный конфликт»: ситуация, когда в результате последовательности подстановок в ROBDD-части возникло ROBDD-следствие «$x_k = \alpha$», а в КНФ-части — выведенное по правилу единичного дизъюнкта присвоение «$x_k = \overline{\alpha}$». Однако в целом механизмы разбора конфликтов аналогичны механизмам, используемым в современных SAT-решателях (см. [16]), поэтому здесь дополнительное внимание этим процедурам не уделяется.

Еще одним полезным свойством гибридного вывода является возможность довольно естественной организации на ROBDD т. н. «отсроченных вычислений». Рассмотрим следующие условия, которые определяют ситуацию, в некотором роде двойственную ситуации возникновения ROBDD-следствия:

i) Для некоторой переменной $x_q \in X$ в ROBDD B любой путь π из корня в терминальную вершину «0» обязательно проходит через некоторую вершину, помеченную переменной x_q.

ii) Имеет место $|\Delta^0(x_q)| = 1$.

Установим справедливость следующей теоремы.

Теорема 5. Пусть B — произвольная ROBDD, и относительно некоторой переменной x_q в B справедливы условия *i* и *ii*. Тогда в ROBDD B невозможны ROBDD-следствия ни для каких переменных из множества $X \setminus \{x_q\}$. Трудоемкость процедуры проверки условий *i*–*ii* ограничена сверху величиной $O(n \cdot |B|)$.

Доказательство. Пусть в ROBDD B для некоторой переменной $x_q \in X$ выполняются условия *i* и *ii*. Не ограничивая общности, полагаем, что $\Delta^0(x_q) = \{1\}$. Используя рассуждения, полностью аналогичные тем, посредством которых была доказана лемма 1, можно показать, что в этом случае для любой вершины, помеченной переменной x_q, ее low-ребенком является терминальная вершина «1».

Теперь предположим, что существует такая переменная $x_p \in X \setminus \{x_q\}$, для которой выполнены условия 1–2 вывода некоторого ее ROBDD-следствия. Для данной переменной возможны следующие два варианта ее расположения относительно x_q в порядке означивания переменных в ROBDD B:

$$1 : x_1 \prec \ldots \prec x_q \prec \ldots \prec x_p \prec \ldots$$

$$2 : x_1 \prec \ldots \prec x_p \prec \ldots \prec x_q \prec \ldots$$

Рассмотрим первый случай. Из вышесказанного следует, что low-ребенком любой вершины, помеченной переменной x_q, является терминальная вершина «1». Данный факт означает, что существует обходной путь из корня ROBDD в терминальную вершину «1», не проходящий через вершины, помеченные переменной x_p, то есть для данной переменной вывод ее ROBDD-следствия невозможен.

Рассмотрим второй случай. Как было отмечено выше, условия 1 и 2 означают, что одним из детей любой вершины, помеченной переменной x_p, является «0». Но тогда существуют обходные пути из вершин, помеченных x_p, в «0», не проходящие через вершины, помеченные x_q, что противоречит предположению о выполнимости условий *i* и *ii*. Все проделанные рассуждения переносятся на случай $\Delta^0(x_q) = \{0\}$.

Отметим возможность совмещения процедур подстановки, проверки условий 1–2 и условий *i*–*ii*. В самом деле, проверка условий *i*–*ii* гарантируется использованием аналогов процедур *check_path_consistency*() и *check_for_zero_child*(), которые можно также «встроить» в процедуру *check_imp*(). При этом порядок сложности полученной процедуры останется прежним.

Все сказанное позволяет заключить, что сложность процедуры подстановки значений переменных в ROBDD с отслеживанием ситуаций возникновения ROBDD-следствий и выполнения относительно некоторых переменных (не обязательно одной) условий *i*–*ii* ограничена сверху величиной $O(n \cdot |B|)$. Теорема 5 доказана. ■

Данная теорема позволяет сформировать механизмы отсроченных вычислений при подстановке выведенных в процессе гибридного вывода значений некоторых переменных: если для текущей ROBDD B выполнены *i*–*ii* относительно x_q и из КНФ-части

выведено значение некоторой переменной x_k, $k \neq q$, нет смысла на данном этапе подставлять соответствующее значение в B — ничего нового выведено не будет. После присвоения или вывода из КНФ-части некоторого значения для x_q целесообразно осуществить в B подстановку сразу всех накопленных к этому моменту значений переменных, а также вывод всех возможных ROBDD-следствий, используя для этого процедуру *assign*().

3.2. Процедуры изменения порядка означивания переменных в ROBDD

Порядок означивания переменных существенным образом влияет на число вершин в ROBDD, поэтому одной из центральных проблем при использовании ROBDD в решении практических задач является выбор «хорошего» порядка. Известны различные подходы к этой проблеме. Так, в [24] предлагается изменять порядок означивания динамически — непосредственно в процессе построения ROBDD. Алгоритм, используемый для этой цели в [24], сходен в своей основе с известным методом пузырьковой сортировки (bubble sort, [31]). Сложность данного алгоритма в общем случае экспоненциальна. Более того, нет никакой гарантии, что ее использование даст в конечном счете порядок, лучший, чем заданный изначально.

В гибридном (SAT+ROBDD)-выводе часто есть веские основания считать некоторый порядок лучше текущего. Как правило, это связано со статистикой конфликтности, которая меняется в процессе вывода (переменные с меньшей конфликтностью на предыдущих этапах могут демонстрировать большую конфликтность на последующих). Тем самым возникает следующая задача.

Определение 7. Дана ROBDD $B(f)$, представляющая булеву функцию $f : \{0,1\}^n \to \{0,1\}$, построенная в соответствии с заданным порядком τ означивания переменных из множества $X = \{x_1, \ldots, x_n\}$. Требуется построить ROBDD $B'(f)$, представляющую ту же самую функцию, в которой порядок означивания переменных из X есть τ', отличный от τ. Назовем данную проблему проблемой модификации ROBDD в соответствии с новым порядком.

Для решения данной задачи далее предлагается подход, в корне отличающийся от метода, использующего пузырьковую сортировку. В его основе лежит возможность гарантированного решения за полиномиальное время задачи установки произвольной переменной из X на заданную позицию в новом порядке означивания переменных в ROBDD.

Пусть дана произвольная ROBDD $B(f)$, представляющая булеву функцию f от n переменных и построенная в соответствии с порядком их означивания

$$\tau : x_1 \prec x_2 \prec \ldots \prec x_n. \tag{4}$$

Рассмотрим произвольную подстановку на множестве $\{1, \ldots, n\}$

$$\sigma\,(\tau \to \tau') = \begin{pmatrix} 1 & 2 & \ldots & n \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \end{pmatrix},$$

$\{\alpha_1, \ldots, \alpha_n\} = \{1, \ldots, n\}$. Будем говорить, что данная подстановка задает изменение исходного порядка τ (4) на порядок τ', подразумевая под этим следующее. Столбец подстановки с номером $i \in \{1, \ldots, n\}$, имеющий вид $\begin{pmatrix} i \\ j \end{pmatrix}$, $j \in \{1, \ldots, n\}$, интерпретирует тот факт, что в новом порядке τ' переменная x_j будет находиться на позиции

с номером i. Например, исходный порядок $\tau : x_1 \prec x_2 \prec x_3$ подстановкой $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ изменяется на порядок $\tau' : x_2 \prec x_3 \prec x_1$. Исходному порядку в этом смысле соответствует тождественная подстановка $E_\tau = \begin{pmatrix} 1 & \dots & n \\ 1 & \dots & n \end{pmatrix}$. Также будем говорить, что порядок τ' определяется подстановкой $\sigma\,(\tau \to \tau')$ относительно порядка τ.

Определение 8. Дана ROBDD $B(f)$, построенная в соответствии с заданным порядком τ означивания переменных множества X. Требуется построить ROBDD $B'(f)$, в которой порядок означивания переменных определяется относительно исходного порядка τ произвольной подстановкой следующего вида:

$$\begin{pmatrix} 1 & \dots & i & \dots & n \\ \alpha_1 & \dots & j & \dots & \alpha_n \end{pmatrix}.$$

Данную проблему называем проблемой установки переменной x_j на позицию с номером i.

Теорема 6. Пусть $B(f)$ — произвольная ROBDD с порядком τ (4). Для произвольных $i, j \in \{1, \dots, n\}$ проблема установки переменной x_j на позицию с номером i решается детерминированным образом за время, ограниченное сверху величиной $O\left(|B(f)|^2\right)$.

Доказательство. Пусть дана ROBDD $B(f)$, представляющая булеву функцию от n переменных, образующих множество X. Считаем, что $B(f)$ построена в соответствии с порядком τ (4). Рассмотрим ROBDD B_j^0 и B_j^1, представляющие булевы функции $f^0 = f|_{x_j=0}, f^1 = f|_{x_j=1}$. Данные ROBDD являются результатом применения процедуры *Restrict* к $B(f)$, что требует времени, ограниченного сверху величиной $O\,(|B(f)|)$. Заметим, что B_j^0 и B_j^1 — ROBDD над множеством булевых переменных $X \setminus \{x_j\}$ с заданным на нем порядком

$$\tau^* : x_1 \prec \dots \prec x_{j-1} \prec x_{j+1} \prec \dots \prec x_n.$$

Построим две ROBDD $(B(x_j), B(\overline{x}_j))$ (рис. 2):

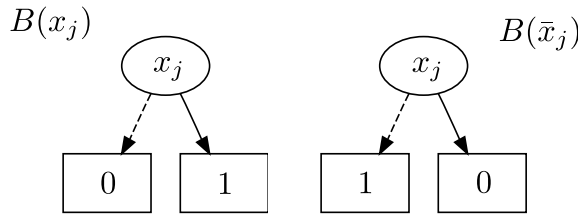

Рис. 2. Элементарные ROBDD

При помощи алгоритма *Apply* построим в соответствии с порядком означивания переменных

$$\tau' : x_1 \prec \dots \prec x_{i-1} \prec x_j \prec x_i \prec \dots \prec x_{j-1} \prec x_{j+1} \prec \dots \prec x_n \tag{5}$$

следующие ROBDD:

$$B^0 = Apply\left(B(\overline{x}_j) \cdot B_j^0\right), B^1 = Apply\left(B(x_j) \cdot B_j^1\right).$$

Рассмотрим ROBDD

$$B'(f) = Apply\left(B^0 \vee B^1\right), \tag{6}$$

построенную в соответствии с порядком τ' (5).

Заметим, что B^0 — ROBDD-представление функции $\overline{x}_j \cdot f|_{x_j=0}$, а B^1 — ROBDD-представление функции $x_j \cdot f|_{x_j=1}$. Таким образом, (6) можно рассматривать как «альтернативный» вид разложения Шеннона булевой функции f по переменной x_j. Тем самым ROBDD $B'(f)$ представляет булеву функцию f в соответствии с порядком (5) означивания переменных множества X.

Поскольку $|B(\overline{x}_j)| = |B(x_j)| = 3$, то сложность построения ROBDD B^0 и B^1 заведомо ограничена сверху величиной $O(|B(f)|)$. Это есть следствие оценки трудоемкости алгоритма *Apply* и того факта, что порядок τ^* подчинен порядку τ' (то есть из $x' \prec_{\tau^*} x''$ следует, что $x' \prec_{\tau'} x''$). В силу сказанного, число вершин в ROBDD B^0 и B^1 также ограничивается сверху величиной $O(|B(f)|)$. Отсюда (снова используя оценку сложности алгоритма *Apply*) имеем: сложность построения ROBDD $B'(f)$ ограничивается сверху величиной $O\left(|B(f)|^2\right)$. Теорема 6 доказана. ■

Данная теорема позволяет предложить следующий алгоритм изменения фиксированного начального порядка τ на новый порядок τ'.

Следствие 2. Пусть $B(f)$ — ROBDD, представляющая булеву функцию f от n переменных в соответствии с порядком означивания переменных τ. Проблема модификации $B(f)$ в соответствии с произвольным порядком τ' сводится к l-кратному ($l \leqslant n$) решению проблемы установки переменной на позицию с заданным номером.

Доказательство. Докажем данный факт, предъявив в явном виде соответствующий алгоритм. Пусть имеется ROBDD $B(f)$, представляющая булеву функцию f от n переменных в соответствии с некоторым порядком означивания переменных τ, и рассматривается некоторый порядок τ', отличный от τ. Тем самым τ' определяется относительно τ некоторой подстановкой, отличной от E_τ. Обозначим вторую строку данной подстановки через $\alpha_{\tau'} = (\alpha_1, \ldots, \alpha_n)$. Описываемый ниже алгоритм получает на входе ROBDD $B(f)$, а также порядки τ и τ'.

0. Пусть $m = 1$.
1. Если $\alpha_m = m$, то полагаем $m = m+1$ и переходим к шагу 1; в противном случае переходим к шагу 2.
2. Решаем проблему установки переменной x_{α_m} на позицию с номером m в текущей ROBDD.
3. Если $m < n$, то полагаем $m = m + 1$ и переходим к шагу 1; в противном случае завершаем выполнение алгоритма.

Теперь покажем, что описанный алгоритм решает проблему модификации ROBDD в соответствии с новым порядком τ'. На каждой итерации алгоритма переменная x_{α_m} устанавливается в ROBDD на позицию с номером m. При этом, как следует из доказательства теоремы 6, все переменные, номера которых меньше m (т. е. установленные на предыдущих итерациях), своего порядка в текущей ROBDD не изменяют. Обобщая сказанное, заключаем, что для корректного решения проблемы модификации ROBDD в соответствии с новым порядком τ' достаточно не более n раз решить проблему установки переменной на заданную позицию, что и делает описанный алгоритм. Следствие доказано. ■

Обратим внимание на то, что данный алгоритм, вообще говоря, не является полиномиальным. Однако фактически он разбит на $l, l \leqslant n$, процедур, каждая из которых

устанавливает некоторую переменную на заданную позицию и сама по себе работает за полиномиальное от числа вершин в подаваемой ей на вход ROBDD время. На практике алгоритм показывает хорошие результаты в задачах модификации порядка ROBDD, возникающих в гибридном (SAT+ROBDD)-выводе. Его эффективность особенно заметна при незначительных отличиях в новом и старом порядках (когда большая часть переменных не меняет своего местоположения). Следует подчеркнуть, что для процедуры, использующей пузырьковую сортировку, в общем случае невозможны полиномиальные оценки даже в отношении задачи установки переменной на заданную позицию.

Заключение

Настоящая работа посвящена гибридному (SAT+ROBDD)-выводу, используемому в задачах обращения дискретных функций. В соответствии с гибридной стратегией некоторый алгоритм на основе DPLL действует в отношении КНФ $C(f)$, кодирующей задачу обращения полиномиально вычислимой функции f в некоторой точке. После каждого рестарта вместо чистки базы конфликтных дизъюнктов используется процедура построения ROBDD-представления характеристической функции системы вида (3). Дальнейший вывод идет как на исходной КНФ, так и на ROBDD, представляющей соответствующую базу накопленных ограничений.

Построены аналоги основных операций с булевыми ограничениями, используемыми в нехронологическом DPLL-выводе: аналог подстановки с отслеживанием срабатывания правила единичного дизъюнкта — процедура *assign*() (теорема 4); аналог CL-процедуры — применение *Apply* к текущей ROBDD и новому ограничению-дизъюнкту; аналог механизма действия структур watched literals определяется условиями *i*–*ii* и теоремой 5. Для всех предложенных алгоритмов приведены сложностные оценки. Кроме этого, предложен новый алгоритм модификации порядка в заданной ROBDD, показывающий хорошие практические результаты.

Отметим также, что гибридный (SAT+ROBDD)-подход к обращению дискретных функций был программно реализован с использованием стандарта MPI [32] в виде решателя, функционирующего в распределенной вычислительной среде. В решателе используется межпроцессорный обмен накапливаемыми ограничениями, передаваемыми в виде ROBDD. В этом его принципиальное отличие от недавних зарубежных разработок (см. [33]), в которых при решении SAT-задач также используется межпроцессорное взаимодействие. Построенный решатель показал высокую эффективность на задачах обращения некоторых криптографических функций.

Авторы выражают глубокую благодарность А. Е. Хмельнову за внимание к представленным в статье исследованиям, дружеские советы и конструктивные замечания.

ЛИТЕРАТУРА

1. `http://www.satlive.org`
2. `www.isa.ewi.tudelft.nl/jsat`
3. *Hachtel G. D., Somenzi F.* Logic synthesis and verification algorithms. Kluwer Ac. Publ, 2002.
4. *Кларк Э. М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking. М.: МЦНМО, 2002.
5. *Гаранина Н. О., Шилов Н. В.* Верификация комбинированных логик знаний, действий и времени в моделях // Системная информатика. 2005. Вып. 10. С. 114–173.

6. *Семенов А. А., Заикин О. С., Беспалов Д. В., Ушаков А. А.* SAT-подход в криптоанализе некоторых систем поточного шифрования // Вычислительные технологии. 2008. Т. 13. № 6. С. 134–150.
7. Системная компьютерная биология / под ред. Н. А. Колчанова, С. С. Гончарова, В. А. Лихошвая, В. А. Иванисенко. Новосибирск: Изд-во СО РАН, 2008. 767 с.
8. *Davis M., Logemann G., Loveland D.* A machine program for theorem proving // Comm. ACM. 1962. V. 5. P. 394–397.
9. *Marqeus-Silva J. P., Sakallah K. A.* GRASP: A search algorithm for propositional satisfiability // IEEE Trans. Comp. 1999. V. 48. No. 5. P. 506–521.
10. *Moskewicz M., Madigan C., Zhao Y., et al.* Chaff: Engineering an Efficient SAT Solver // Proc. Design Automat. Conf. (DAC). 2001. P. 530–535.
11. `http://minisat.se/MiniSat.html` — MiniSat.
12. *Семенов А. А., Заикин О. С.* Неполные алгоритмы в крупноблочном параллелизме комбинаторных задач // Вычислительные методы и программирование. 2008. Т. 9. С. 108–118.
13. *Семенов А. А.* Декомпозиционные представления логических уравнений в задачах обращения дискретных функций // Изв. РАН. Теория и системы управления. 2009. № 5. С. 47–61.
14. *Закревский А. Д.* Логические уравнения. Минск: Наука и техника, 1975.
15. *Семенов А. А., Беспалов Д. В.* Технологии решения многомерных задач логического поиска // Вестник Томского госуниверситета. Приложение. 2005. № 14. С. 61–73.
16. *Zhang L., Madigan C., Moskewicz M., Malik S.* Efficient conflict driven learning in a boolean satisfiability solver // Proc. Intern. Conf. on Computer Aided Design (ICCAD). 2001. P. 279–285.
17. *Lynce I., Marques-Silva J. P.* Efficient data structures for backtrack search SAT solvers // Ann. Mathem. Artific. Intellig. 2005. V. 43. P. 137–152.
18. *Семенов А. А., Заикин О. С., Беспалов Д. В. и др.* Решение задач обращения дискретных функций на многопроцессорных вычислительных системах // Труды Четвертой Междунар. конф. PACO'2008 (Москва, 26–29 октября 2008).М., 2008. С. 152–176.
19. *Заикин О. С., Семенов А. А.* Технология крупноблочного параллелизма в SAT-задачах // Проблемы управления. 2008. № 1. С. 43–50.
20. *Семенов А. А.* Трансляция алгоритмов вычисления дискретных функций в выражения пропозициональной логики // Прикладные алгоритмы в дискретном анализе. Сер.: Дискретный анализ и информатика. 2008. Вып. 2. С. 70–98.
21. *Cook S. A.* The complexity of theorem-proving procedures // Proc. 3rd Ann. ACM Symp. on Theory of Computing, ACM. 1971. P. 151–159. [Пер.: *Кук С. А.* Сложность процедур вывода теорем // Кибернетический сборник. Новая серия. 1975. Вып. 12. С. 5–15.]
22. *Lee C. Y.* Representation of Switching Circuits by Binary-Decision Programs // Bell Systems Technical J. 1959. V. 38. P. 985–999.
23. *Bryant R. E.* Graph-Based Algorithms for Boolean Function Manipulation // IEEE Trans. Comp. 1986. V. 35. No. 8. P. 677–691.
24. *Meinel Ch., Theobald T.* Algorithms and Data Structures in VLSI-Design: OBDD-Foundations and Applications. Springer Verlag, 1998.
25. *Хмельнов А. Е., Игнатьев А. С., Семенов А. А* Двоичные диаграммы решений в логических уравнениях и задачах обращения дискретных функций // Вестник НГУ. Сер.: Информационные технологии. 2009. Т. 7. № 4. С. 36–52.
26. *Семенов А. А.* О преобразованиях Цейтина в логических уравнениях // Прикладная дискретная математика. 2009. № 4. С. 28–50.

27. *Цейтин Г. С.* О сложности вывода в исчислении высказываний // Записки научных семинаров ЛОМИ АН СССР. 1968. Т. 8. С. 234–259.
28. *Beame P., Kautz H., Sabharwal A.* Understanding the power of clause learning // Proc. Of 18th Intern. Joint Conf. on Artificial Intellegence (IJCAI). 2003. P. 1194–1201.
29. *Игнатьев А. С., Семенов А. А., Хмельнов А. Е.* Использование двоичных диаграмм решений в задачах обращения дискретных функций // Вестник Томского госуниверситета. Сер.: Управление, вычислительная техника, информатика. 2009. № 1(6). С. 115–129.
30. *Bryant R. E.* Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams // ACM Comp. Surv. 1992. V. 24. No. 3. P. 293–318.
31. *Левитин А.* Алгоритмы. Введение в разработку и анализ. М.: Вильямс, 2006.
32. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002.
33. *Schubert T., Lewis M., Becker B.* PaMiraXT: Parallel SAT Solving with Threads and Message Passing // J. Satisf., Bool. Mod. Comp. 2009. V. 6. P. 203–222.

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

DOI 10.17223/20710410/7/8

УДК 519.17

МИНИМАЛЬНЫЕ РЕБЕРНЫЕ РАСШИРЕНИЯ НЕКОТОРЫХ ПРЕДПОЛНЫХ ГРАФОВ

М. Б. Абросимов

Саратовский государственный университет им. Н. Г. Чернышевского, г. Саратов, Россия

E-mail: mic@rambler.ru

Рассматриваются минимальные реберные k-расширения предполных графов — графов, в которых есть вершина, смежная со всеми остальными. Доказывается лемма, позволяющая оценить предельные значения k, при которых предполный граф может иметь минимальные реберные k-расширения, а также указать их общий вид. Дается полное описание всех минимальных реберных k-расширений для предполных графов, являющихся соединением полного графа с вполне несвязным графом, цепью и циклом.

Ключевые слова: *предполный граф, минимальное реберное расширение, отказоустойчивая реализация.*

Введение

Неориентированным графом (далее просто *графом*) называется пара $G = (V, \alpha)$, где α — симметричное и антирефлексивное отношение на множестве вершин V, называемое *отношением смежности*. Если $(u, v) \in \alpha$, то говорят, что вершины u и v *смежны*, и эти вершины соединены ребром (u, v). При этом (u, v) и (v, u) — это одно и то же ребро, которое обозначают $\{u, v\}$. Также говорят, что ребро $\{u, v\}$ *инцидентно* вершинам u и v. Основные определения даются согласно [1].

Предполным графом называется граф, имеющий хотя бы одну *полную* вершину, то есть вершину, смежную со всеми остальными. Граф, все вершины которого являются полными, называется *полным графом* и обозначается K_n. В общем виде предполный граф можно записать как соединение одновершинного графа K_1 и некоторого n-вершинного графа G_n: $K_1 + G_n$. Если в предполном графе p полных вершин, то такой граф можно записать как $K_p + G$.

Под *соединением* двух графов $G_1 = (V_1, \alpha_1)$ и $G_2 = (V_2, \alpha_2)$, не имеющих общих вершин, понимается граф

$$G_1 + G_2 = (V_1 \cup V_2, \alpha_1 \cup \alpha_2 \cup V_1 \times V_2 \cup V_2 \times V_1).$$

Под *объединением* двух графов $G_1 = (V_1, \alpha_1)$ и $G_2 = (V_2, \alpha_2)$ понимается граф

$$G_1 \cup G_2 = (V_1 \cup V_2, \alpha_1 \cup \alpha_2).$$

В 1976 г. Хейз в работе [2] предложил основанную на графах модель для исследования отказоустойчивости систем.

Технической системе Σ сопоставляется помеченный граф $G(\Sigma)$, вершины которого соответствуют элементам системы Σ, ребра — связям между элементами, а метки

указывают тип элементов. Под отказом элемента технической системы Σ понимается удаление соответствующей ему вершины из графа системы $G(\Sigma)$ и всех связанных с ней ребер. Говорят, что система Σ^* является *k-отказоустойчивой реализацией* системы Σ, если отказ любых k элементов системы Σ^* приводит к графу, в который можно вложить граф системы Σ с учетом меток вершин. Построение k-отказоустойчивой реализации системы Σ можно представить себе как введение в нее определенного числа новых элементов и связей. При этом предполагается, что в нормальном режиме работы избыточные элементы и связи маскируются, а в случае отказа происходит реконфигурация системы до исходной структуры.

Пусть в системе Σ встречается t различных типов элементов. Очевидно, что любая ее k-отказоустойчивая реализация должна содержать не менее k дополнительных элементов каждого типа. Легко видеть, что такого числа дополнительных элементов достаточно для построения k-отказоустойчивой реализации системы Σ. В самом деле, добавим k элементов каждого типа и соединим их все между собой и с элементами системы Σ. Тогда любой отказавший элемент можно будет заменить одним из добавленных элементов соответствующего типа. Построенную таким образом k-отказоустойчивую реализацию можно назвать *тривиальной*.

k-Отказоустойчивая реализация Σ^* системы Σ, состоящей из t элементов различного типа, называется *оптимальной*, если система Σ^* отличается от системы Σ на k элементов каждого из t типов системы Σ и среди всех k-отказоустойчивых реализаций с тем же числом элементов система Σ^* имеет наименьшее число связей.

На практике элементы технических систем часто оказываются однотипными. При исследовании отказоустойчивости в подобных системах метки элементов опускаются и в качестве графа системы рассматривается граф без меток. В этом случае оптимальная k-отказоустойчивая реализация будет содержать в точности k дополнительных элементов.

Хейз предложил процедуры построения оптимальной k-отказоустойчивой реализации для цепи, цикла и помеченного дерева. Позднее Хейз совместно с Харари в работе [3] обобщили модель на случай отказов связей между элементами, предложив понятие *реберной отказоустойчивости*. Модель отказоустойчивости, в которой рассматриваются только отказы элементов, было предложено называть *вершинной отказоустойчивостью* (см. [4]).

Понятие минимального k-расширения является общеграфовым аналогом для конструкций отказоустойчивых реализаций. Исключая понятие «отказа» элемента или связи системы, рассматриваются части графа, получающиеся удалением тех или иных его элементов. Оказалось, что задачи проверки вершинной и реберной отказоустойчивости являются NP-полными (см. [5]), поэтому интерес представляет аналитическое описание минимальных вершинных и реберных k-расширений для различных классов графов.

В работе [6] удалось полностью описать вид всех минимальных вершинных k-расширений предполных графов при любом натуральном k. Для реберных расширений такой схемы, позволяющей строить все или хотя бы одно минимальное реберное k-расширение любого предполного графа, не известно и, видимо, получить ее не удастся. В данной работе предлагается решение проблемы для некоторых частных случаев предполных графов: для соединений полного графа с вполне несвязным графом, с цепью и с циклом.

1. Леммы о реберных k-расширениях предполных графов

Граф $G^* = (V^*, \alpha^*)$ называется *минимальным реберным k-расширением* (k — натуральное) n-вершинного графа $G = (V, \alpha)$, если выполняются следующие условия:

1) G^* является реберным k-расширением G, то есть граф G допускает вложение в каждый граф, получающийся из G^* удалением любых его k ребер;

2) G^* содержит n вершин, то есть $|V^*| = |V|$;

3) α^* имеет минимальную мощность при выполнении условий 1 и 2.

В отличие от минимальных вершинных k-расширений, минимальные реберные k-расширения существуют не при всех значениях k. Например, полный граф K_n (неориентированный или ориентированный, с петлями или без) не имеет минимального реберного k-расширения ни при каких значениях k. Однако, если разрешить в графе кратные дуги, то тогда любой граф будет иметь минимальные реберные k-расширения. Будем использовать обозначение $G - \{u, v\}$ для графа, получающегося из графа G удалением ребра $\{u, v\}$.

Сформулируем и докажем лемму о реберных k-расширениях произвольных предполных графов.

Лемма 1. Пусть n-вершинный предполный граф G имеет в точности p полных вершин. Если G имеет минимальное реберное k-расширение, то оно содержит не менее $p + 2k$ полных вершин.

Доказательство. Пусть G — предполный граф с p полными вершинами, а G^* — его минимальное реберное k-расширение. Рассмотрим граф G', получающийся из G^* удалением следующих k ребер: в качестве очередного ребра удаляем

а) ребро, соединяющее две полные вершины. Если таких ребер нет, то

б) ребро, инцидентное полной вершине (это возможно, только если останется единственная полная вершина). Если нет ребер из пунктов *а* и *б*, то

в) любое ребро.

Очевидно, что если среди выбранных ребер окажется хотя бы одно ребро из пунктов *б* или *в*, то получившийся граф не будет предполным и не будет допускать вложения исходного графа G. Таким образом, граф G' должен содержать не менее p полных вершин и получаться из графа G^* удалением k ребер, соединяющих полные вершины графа G^*. После удаления каждого такого ребра количество полных вершин уменьшается на две, что и доказывает утверждение. ■

Лемма 1 позволяет оценить, при каких значениях k предполный граф может иметь минимальные реберные k-расширения, а также указать общий вид этих минимальных реберных k-расширений.

Следствие 1. Пусть n-вершинный предполный граф G имеет в точности p полных вершин, тогда G не имеет минимальных реберных k-расширений при $k > \dfrac{n-p}{2}$.

Следствие 2. Пусть n-вершинный предполный граф G вида $K_p + G_n$ имеет минимальное реберное k-расширение G^*. Тогда граф G^* имеет вид $K_{p+2k} + G_{n-2k}$, где G_{n-2k} — подходящий $(n-2k)$-вершинный граф, такой, что граф $K_{2k} + G_{n-2k}$ является реберным k-расширением графа G_n.

Доказательство.

Первая часть утверждения непосредственно следует из леммы.

Докажем, что если граф $K_{p+2k} + G_{n-2k}$ является минимальным реберным k-расширением графа $K_p + G_n$, то граф $K_{2k} + G_{n-2k}$ является реберным k-расширением

графа G_n. Предположим, что это неверно. Это означает, что можно выбрать k ребер графа $K_{2k} + G_{n-2k}$, таких, что получающийся после их удаления граф H не будет допускать вложения графа G_n. Но тогда удаление этих же ребер в графе $K_p + (K_{2k} + G_{n-2k}) = K_{p+2k} + G_{n-2k}$ даст граф $K_p + H$, который не будет допускать вложения графа $K_p + G_n$, а это противоречит предположению. ■

Следствие 2 позволяет свести задачу поиска минимальных реберных k-расширений графов вида $K_p + G_n$ к поиску реберных k-расширений графа G_n среди графов вида $K_{2k} + G_{n-2k}$, а также оценить количество дополнительных ребер.

Следствие 3. Пусть n-вершинный предполный граф G вида K_p+G_n имеет минимальное реберное k-расширение вида $K_{p+2k}+G_{n-2k}$. Тогда количество дополнительных ребер равно

$$2nk - 2k^2 - k + m^* - m, \tag{1}$$

где m — число ребер графа G_n, а m^* — число ребер графа G_{n-2k}.

Доказательство. Определим количество ребер графа $K_p + G_n$.

Граф K_p содержит $\dfrac{p(p-1)}{2}$ ребер.

Граф G_n содержит m ребер.

Каждая вершина из K_p соединена с каждой из n вершин графа G, что дает pn ребер.

Суммируя, получаем, что граф $K_p + G_n$ имеет $\dfrac{p(p-1)}{2} + pn + m$ ребер.

Аналогично определим количество ребер графа $K_{p+2k} + G_{n-2k}$:

$$\begin{gathered}\frac{(p+2k)(p+2k-1)}{2} + (p+2k)(n-2k) + m^* = \\ = \frac{p^2 + 4pk - p + 4k^2 - 2k}{2} + pn + 2nk - 2kp - 4k^2 + m^* = \\ = \frac{p(p-1)}{2} + pn - 2k^2 - k + 2nk + m^*.\end{gathered}$$

Вычитая, получим

$$2nk - 2k^2 - k + m^* - m.$$

■

Докажем еще одно полезное утверждение относительно минимальных реберных k-расширений произвольных графов.

Лемма 2. Пусть n-вершинный граф G имеет минимальное реберное k_1-расширение, которым является полный граф K_n. Тогда при $k > k_1$ граф G не имеет минимальных реберных k-расширений.

Доказательство. Для доказательства достаточно заметить, что если некоторый граф G^* является минимальным реберным k-расширением графа G, то любой граф H, получающийся из G^* удалением произвольного ребра, будет реберным $(k-1)$-расширением графа G. Из этого, в частности, следует, что любое минимальное реберное $(k+1)$-расширение графа G содержит больше ребер, чем минимальное реберное k-расширение графа G. Наконец, так как полный граф K_n имеет максимальное число ребер среди всех n-вершинных графов, то это и доказывает лемму. ■

2. Соединение полного и вполне несвязного графов: $K_m + O_n$

n-Вершинный граф без ребер называется *вполне несвязным* и обозначается O_n. Рассмотрим графы вида $K_m + O_n$.

Теорема 1. При $k \leqslant n/2$ граф $K_{m+2k}+O_{n-2k}$ является единственным с точностью до изоморфизма минимальным реберным k-расширением графа K_m+O_n. При $k > n/2$ граф $K_m + O_n$ не имеет минимальных реберных k-расширений.

Доказательство. По лемме 1 число полных вершин минимального реберного k-расширения графа $K_m + O_n$ должно быть не менее $m + 2k$. По следствию 1 из леммы 1 при $n < 2k$ граф $K_m + O_n$ не имеет минимальных реберных k-расширений. По следствию 2 из леммы 1 при $n \geqslant 2k$ граф $K_m + O_n$ может иметь минимальные реберные k-расширения вида $K_{m+2k} + G_{n-2k}$, где G_{n-2k} — подходящий $(n - 2k)$-вершинный граф.

Убедимся, что при $n \geqslant 2k$ граф вида $K_{m+2k} + O_{n-2k}$ является реберным k-расширением графа $K_m + O_n$. Все ребра графа $K_{m+2k} + O_{n-2k}$ делятся на две группы:

1) ребра, соединяющие две полные вершины;

2) ребра, соединяющие одну полную вершину с вершиной из части O_{n-2k}.

Удаление ребра первого типа сокращает количество полных вершин на две, а удаление ребра второго типа — на одну. После удаления k любых ребер сохранится не менее m полных вершин.

Заметим, что из всех кандидатов на роль графа G_{n-2k} граф O_{n-2k} имеет минимально возможное число ребер. Следовательно, граф $K_{m+2k} + O_{n-2k}$ является и минимальным реберным k-расширением графа $K_m + O_n$ при $n \geqslant 2k$. ■

Следствие 4. Число дополнительных ребер в минимальном реберном k-расширении графа $K_m + O_n$ при $k \leqslant n/2$ составляет $2nk - 2k^2 - k$.

Доказательство. Воспользуемся следствием 3. В нашем случае графы G_n и G_{n-2k} являются вполне несвязными графами, поэтому $m = m^* = 0$. Подставляя в формулу (1), получим искомое значение числа дополнительных ребер: $2nk - 2k^2 - k$. ■

Замечание 1. Граф вида $K_1 + O_n$ называется *звездой* или *звездным графом*. На рис. 1 показаны минимальные реберные 1-расширения звезд $K_1 + O_2$ и $K_1 + O_3$. Для наглядности полные вершины помечены черным.

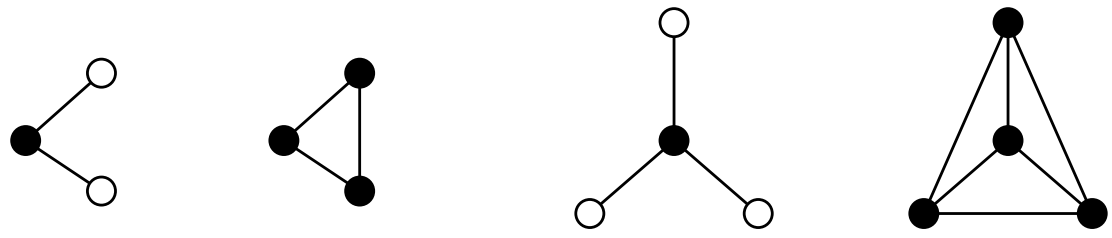

Рис. 1. Звезды $K_1 + O_2$ и $K_1 + O_3$ и их минимальные реберные 1-расширения

3. Соединение полного графа и цепи: $K_m + P_n$

Путем называется последовательность ребер в графе, такая, что конец одного ребра является началом другого. Граф называется *связным*, если между двумя любыми

его вершинами существует путь. Любой максимальный связный подграф графа называется *компонентой связности* или просто *компонентой*. Граф называется *k-реберно связным*, если не менее k ребер необходимо удалить, чтобы нарушить его связность. Например, полный граф K_n является $(n-1)$-реберно связным.

Цепью P_n называется граф $G = (V, \alpha)$, где $V = \{v_1, v_2, \ldots, v_n\}$ и $\alpha = \{(v_i, v_j) : |i - j| = 1\}$. Одновершинная цепь P_1 называется *тривиальной*. Будем говорить, что n-вершинный граф G *может быть покрыт* (не более чем) p цепями, если существует n-вершинный граф, являющийся объединением (не более чем) p цепей и вкладывающийся в граф G. Например, полный граф K_n может быть покрыт одной цепью P_n, а вполне несвязный граф O_n может быть покрыт n цепями (тривиальными).

Рассмотрим графы вида $K_m + P_n$.

При $n \leqslant 2$ граф $K_m + P_n$ изоморфен графу K_{m+n} и не имеет минимальных реберных k-расширений ни при каких значениях k. Пусть далее $n > 2$. Рассмотрим граф $K_{m+2} + G$, где G — $(n-2)$-вершинный граф, такой, что любой граф, получающийся из G удалением одного ребра, можно покрыть не более чем тремя цепями. Обозначим для определенности вершины подграфа K_{m+2} графа $K_{m+2} + G$ через $v_1, \ldots, v_{m+2}$. Покажем, что граф $K_{m+2} + G$ является реберным 1-расширением для графа $K_m + P_n$. Заметим, что ребра графа $K_{m+2} + G$ можно разделить на три типа:

1) ребра, соединяющие полные вершины $v_1, \ldots, v_{m+2}$;

2) ребра, соединяющие полную вершину из $v_1, \ldots, v_{m+2}$ и вершину подграфа G;

3) ребра, соединяющие вершины подграфа G.

Рассмотрим удаление из графа $K_{m+2} + G$ ребра типа 1. Пусть для определенности удалено ребро $\{v_1, v_2\}$. Поскольку часть графа G допускает покрытие не более чем тремя цепями, то, очевидно, и сам граф G может быть покрыт тремя цепями. Рассмотрим худший случай, когда для покрытия необходимы три цепи. Пусть цепи P_{n_1}, P_{n_2} и P_{n_3} образуют покрытие графа G. Вложение графа $K_m + P_n$ в граф $(K_{m+2} + G) - \{v_1, v_2\}$ возможно следующим образом. Полные вершины — $v_3, \ldots, v_{m+2}$, а оставшиеся вершины образуют цепь: $P_{n_1} v_1 P_{n_2} v_2 P_{n_3}$.

Рассмотрим удаление из $K_{m+2} + G$ ребра типа 2. Пусть цепи P_{n_1}, P_{n_2} и P_{n_3} образуют покрытие графа G. Пусть для определенности удалено ребро, соединяющее полную вершину v_1 с некоторой вершиной w цепи P_{n_1}. Вложение графа $K_m + P_n$ в граф $(K_{m+2} + G) - \{v_1, w\}$ возможно следующим образом. Полные вершины — $v_3, \ldots, v_{m+2}$, а оставшиеся вершины образуют цепь: $P_{n_1} v_2 P_{n_2} v_1 P_{n_3}$.

Рассмотрим удаление из $K_{m+2} + G$ ребра типа 3. Пусть для определенности удалено ребро, соединяющее вершины v и w графа G. По условию граф $G - \{v, w\}$ допускает покрытие не более чем тремя цепями, и пусть это снова будут цепи P_{n_1}, P_{n_2} и P_{n_3}. Вложение графа $K_m + P_n$ в $(K_{m+2} + G) - \{v, w\}$ возможно следующим образом. Полные вершины — $v_3, \ldots, v_{m+2}$, а оставшиеся вершины образуют цепь: $P_{n_1} v_1 P_{n_2} v_2 P_{n_3}$.

Таким образом, граф $K_{m+2} + G$ является реберным 1-расширением графа $K_m + P_n$.

Заметим, что условие покрытия графа G с удаленным ребром тремя цепями является существенным, так как более трех цепей нельзя объединить в одну с помощью двух дополнительных полных вершин. В самом деле, граф $K_2 + G$ по следствию 2 из леммы 1 должен быть реберным 1-расширением цепи P_n. Если граф G с удаленным ребром может быть покрыт тремя цепями, то, как мы уже видели, граф $K_2 + G$ является реберным 1-расширением цепи P_n. Если же это неверно, т. е. граф G с удаленным ребром e может быть покрыт не менее чем 4 цепями, то цепь P_n нельзя будет вложить в граф $K_2 + (G - e)$.

Исследуем свойства графа G:

1) G содержит $n-2$ вершин;

2) любой граф, получающийся из G удалением одного ребра, можно покрыть не более чем тремя цепями;

3) граф, получающийся из G удалением одного ребра, имеет не более трех компонент связности, а следовательно, и сам граф G имеет не более трех компонент связности.

Рассмотрим граф, являющийся объединением двух цепей: $P_{n_1} \cup P_{n_2}$. При удалении любого ребра такой граф распадется на три цепи. Таким образом, объединение двух цепей может быть использовано в качестве графа G, причем число ребер в таком графе есть $n-4$. Заметим, что $P_{n_1} \cup P_{n_2}$ отличается в точности на одно ребро от объединения трех цепей. С учетом следствия 2 из леммы 1 получаем, что минимальное реберное 1-расширение графа $K_m + P_n$ имеет вид $K_{m+2} + G$, где G — $(n-2)$-вершинный граф, каждая часть которого, получающаяся удалением одного ребра, может быть покрыта не более чем тремя цепями. Условиям 2, 3 удовлетворяют следующие n-вершинные графы:

— Объединение двух цепей: $P_{n_1} \cup P_{n_2}$, $n_1 + n_2 = n$, $1 \leqslant n_1 \leqslant n_2 \leqslant n-1$. Таких графов будет $[n/2]$.
— При $n \geqslant 5$: цикл и две изолированные вершины: $O_2 \cup C_{n-2}$.
— При $n = 5$: $O_1 \cup K_{1,3}$.

Из всего сказанного получается

Теорема 2. Относительно минимальных реберных 1-расширений предполных графов вида $K_m + P_n$ справедливо следующее:

— при $n = 1, 2$: минимальных реберных 1-расширений нет;
— при $n = 3$: минимальное реберное 1-расширение единственно и имеет вид K_{m+n};
— при $n \geqslant 4$: существует $([n/2] - 1)$ минимальных реберных 1-расширений вида

$$K_{m+2} + (P_{n_1} \cup P_{n_2}), \quad n_1 + n_2 = n - 2, \quad 1 \leqslant n_1 \leqslant n_2 \leqslant n - 3;$$

— при $n = 7$: имеется минимальное реберное 1-расширение вида $K_{m+2} + (O_1 \cup K_{1,3})$;
— при $n \geqslant 7$: имеется минимальное реберное 1-расширение вида $K_{m+2} + (O_2 \cup C_{n-4})$.

Следствие 5. Число дополнительных ребер в минимальном реберном 1-расширении графа $K_m + P_n$ при $n > 3$ составляет $2n - 6$.

Доказательство. Воспользуемся следствием 3.

Так как граф G_n в данном случае — это цепь P_n, то $m = n - 1$.

При $n > 3$ одно из минимальных реберных 1-расширений графа $K_m + P_n$ имеет вид

$$K_{m+2} + (P_{n_1} \cup P_{n_2}),$$

где $n_1 + n_2 = n - 2$, $1 \leqslant n_1 \leqslant n_2 \leqslant n - 3$. Имеем $m^* = n - 4$.

Подставляя значения m и m^* в формулу (1) при $k = 1$, получаем

$$2n - 2 - 1 + n - 4 - (n - 1) = 2n - 6.$$

Таким образом, число дополнительных ребер составляет $2n - 6$. ■

На рис. 2 представлен граф $K_1 + P_7$ и 4 его минимальных реберных 1-расширения. Полные вершины обозначены черными точками. Для наглядности некоторые ребра, соединяющие полные вершины с остальными, опущены. Количество дополнительных ребер минимальных реберных 1-расширений равно 8.

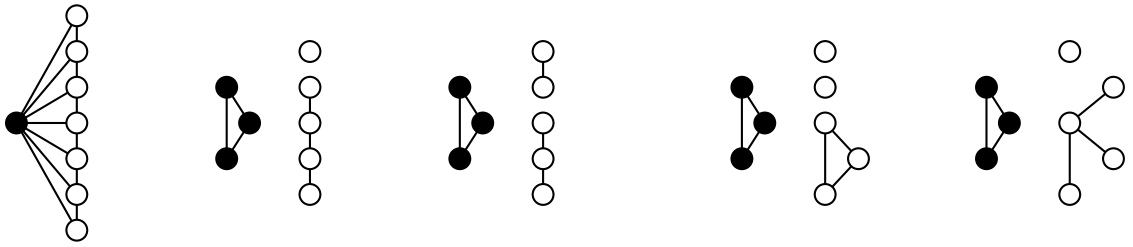

Рис. 2. Граф $K_1 + P_7$ и все его минимальные реберные 1-расширения

Теорему 2 можно обобщить:

Теорема 3. Относительно минимальных реберных k-расширений предполных графов вида $K_m + P_n$ справедливо следующее:

1) при $n < 2k$: минимальных реберных k-расширений нет;

2) при $n = 2k$: минимальное реберное k-расширение единственно с точностью до изоморфизма и имеет вид K_{2k+m};

3) при $2k < n \leqslant 4k + 1$: минимальное реберное k-расширение единственно с точностью до изоморфизма и имеет вид $K_{2k+m} + O_{n-2k}$;

4) при $n > 4k+1$: минимальные реберные k-расширения имеют вид $K_{2k+m} + G_{n-2k}$, где G_{n-2k} — $(n-2k)$-вершинный граф с $n-3k-1$ ребрами, такой, что после удаления любых его k-ребер оставшаяся часть может быть покрыта не более чем $2k+1$ цепями.

Доказательство.

Пункт 1 вытекает из следствия 1 леммы 1.

Пункт 2 получается из следствия 2 леммы 1.

Пусть далее $n > 2k$.

Из следствия 2 леммы 1 следует, что если граф $K_m + P_n$ имеет минимальное реберное k-расширение, то оно имеет вид $K_{2k+m} + G$, где G — $(n-2k)$-вершинный граф. Обозначим для определенности через $v_1, \ldots, v_{2k+m}$ полные вершины из части K_{2k+m} графа $K_{2k+m} + G$. Рассмотрим граф, получающийся после удаления k ребер из графа $K_{2k+m} + G$. В нем останется, по крайней мере, m полных вершин, пусть для определенности это будут вершины $v_{2k+1}, \ldots, v_{2k+m}$. Оставшиеся вершины должны образовывать цепь. Обозначим через K^* подграф, образованный из вершин $v_1, \ldots, v_{2k}$, а через G^* — подграф, образованный из вершин графа G. Для построения цепи имеется $2k$ вершин $v_1, \ldots, v_{2k}$ и вершины подграфа G^*. Граф K_{2k} является $(2k-1)$-реберно связным, поэтому удаление из него любых k ребер его связности не нарушит. Предположим, что подграф G^* можно покрыть не более чем $2k+1$ цепями $P_{n_1}, \ldots, P_{n_l}$. Тогда легко построить n-вершинную цепь, соединяя концы цепей $P_{n_1}, \ldots, P_{n_l}$ вершинами $v_1, \ldots, v_{2k}$. Если же подграф G^* нельзя покрыть не более чем $2k+1$ цепями, то n-вершинную цепь построить невозможно.

Заметим, что минимальным по числу ребер p-вершинным графом, который удовлетворяет приведенному условию, при $p \leqslant 2k+1$ является вполне несвязный граф O_p, а при $p > 2k+1$ — объединение $k+1$ цепей $P_{n_1}, \ldots, P_{n_{k+1}}$. В самом деле, удаление одного ребра разбивает цепь на две цепи, следовательно, удаление k ребер приведет к образованию $2k+1$ цепей. Легко подсчитать и число ребер в p-вершинном графе, являющемся объединением $k+1$ цепей: $p-k-1$. ■

Следствие 6. Число дополнительных ребер в минимальном реберном k-расширении графа $K_m + P_n$ при $2k < n \leqslant 4k + 1$ составляет

$$2nk - 2k^2 - k - n + 1,$$

а при $n > 4k + 1$ —

$$2nk - 2k^2 - 4k.$$

Доказательство. Воспользуемся следствием 3.

Так как граф G_n в данном случае — это цепь P_n, то $m = n - 1$.

При $2k < n \leqslant 4k+1$ минимальное реберное k-расширение графа K_m+P_n имеет вид $K_{2k+m}+O_{n-2k}$, поэтому $m^* = 0$. Подставляя значения m и m^* в формулу (1), получаем

$$2nk - 2k^2 - k - (n - 1) = 2nk - 2k^2 - k - n + 1.$$

При $n > 4k + 1$ минимальные реберные k-расширения графа $K_m + P_n$ имеют вид $K_{2k+m} + G_{n-2k}$, где граф G_{n-2k} имеет $m^* = n - 3k - 1$ ребер.

Подставляя значения m и m^* в формулу (1), получаем

$$2nk - 2k^2 - k + n - 3k - 1 - (n - 1) = 2nk - 4k - 2k^2.$$

■

На рис. 3 представлены минимальные реберные 2-расширения для графа $K_1 + P_{10}$. По-прежнему полные вершины обозначаются черными точками и для наглядности некоторые ребра, соединяющие полные вершины с остальными, опущены. Количество дополнительных ребер минимальных реберных 2-расширений равно 24.

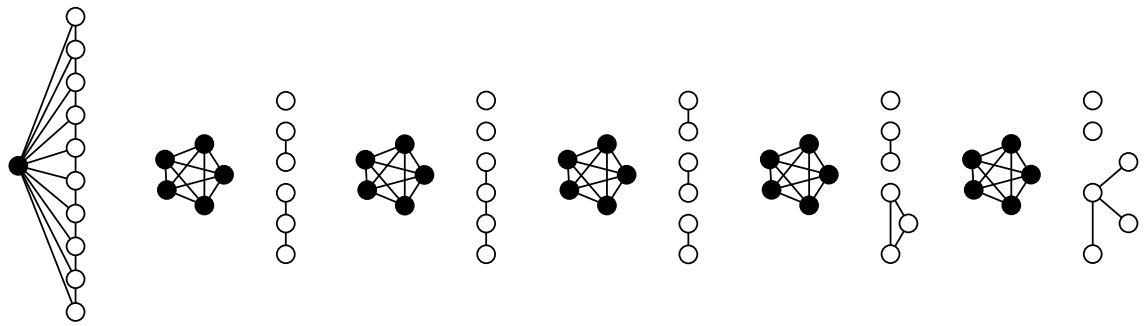

Рис. 3. Граф $K_1 + P_{10}$ и все его минимальные реберные 2-расширения

4. Соединение полного графа и цикла: $K_m + C_n$

Связный n-вершинный граф, все вершины которого имеют степень 2, называется *циклом* и обозначается C_n. Очевидно, что цикл не может иметь менее 3 вершин. Связный граф без циклов называется *деревом*. Вершина дерева, имеющая степень 1, называется *листом*.

Теорема 4. Относительно минимальных реберных 1-расширений графа $K_m + C_n$ справедливы следующие утверждения:

1) при $n = 3$ минимальных реберных 1-расширений нет;

2) при $n = 4$ существует единственное минимальное реберное 1-расширение вида K_{m+4};

3) при $n \geqslant 5$ одно из минимальных реберных 1-расширений имеет вид $K_{m+2} + P_{n-2}$;

4) при $n = 6$ одно из минимальных реберных 1-расширений имеет вид $K_{m+3} + O_3$;

5) при $n \geqslant 6$ одно из минимальных реберных 1-расширений имеет вид $K_{m+2} + (O_1 \cup C_{n-2})$;

6) других минимальных реберных 1-расширений, кроме описанных в п. 2–5, у графа $K_m + C_n$ нет.

Доказательство. При $n = 3$ граф $K_m + C_n$ изоморфен полному графу K_{m+n} и поэтому минимальных реберных 1-расширений не имеет. Далее рассматриваем случай $n > 3$.

По следствию 2 из леммы 1, если граф $K_m + C_n$ имеет минимальное реберное 1-расширение, то его можно представить в виде $K_{m+2} + G_{n-2}$. Обозначим для определенности через $v_1, \ldots, v_{m+2}$ полные вершины в части K_{m+2}, а через $u_1, \ldots, u_{n-2}$ — вершины части G_{n-2}.

Убедимся, что при $n > 3$ граф $K_{m+2} + P_{n-2}$ является реберным 1-расширением графа $K_m + C_n$. Аналогично доказательству теоремы 2 все ребра разделим на три типа: ребра внутри K_{m+2}, ребра внутри P_{n-2} и ребра, соединяющие вершины из K_{m+2} и из P_{n-2}. Заметим, что при удалении ребра любого типа остается, по крайней мере, m полных вершин, поэтому достаточно убедиться, что остальные вершины образуют n-вершинный цикл. Рассмотрим удаление ребра каждого типа.

1) При удалении ребра в части K_{m+2}, для определенности пусть это будет ребро $\{v_1, v_2\}$, остаются полные вершины $v_3, \ldots, v_{m+2}$, а остальные вершины образуют цикл $v_1, u_1, v_2, u_2, \ldots, u_{n-2}, v_1$.

2) При удалении ребра в части P_{n-2} эта цепь распадается на две цепи: P_{n_1} и P_{n_2}. Полные вершины — $v_3, \ldots, v_{m+2}$, а остальные вершины образуют цикл $v_1, P_{n_1}, v_2, P_{n_2}, v_1$.

3) При удалении ребра, соединяющего вершины из K_{m+2} и P_{n-2}, для определенности пусть это будет ребро $\{v_1, u_p\}$, где $1 \leqslant p \leqslant n-2$, остаются полные вершины $v_3, \ldots, v_{m+2}$, а остальные вершины образуют цикл или $v_1, u_1, u_2, \ldots, u_{n-2}, v_2, v_1$ (если $p \neq 1$), или $v_2, u_1, \ldots, u_{n-2}, v_1, v_2$ (если $p = 1$).

Таким образом, при $n > 3$ граф $K_{m+2} + P_{n-2}$ действительно является реберным 1-расширением графа $K_m + C_n$. При $n = 4$ граф $K_{m+2} + P_2$ изоморфен графу K_{m+4}.

Пусть $K_{m+2} + G_{n-2}$ является минимальным реберным 1-расширением графа $K_m + C_n$. Тогда граф G_{n-2} содержит не более $n-3$ ребер. Рассмотрим удаление произвольного ребра e в G_{n-2}. По предположению вершины графа G_{n-2} вместе с двумя полными вершинами образуют цикл, следовательно, граф $(G_{n-2} - e)$ содержит не более двух компонент связности и может быть покрыт двумя цепями. Первое условие означает, что число ребер в G_{n-2} не менее $n-3$. Поскольку цепь P_{n-2} содержит в точности $n-3$ ребра, то граф $K_{m+2} + P_{n-2}$ является минимальным реберным 1-расширением графа $K_m + C_n$, а любое другое минимальное реберное 1-расширение, если оно есть, имеет вид $K_{m+2} + G_{n-2}$, где граф G_{n-2} имеет $n-2$ вершин и $n-3$ ребер. Так как граф $(G_{n-2} - e)$ содержит не более двух компонент связности, то граф G_{n-2} либо состоит из одной компоненты связности и тогда является деревом, либо из одной $(n-3)$-вершинной двусвязной компоненты с $n-3$ ребрами, то есть цикла, и одной изолированной вершины.

Исследуем, каким может быть дерево в первом случае. Покажем, что оно не может иметь более трех листьев. Предположим, что это не так и граф G_{n-2} является деревом с более чем тремя листьями. Рассмотрим граф, получающийся удалением из G_{n-2} ребра при некотором листе v. Получим дерево с числом вершин на одну меньше и

не менее чем с тремя листьями и изолированную вершину: $T_{n-3} \cup O_1$. По условию граф $K_2 + (T_{n-3} \cup O_1)$ должен содержать n-вершинный цикл. Вершина v в этом цикле должна соседствовать с обеими полными вершинами v_1 и v_2. Другим соседом полной вершины v_1 будет один из листьев дерева. Далее последует цепь (единственная) до другого листа, затем вершина v_2. Оставшиеся листья не могут быть присоединены к циклу. Таким образом, дерево не может иметь более трех листьев. Один лист может иметь только тривиальное дерево (состоящее из одной вершины). Если листа два, то получаем рассмотренный ранее случай цепи.

Исследуем случай трех листьев. В этом случае дерево, очевидно, имеет одну вершину степени 3, три вершины степени 1 и может иметь несколько вершин степени 2. Однако, если хотя бы один лист не смежен с вершиной степени 3, то, удалив ребро при таком листе, мы опять получим граф вида $T_{n-3} \cup O_1$. Повторяя предыдущие рассуждения, получим противоречие. Итак, деревом с тремя листьями может быть лишь 4-вершинная звезда $K_1 + O_3$.

Таким образом, граф G_{n-2} может иметь вид $O_1 \cup C_{n-3}$, P_{n-2} или $K_1 + O_3$. ■

Следствие 7. Число дополнительных ребер в минимальном реберном 1-расширении графа $K_m + C_n$ при $n \geqslant 5$ составляет $2n - 6$.

Доказательство. Воспользуемся следствием 3.

Так как граф G_n в данном случае — это цикл C_n, то $m = n$.

При $n \geqslant 5$ минимальное реберное 1-расширение графа $K_m + C_n$ имеет вид $K_{m+2} + P_{n-2}$, поэтому $m^* = n - 3$. Подставляя значения m и m^* в формулу (1) при $k = 1$, получаем $2n - 2 - 1 + n - 3 - n = 2n - 6$. ■

Таким образом, графы $K_1 + C_4$ и $K_1 + C_5$ имеют единственное минимальное реберное 1-расширение, граф $K_1 + C_6$ имеет три минимальных реберных 1-расширения, а при $n > 6$ граф $K_1 + C_n$ имеет два минимальных реберных 1-расширения. На рис. 4 представлены все минимальные реберные 1-расширения для графа $K_1 + C_6$. Как и ранее, полные вершины обозначаются черными точками и для наглядности некоторые ребра, соединяющие полные вершины с остальными, опущены. Количество дополнительных ребер минимальных реберных 1-расширений равно 6.

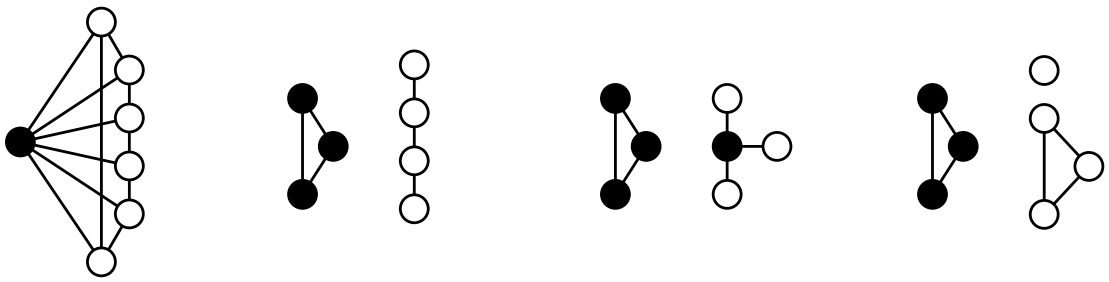

Рис. 4. Граф $K_1 + C_6$ и все его минимальные реберные 1-расширения

Теорему 4 можно обобщить:

Теорема 5. Относительно минимальных реберных k-расширений предполных графов вида $K_m + C_n$ при $k > 1$ справедливо следующее:

1) при $n < 2k$ и $n = 4$: минимальных реберных k-расширений нет;

2) при $n = 2k$: минимальное реберное k-расширение единственно: K_{2k+m};

3) при $2k < n \leqslant 4k$: минимальное реберное k-расширение единственно: $K_{2k+m} + O_{n-2k}$;

4) при $n > 4k$: минимальные реберные k-расширения имеют вид $K_{2k+m} + G_{n-2k}$, где G_{n-2k} — это $(n-2k)$-вершинный граф с $n-3k$ ребрами, такой, что после удаления любых его k ребер оставшаяся часть может быть покрыта не более чем $2k$ цепями.

Доказательство. Пункты 1 и 2 вытекают из следствий 1 и 2 леммы 1.

Рассмотрим $n > 2k$.

Рассуждаем аналогично доказательству теоремы 3. Из следствия 2 леммы 1 следует, что если граф $K_1 + C_n$ имеет минимальное реберное k-расширение, то оно имеет вид $K_{2k+m} + G$, где G — это $(n-2k)$-вершинный граф. Обозначим для определенности через $v_1, \ldots, v_{2k+m}$ полные вершины из части K_{2k+m} графа $K_{2k+m} + G$. Рассмотрим граф, получающийся после удаления k ребер из графа $K_{2k+m} + G$. В нем останется, по крайней мере, m полных вершин, пусть для определенности это будут вершины $v_{2k+1}, \ldots, v_{2k+m}$. Оставшиеся вершины должны образовывать цикл. Обозначим через K^* подграф, образованный из вершин $v_1, \ldots, v_{2k}$, а через G^* — подграф, образованный из вершин графа G. Для построения цикла имеется $2k$ вершин $v_1, \ldots, v_{2k}$ и вершины подграфа G^*. Граф K_{2k} является $(2k-1)$-реберно связным, поэтому удаление из него любых k ребер его связности не нарушит. Предположим, что подграф G^* можно покрыть не более чем $2k$ цепями $P_{n_1}, \ldots, P_{n_l}$. Тогда легко построить n-вершинный цикл, соединяя концы цепей $P_{n_1}, \ldots, P_{n_l}$ с вершинами $v_1, \ldots, v_{2k}$. Если же подграф G^* нельзя покрыть не более чем $2k$ цепями, то n-вершинний цикл построить невозможно.

Заметим, что минимальным по числу ребер p-вершинным графом, который удовлетворяет приведенному условию, при $p \leqslant 2k$ является вполне несвязный граф O_p, а при $p > 2k$ — объединение k цепей $P_{n_1}, \ldots, P_{n_k}$. В самом деле, удаление одного ребра разбивает цепь на две цепи, следовательно, удаление k ребер приведет к образованию $2k$ цепей. Легко подсчитать и число ребер в p-вершинном графе, являющемся объединением k цепей: $p-k$. Вспоминая, что $p = n-2k$, завершаем доказательство. ■

Следствие 8. Число дополнительных ребер в минимальном реберном k-расширении графа $K_m + C_n$ при $2k < n \leqslant 4k$ составляет

$$2nk - 2k^2 - k - n,$$

а при $n > 4k$ —

$$2nk - 2k^2 - 4k.$$

Доказательство. Воспользуемся следствием 3.

Так как граф G_n в данном случае — это цикл C_n, то $m = n$.

При $2k < n \leqslant 4k$ минимальное реберное k-расширение графа $K_m + C_n$ имеет вид $K_{2k+m} + O_{n-2k}$, поэтому $m^* = 0$. Подставляя значения m и m^* в формулу (1), получаем

$$2nk - 2k^2 - k - n.$$

При $n > 4k$ минимальные реберные k-расширения графа $K_m + P_n$ имеют вид $K_{2k+m} + G_{n-2k}$, где граф G_{n-2k} имеет $m^* = n - 3k$ ребер.

Подставляя значения m и m^* в формулу (1), получаем

$$2nk - 2k^2 - k + n - 3k - 1 - n = 2nk - 4k - 2k^2 - 1.$$

■

ЛИТЕРАТУРА

1. *Богомолов А. М., Салий В. Н.* Алгебраические основы теории дискретных систем. М.: Наука, 1997.
2. *Hayes J. P.* A graph model for fault-tolerant computing system // IEEE Trans. Comput. 1976. V. C25. No. 9. P. 875–884.
3. *Harary F., Hayes J. P.* Edge fault tolerance in graphs // Networks. 1993. V. 23. P. 135–142.
4. *Harary F., Hayes J. P.* Node fault tolerance in graphs // Networks. 1996. V. 27. P. 19–23.
5. *Абросимов М. Б.* О вычислительной сложности расширений графов // Прикладная дискретная математика. Приложение. 2009. № 1. С. 94–95.
6. *Абросимов М.Б.* Минимальные k-расширения предполных графов // Изв. вузов. Математика. 2003. № 6(493). С. 3–11.

DOI 10.17223/20710410/7/9
УДК 519.6

СВОЙСТВА ПУТЕЙ В ГРАФАХ И МУЛЬТИГРАФАХ

В. М. Фомичев

Институт проблем информатики РАН, г. Москва, Россия

E-mail: fomichev@nm.ru

Для n-вершинного сильносвязного орграфа оценена длина кратчайшего полного пути и, при наличии петли в графе, экспонент матрицы смежности вершин. Получена полиномиальная оценка субэкспонента системы матриц смежности вершин n-вершинных графов $\Gamma_1, \ldots, \Gamma_p$, объединение которых сильно связно. Полученные результаты могут использоваться для исследования существенных переменных координатных функций, определяющих композиции преобразований множества конечных слов.

Ключевые слова: *полный путь, кратчайший путь, экспонент, субэкспонент.*

Введение

Решение ряда прикладных задач методами теории графов часто увязывается с исследованием свойств множества путей в определенных графах. К таким задачам в криптологии относится, например, исследование существенной зависимости функций от переменных [1, гл. 10]. В криптографических системах аутентификации изучение множеств существенных переменных функций важно для построения преобразований, распространяющих искажения [1, 2]. В криптографических системах шифрования эти задачи решаются для оценки эффективности алгебраических атак, основанных на последовательном опробовании элементов ключа [3, с. 397].

Важным понятием в теории графов является расстояние (длина кратчайшего пути) между двумя вершинами графа, которое может определяться двояко для пары одинаковых вершин. Если в транспортных задачах естественно рассматривать графы без петель и предполагать, что расстояние между двумя одинаковыми вершинами равно 0, то в некоторых других задачах, например при исследовании существенной зависимости от переменных композиций функций, допускается наличие петель в графе и полагается, что расстояние от x до x равно длине кратчайшего цикла, проходящего через вершину x. В соответствии с этим следует различать и другие производные понятия теории графов: «эксцентриситет вершины», «радиус графа», «диаметр графа» и др. Часто используются определения первого типа, связанные с расстояниями в графе (см., например, [4, с. 262]); в данной же работе — определения второго типа, которые в большей мере отвечают прикладным задачам криптологии.

Далее в соответствии с [5, 6 и др.] положим, что множество ребер графа, так же как и множество дуг орграфа, может содержать петли, а мультимножество ребер неориентированного (дуг ориентированного) мультиграфа содержит параллельные ребра (дуги), в частности, может содержать параллельные петли. В помеченном мультиграфе параллельные ребра (дуги) различаются, если им присвоены разные метки. Заметим — объединение нескольких графов (орграфов) с одинаковым множеством вершин может быть как графом (орграфом), так и мультиграфом.

Некоторые теоретико-графовые задачи состоят в определении путей с заданными характеристиками (с фиксированными начальной и конечной вершинами, с фиксиро-

ванной длиной) в различных графах и мультиграфах. Напомним [7, с. 143, следствие 1 теоремы 2], что в n-вершинном графе Γ число путей длины ℓ из вершины i в вершину j определяется элементом $m_{ij}^{(\ell)}$ матрицы M^ℓ, где M — матрица смежности вершин графа Γ, $i, j \in \{1, ..., n\}, \ell \geqslant 1$. Вместе с тем эта теорема может использоваться не во всех случаях. Например:

1) указанная теорема требует уточнения для помеченных графов (мультиграфов), чтобы различать пути длины ℓ из i в j с разными метками;
2) при больших n, ℓ высока вычислительная сложность подсчета матриц $M^\ell, \ell \geqslant 1$.

Данная работа посвящена получению оценок длин путей с определенными свойствами в n-вершинных ориентированных и неориентированных графах и мультиграфах. Без ущерба для общности рассмотрим лишь связные (не обязательно сильносвязные) ориентированные графы и мультиграфы.

1. Свойства систем неотрицательных матриц

Матрица $M = (m_{ij})$ над полем действительных чисел положительна (неотрицательна), если положительны (неотрицательны) все ее элементы. Это свойство записывается так: $M > 0$ ($M \geqslant 0$). Заметим, что множество квадратных неотрицательных матриц размера n содержит множество матриц смежности вершин n-вершинных графов.

При последовательном возведении в степени $1, 2, \ldots, \ell, \ldots$ квадратная неотрицательная матрица M может на некотором шаге стать положительной. В этом случае матрица M называется примитивной [8], а наименьшее натуральное ℓ, при котором $M^\ell > 0$, называется экспонентом матрицы M (обозначается $\exp M$). Если такого ℓ не существует, то $\exp M = \infty$.

Субэкспонентом матрицы M (обозначается $\mathrm{sbxp}\, M$) называется наименьшее натуральное число ℓ, при котором $M_\Sigma^\ell > 0$, где

$$M_\Sigma^\ell = M + M^2 + \ldots + M^\ell.$$

Если M — матрица смежности вершин некоторого графа Γ, то в силу следствия 1 теоремы 2 [7, с. 143] $\mathrm{sbxp}\, M$ совпадает с диаметром графа Γ.

Понятия экспонента и субэкспонента обобщены на систему квадратных неотрицательных матриц $\mathfrak{M} = \{M_1, \ldots, M_p\}$ одинакового размера [1, гл. 10]. Пусть $N_p = \{1, \ldots, p\}$, N_p^* — множество всех слов в алфавите N_p. Слову $w = s_1 \ldots s_\ell$ из N_p^* при заданной системе матриц M однозначно соответствует матрица $M_{s_1} \cdot \ldots \cdot M_{s_\ell}$, являющаяся элементом мультипликативной полугруппы $\langle \mathfrak{M} \rangle$ неотрицательных матриц, порожденной системой $\mathfrak{M}$. Обозначим $M_{s_1} \cdot \ldots \cdot M_{s_\ell} = M(w) = (m_{ij}(w))$.

Экспонентом системы матриц $\mathfrak{M}$ (обозначается $\exp \mathfrak{M}$) называется наименьшая длина ℓ слова $w \in N_p^*$, при котором $M(w) > 0$. Если такого слова не существует, то полагаем $\exp \mathfrak{M} = \infty$.

Субэкспонентом системы матриц $\mathfrak{M}$ (обозначается $\mathrm{sbxp}\, \mathfrak{M}$) называется наименьшая длина ℓ слова $w = s_1 \ldots s_\ell$ из N_p^*, при котором $M_\Sigma(w) > 0$, где

$$M_\Sigma(w) = M_{s_1} + M_{s_1} \cdot M_{s_2} + \ldots + M(w).$$

Если такое слово не существует, то полагаем $\mathrm{sbxp}\, \mathfrak{M} = \infty$.

Утверждение 1. Для любой системы $\mathfrak{M}$ квадратных неотрицательных матриц одинакового размера $\mathrm{sbxp}\, \mathfrak{M} \leqslant \exp \mathfrak{M}$.

Заметим, что имеются системы матриц с конечным субэкспонентом и бесконечным экспонентом. Такова, например, любая система подстановочных матриц порядка n, порождающая группу, изоморфную транзитивной группе подстановок степени n. В частности, подстановочная матрица, соответствующая полноцикловой подстановке, имеет конечный субэкспонент и бесконечный экспонент.

На множестве матриц (над полем R) заданного размера имеется частичный порядок: $M \geqslant M'$ тогда и только тогда, когда $m_{ij} \geqslant m'_{ij}$ для всех допустимых i, j, где $M' = (m'_{ij})$. Указанный частичный порядок индуцирует квазипорядок на множестве систем матриц (над полем R) заданного размера: $\mathfrak{M} \geqslant \mathfrak{M}'$ тогда и только тогда, когда для любой матрицы $M' \in \mathfrak{M}'$ имеется матрица $M \in \mathfrak{M}$, такая, что $M \geqslant M'$. Непосредственно из определений следуют «монотонные» свойства.

Утверждение 2. Если $\mathfrak{M} \geqslant \mathfrak{M}'$ или $\mathfrak{M} \supseteq \mathfrak{M}'$, где $\mathfrak{M}$, $\mathfrak{M}'$ — системы квадратных неотрицательных матриц одинакового размера, то $\operatorname{sbxp}\mathfrak{M} \leqslant \operatorname{sbxp}\mathfrak{M}'$ и $\exp\mathfrak{M} \leqslant \exp\mathfrak{M}'$.

С точки зрения теории признаков в полугруппах подмножество квадратных положительных матриц размера n можно рассматривать как полугрупповой признак в полугруппе квадратных неотрицательных матриц размера n [1, разд. 9.1], показатель этого признака в системе образующих $\mathfrak{M}$ совпадает с $\exp\mathfrak{M}$.

2. Число путей с заданной меткой в объединении орграфов

В графе Γ с множеством вершин $\{1, \ldots, n\}$ обозначим через $d_{i,j}$ длину кратчайшего пути из вершины i в вершину j (расстояние от i до j), тогда диаметр графа Γ есть

$$\operatorname{diam}\Gamma = \max\{d_{i,j} : i, j = 1, \ldots, n\}.$$

Пусть Γ_s — граф с множеством вершин $\{1, \ldots, n\}$, где все дуги помечены числом s, и $M_s = (m_{ij}(s))$ — матрица смежности вершин графа Γ_s, $s = 1, \ldots, p$. Тогда объединение графов $\Gamma^{(p)} = \Gamma_1 \cup \ldots \cup \Gamma_p$ в зависимости от объединяемых множеств дуг есть либо граф, либо мультиграф, которому соответствует система матриц смежности $\mathfrak{M} = \{M_1, \ldots, M_p\}$. Любой путь длины ℓ в мультиграфе (графе) $\Gamma^{(p)}$ помечен словом из N_p^* длины ℓ. Далее рассмотрим мультиграф $\Gamma^{(p)}$, рассуждения для графа проводятся аналогично.

Для данных мультиграфов верно обобщение следствия 1 теоремы 2 [7, с. 143].

Теорема 1. В мультиграфе $\Gamma^{(p)}$ число путей длины ℓ из вершины i в вершину j с меткой $w = (s_1, \ldots, s_\ell)$ равно $m_{ij}(w)$.

Доказательство. Индукция по ℓ.

Для $\ell = 1$ теорема следует из определения матриц M_s, $s = 1, \ldots, p$, и определения мультиграфа $\Gamma^{(p)}$.

Пусть теорема доказана для $\ell - 1$, где $\ell > 1$, для любой пары вершин (i, j) мультиграфа $\Gamma^{(p)}$ и для любого слова $u = (s_1, \ldots, s_{\ell-1}) \in N_p^*$.

Докажем теорему для ℓ, для любой пары (i, j) вершин мультиграфа $\Gamma^{(p)}$ и для любого слова $w \in N_p^*$.

По определению $M(w) = M(u) \cdot M_{s_\ell}$, отсюда получаем по правилу умножения матриц

$$m_{ij}(w) = \sum_{r=1}^{n} m_{ir}(u) m_{rj}(s_\ell). \tag{1}$$

По предположению индукции $m_{ir}(u)$ есть число путей длины $\ell - 1$ с меткой u из i в r в мультиграфе $\Gamma^{(p)}$. Тогда $m_{ir}(u)m_{rj}(s_\ell)$ есть число путей длины ℓ с меткой w из i

в j в $\Gamma^{(p)}$, таких, что вершина r предшествует вершине j. Суммируя по r, получаем общее число путей длины ℓ с меткой w из i в j в $\Gamma^{(p)}$. ■

Следствие 1. $\operatorname{diam}\Gamma^{(p)} \leqslant \operatorname{sbxp}\mathfrak{M}$, в частности, если $\Gamma_1 = \ldots = \Gamma_p = \Gamma$, то $\operatorname{diam}\Gamma = \operatorname{sbxp} M$.

Доказательство.

В соответствии с теоремой 1 $M_\Sigma(w) > 0$ тогда и только тогда, когда для любой пары вершин (i, j) в $\Gamma^{(p)}$ имеется путь из i в j, у которого метка совпадает с началом слова w. Значит, если $M_\Sigma(w) > 0$, то $\operatorname{diam}\Gamma^{(p)}$ не превышает длину слова w, так как по определению диаметр мультиграфа есть максимум по всем парам (i, j) длин кратчайших путей из i в j без ограничения на метки. Отсюда $\operatorname{diam}\Gamma^{(p)} \leqslant \operatorname{sbxp}\mathfrak{M}$.

При условии $\Gamma_1 = \ldots = \Gamma_p = \Gamma$ верно и обратное неравенство в силу отсутствия меток в графе Γ, то есть в этом случае $\operatorname{diam}\Gamma^{(p)} = \operatorname{sbxp}\mathfrak{M}$. ■

3. О длинах путей между заданными вершинами

В некоторых приложениях возникает задача определения в графе Γ всех длин путей из вершины i в вершину j. В частности, существует ли натуральное число λ_{ij}, такое, что для любого $\ell \geqslant \lambda_{ij}$ в графе Γ имеется путь длины ℓ из вершины i в вершину j?

В графе Γ всякий путь, проходящий через вершину, в которой имеется петля, назовем путем с петлей. Ниже используем следующую лемму.

Лемма 1. Если в графе Γ имеется путь из i в j длины λ с петлей, то для любого $\ell \geqslant \lambda$ имеется путь из i в j длины ℓ.

Обозначим в графе Γ, где i, j, r — вершины графа:
$d_{i,r,j}$ — длину кратчайшего пути из i в j, проходящего через r, где $d_{i,r,j} = d_{i,j}$ при $r \in \{i, j\}$;
$\operatorname{diam}^r\Gamma = \max\{d_{i,r,j} : i, j = 1, \ldots, n\}$ — назовем эту величину r-диаметром графа;
$e(i) = \max\{d_{i,j} : j = 1, \ldots, n\}$ — эту величину называют эксцентриситетом вершины i;
$p(i) = \max\{d_{j,i} : j = 1, \ldots, n\}$ — назовем эту величину периферийностью вершины i.

В случае неориентированного графа $e(i) = p(i)$, $i = 1, \ldots, n$.

Лемма 2. При $n \geqslant 2$ в n-вершинном сильносвязном орграфе Γ:

а) если $r \notin \{i, j\}$, то $\operatorname{diam}\Gamma \leqslant \operatorname{diam}^r\Gamma \leqslant p(r) + e(r) \leqslant 2n - 2$;

б) если $r \in \{i, j\}$, то $\operatorname{diam}\Gamma = \operatorname{diam}^r\Gamma \leqslant n$.

Доказательство. Если $r \notin \{i, j\}$, то по определению

$$\operatorname{diam}^r\Gamma = \max\{d_{i,r} + d_{r,j} : i, j = 1, \ldots, n\},$$

где $d_{i,r} \leqslant p(r) \leqslant n - 1$, $d_{r,j} \leqslant e(r) \leqslant n - 1$. Значит, $\operatorname{diam}^r\Gamma \leqslant p(r) + e(r) \leqslant 2n - 2$ при $n \geqslant 2$.

Если $r \in \{i, j\}$, то $\operatorname{diam}^r\Gamma = \max\{d_{i,j} : i, j = 1, \ldots, n\} \leqslant n$. ■

Замечание 1. Оценка $\operatorname{diam}^r\Gamma \leqslant 2n - 2$ достижима при любом $n \geqslant 2$. В частности, она достигается для n-вершинного графа с множеством дуг $\{(i, i+1), (i+1, i) : i = 1, \ldots, n-1\}$; в этом случае $\operatorname{diam}^n\Gamma = d_{1,n,1} = 2n - 2$.

Теорема 2. Если n-вершинный граф Γ сильно связен и имеет петлю в вершине r, то для любых $i, j \in \{1, \ldots, n\}$ и любого $\ell \geqslant \operatorname{diam}^r\Gamma$ имеется путь из i в j длины ℓ.

Доказательство. Так как граф Γ сильно связен, то для любых $i, j \in \{1, \ldots, n\}$ имеется путь с петлей из i в j, проходящий через r; длина $d_{i,r,j}$ кратчайшего такого

пути не превышает $\operatorname{diam}{}^r\Gamma$. Вместе с тем в соответствии с определением величины $\operatorname{diam}{}^r\Gamma$ в Γ найдется пара вершин (u, v), такая, что $d_{u,r,v} = \operatorname{diam}{}^r\Gamma$. Отсюда по лемме 1 получаем требуемое утверждение. ■

Следствие 2. При $n \geqslant 2$ для матрицы M смежности вершин графа Γ верно: $\exp M \leqslant \operatorname{diam}{}^r\Gamma \leqslant 2n-2$.

Доказательство. Из теоремы 2 получаем в соответствии со следствием 1 теоремы 2 [7, с. 143], что $M^\ell > 0$ при любом $\ell \geqslant \operatorname{diam}{}^r\Gamma$, где по лемме 2 $\operatorname{diam}{}^r\Gamma \leqslant 2n-2$ при $n \geqslant 2$. ■

Тем самым абсолютная оценка $\exp M \leqslant n^2-2n+2$ для n-вершинных сильносвязных графов [9] уточнена для графов с петлей.

4. О длине кратчайшего полного пути в сильносвязном орграфе

Следствие 1 позволяет поставить вопрос: существует ли мультиграф $\Gamma^{(p)}$ с конечным диаметром и бесконечным $\operatorname{sbxp}\mathfrak{M}$? Здесь получен отрицательный ответ и выведена верхняя оценка для $\operatorname{sbxp}\mathfrak{M}$.

Заметим, если мультиграф $\Gamma^{(p)}$ не сильносвязный, то $\operatorname{sbxp}\mathfrak{M} = \infty$ в соответствии с теоремой 1 и $\operatorname{diam}\Gamma^{(p)} = \infty$ в соответствии с определением диаметра графа. Следовательно, поставленный вопрос относится исключительно к сильносвязным мультиграфам $\Gamma^{(p)}$.

В любом сильносвязном орграфе (мультиграфе) имеется полный, то есть проходящий через все вершины цикл (путь). Оценим длину кратчайшего из полных циклов (путей).

Теорема 3. В n-вершинном сильносвязном орграфе Γ имеются:

а) при $n \geqslant 4$ полный цикл длины не более $\lambda(n) = (n^2-n)/2$;
б) при $n \geqslant 4$ полные пути длины не более $\lambda(n)-1$ с началом в любой вершине;
в) при $n \geqslant 2$ полный путь длины не более $\lambda'(n) = \lambda(n)-n+2$ с началом в некоторой вершине.

Доказательство. Длину пути z в Γ, равную числу дуг пути, обозначим $\operatorname{len}(z)$. Используя индукцию по k, докажем вспомогательное утверждение: в Γ имеется путь длины не более $\lambda'(k)$, проходящий через k различных вершин, $k = 2, 3, \ldots, n$. При $k = n$ вспомогательное утверждение равносильно утверждению «в» теоремы.

По условию орграф Γ сильносвязный, значит, в Γ имеется путь длины 1, проходящий через 2 разные вершины, с началом в любой вершине. Так как $\lambda'(2) = 1$, то утверждение верно при $k = 2$.

Пусть утверждение верно при $k < n$, где $n > 2$, докажем его при $k+1$. Обозначим через $z(k)$ путь длины не более $\lambda'(k)$, проходящий через k различных вершин. Без ущерба для общности положим, что $z(k)$ есть путь из 1 в k, проходящий через вершины $1, \ldots, k$. Так как Γ сильносвязный, то в нем имеется путь z из k в некоторую вершину, отличную от $1, \ldots, k$. Длина $\operatorname{len}(z)$ кратчайшего такого пути не превышает k, так как путь z является простым и все его вершины, кроме первой и последней, образуют бесповторную выборку размера не более $k-1$ из множества $\{1, \ldots, k-1\}$.

Если $\operatorname{len}(z) \leqslant k-1$, то искомый путь $z(k+1)$ определим как соединение путей $z(k)$ и z. Действительно, $z(k+1)$ есть путь, проходящий через $k+1$ различных вершин, и его длина $\operatorname{len}(z(k+1))$ удовлетворяет оценкам

$$\operatorname{len}(z(k+1)) \leqslant \lambda'(k) + k - 1 = \lambda'(k+1).$$

Если $\mathrm{len}(z) = k$, то искомый путь $z(k+1)$ определим как z. Следовательно, при $k \geqslant 2$ в обоих случаях $\mathrm{len}(z(k+1)) \leqslant \lambda'(k+1)$. Утверждение «в» доказано.

Без ущерба для общности положим, что построенный полный путь $z(n)$ есть путь из вершины 1 в вершину n. Соединив путь $z(n)$ с кратчайшим путем z' из вершины n в вершину 1 (длина его $\mathrm{len}(z')$ не превышает $n-1$), получим полный цикл в Γ. Если $\mathrm{len}(z') \leqslant n-2$, то длина полного цикла не превышает $\lambda(n)$. Если $\mathrm{len}(z') = n-1$, то путь z' является полным. Соединив путь z' с кратчайшим путем z'' из вершины 1 в вершину n (длина $\mathrm{len}(z'')$ не превышает $n-1$), получим также полный цикл в Γ длины не больше $2n-2$. Следовательно, при $n \geqslant 4$ в обоих случаях длина полного цикла не превышает $\lambda(n)$. Утверждение «а» доказано.

Если из построенного полного цикла удалить любую дугу, то получим полный путь (начало можно выбрать произвольно) длины не более $\lambda(n)-1$, где $n \geqslant 4$. Утверждение «б» также доказано. ■

Замечание 2. Оценки теоремы 2 совпадают по порядку с точными оценками, что подтверждается примером $2n$-вершинного орграфа Γ с множеством дуг E:

$$E = \{(2n, i), (i, n+1) : i = 1, \ldots, n\} \cup \{(j, j+1) : j = n+1, n+2, \ldots, 2n-1\}.$$

В графе Γ длина полного цикла равна $n(n+1)$, длина полного пути из 1 в n равна n^2-1.

Замечание 3. Теорема 3 верна и для любого n-вершинного сильносвязного мультиграфа.

Теорема 4. Если n-вершинный мультиграф $\Gamma^{(p)}$ сильносвязный, то при $n \geqslant 4$

$$\mathrm{sbxp}\,\mathfrak{M} \leqslant \frac{(n^2-2)(n-1)}{2}.$$

Доказательство. В соответствии с определением $\mathrm{sbxp}\,\mathfrak{M}$ достаточно построить слово $w = s_1 \ldots s_\ell \in N_p^*$, $\ell \leqslant \dfrac{(n^2-2)(n-1)}{2}$, при котором $M_\Sigma(w) > 0$. Опишем n шагов построения, где на i-м шаге строится начальный отрезок слова w, при котором положительны все элементы первых i строк матрицы $M_\Sigma(w), i = 1, \ldots, n$.

Для $w = s_1 \ldots s_\ell$ положим: $w(\tau) = s_1 \ldots s_\tau$, где $\tau = 1, \ldots, \ell$.

1-й шаг. В сильносвязном мультиграфе $\Gamma^{(p)}$ имеется полный цикл C_1 длины $\ell(1)$, где $\ell(1) \leqslant \lambda(n)$ по утверждению «а» теоремы 3. Пусть 1 — начальная вершина цикла C_1 и $w(\ell(1))$ — метка цикла C_1 (слово длины $\ell(1)$ в алфавите N_p^*). Тогда в $\Gamma^{(p)}$ имеется путь из 1 в j длины $\tau(j)$, где $1 \leqslant \tau(j) \leqslant \ell(1)$, отсюда по теореме 1 имеем $m_{1j}(w(\tau(j))) > 0$, $j = 1, \ldots, n$. Следовательно, все элементы первой строки матрицы $\sum_{j=1}^{n} M(w(\tau(j)))$ положительны. Отсюда положительны все элементы первой строки матрицы $M_\Sigma(w(\ell(1)))$, так как

$$M_\Sigma(w(\ell(1))) = \sum_{i=1}^{\ell(1)} M(w(i)) \geqslant \sum_{j=1}^{n} M(w(\tau(j))).$$

Пусть выполнены $i-1$ шагов построения, то есть построено слово $w(\ell(i-1))$, такое, что положительны все элементы первых $i-1$ строк матрицы $M_\Sigma(w(\ell(i-1))), 1 < i \leqslant n$.

***i*-й шаг.** Обозначим через C'_{i-1} путь длины $\ell(i-1)$ с меткой $w(\ell(i-1))$, начинающийся в i. Пусть конечная вершина пути C'_{i-1} есть $\mu \in \{1, \ldots, n\}$. Продолжим слово $w(\ell(i-1))$ до слова $w(\ell(i))$. В $\Gamma^{(p)}$ имеется полный путь C_i длины ℓ_i с началом в μ, где

$\ell_i \leqslant \lambda(n) - 1$ по утверждению «б» теоремы 3; метку пути C_i обозначим w_i. Построим начинающийся в i путь C'_i длины $\ell(i)$ с помощью последовательного соединения путей C'_{i-1} и C_i, где $\ell(i) = \ell(i-1) + \ell_i$ и метка $w(\ell(i))$ пути C'_i получена соединением меток путей C'_{i-1} и C_i: $w(\ell(i)) = w(\ell(i-1))w_i$. Тогда по построению в $\Gamma^{(p)}$ имеется путь из i в j длины $\theta(j)$, где $\ell(i-1) \leqslant \theta(j) \leqslant \ell(i)$, отсюда по теореме 1 имеем $m_{ij}(w(\theta(j))) > 0$, $j = 1, \ldots, n$. Значит, все элементы i-й строки матрицы $\sum_{j=1}^{n} M(w(\theta(j)))$ положительны. Отсюда положительны все элементы i-й строки матрицы $M_\Sigma(w(\ell(i)))$, так как

$$M_\Sigma(w(\ell(i))) = \sum_{s=1}^{\ell(i)} M(w(s)) \geqslant \sum_{j=1}^{n} M(w(\theta(j))).$$

С учетом предположения индукции все элементы первых $i-1$ строк матрицы $M_\Sigma(w(\ell(i)))$ также положительны, так как $M_\Sigma(w(\ell(i))) \geqslant M_\Sigma(w(\ell(i-1)))$. Следовательно, положительны все элементы первых i строк матрицы $M_\Sigma(w(\ell(i)))$, $i = 1, \ldots, n$. Значит, искомое слово w совпадает с $w(\ell(n))$.

Так как $\ell(i) = \ell(i-1) + \ell_i$, где $\ell(i) \leqslant \lambda(n)$ и $\ell_i \leqslant \lambda(n) - 1, i \geqslant 1$, то верна оценка

$$\ell(n) = \ell(1) + \ell_2 + \ldots + \ell_n \leqslant n\lambda(n) - n + 1 = \frac{(n^2 - n)(n-1)}{2}.$$

■

Выводы

Для n-вершинного сильносвязного орграфа (мультиграфа) Γ получены оценки:

1) при $n \geqslant 4$ длина кратчайшего полного пути в Γ не превышает $n(n-1)/2$, эта оценка является точной по порядку;
2) при наличии петли в Γ экспонент матрицы смежности вершин не превышает $2n - 2$, $n \geqslant 2$.

При $n \geqslant 4$ субэкспонент системы матриц смежности вершин n-вершинных графов $\Gamma_1, \ldots, \Gamma_p$, объединение которых сильно связно, не превышает $(n^2 - 2)(n-1)/2$.

ЛИТЕРАТУРА

1. *Фомичёв В. М.* Методы дискретной математики в криптологии. М.: ДИАЛОГ-МИФИ, 2010. 424 с.
2. *Шнайер Б.* Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: ТРИУМФ, 2002. 816 с.
3. *Шеннон К.* Работы по теории информации и кибернетике. М.: ИЛ, 1963.
4. *Харари Ф., Палмер Э.* Перечисление графов. М.: Мир, 1977.
5. *Кофман А.* Введение в прикладную комбинаторику. М.: Наука, 1975. 480 с.
6. `http://dic.academic.ru` — Словарь терминов теории графов.
7. *Берж К.* Теория графов и её применение. М.: ИЛ, 1962. 320 с.
8. *Сачков В. Н., Ошкин И. Б.* Экспоненты классов неотрицательных матриц // Дискретная математика. 1993. Т. 5. Вып. 2. С. 150–159.
9. *Wielandt H.* Unzerlegbare nicht negative Matrizen // Math. Zeitschr. 1950. V. 52 P. 642–648.

СВЕДЕНИЯ ОБ АВТОРАХ

АБРОСИМОВ Михаил Борисович — доцент, кандидат физико-математических наук, доцент Саратовского государственного университета им. Н. Г. Чернышевского, г. Саратов. E-mail: **mic@rambler.ru**

ДЕВЯНИН Петр Николаевич — доктор технических наук, доцент, заместитель заведующего кафедрой Института криптографии, связи и информатики, г. Москва. E-mail: **peter_devyanin@hotmail.com**

ИГНАТЬЕВ Алексей Сергеевич — младший научный сотрудник Института динамики систем и теории управления СО РАН, г. Иркутск. E-mail: **aign@icc.ru**

ПРОКОПЬЕВ Сергей Евгеньевич — г. Москва. E-mail: **prszs@mail.ru**

САЛИЙ Вячеслав Николаевич — профессор, кандидат физико-математических наук, заведующий кафедрой Саратовского государственного университета им. Н. Г. Чернышевского, г. Саратов. E-mail: **saliiVN@info.sgu.ru**

СЕМЕНОВ Александр Анатольевич — кандидат технических наук, заведующий лабораторией дискретного анализа и прикладной логики Института динамики систем и теории управления СО РАН, г. Иркутск. E-mail: **biclop@rambler.ru**

СКОБЕЛЕВ Владимир Владимирович — младший научный сотрудник Института прикладной математики и механики НАН Украины, г. Донецк. E-mail: **skbv@iamm.ac.donetsk.ua**

СКОБЕЛЕВ Владимир Геннадиевич — доктор технических наук, профессор, ведущий научный сотрудник Института прикладной математики и механики НАН Украины, г. Донецк. E-mail: **skbv@iamm.ac.donetsk.ua**

СМЫШЛЯЕВ Станислав Витальевич — студент факультета вычислительной математики и кибернетики Московского государственного университета им. М. В. Ломоносова, г. Москва. E-mail: **smyshsv@gmail.com**

СТЕФАНЦОВ Дмитрий Александрович — аспирант Томского государственного университета, г. Томск. E-mail: **dastephantsov@mail.tsu.ru**

ФИЛИМОНОВ Александр Евгеньевич — студент Томского государственного университета, г. Томск. E-mail: **filimonov1987@gmail.com**

ФОМИЧЕВ Владимир Михайлович — старший научный сотрудник, доцент, доктор физико-математических наук, ведущий научный сотрудник Учреждения Российской академии наук «Институт проблем информатики РАН», г. Москва. E-mail: **fomichev@nm.ru**

АННОТАЦИИ СТАТЕЙ НА АНГЛИЙСКОМ ЯЗЫКЕ

Smyshlyaev S. V. **ON CRYPTOGRAPHIC WEAKNESSES OF SOME CLASSES OF BINARY SEQUENCE TRANSFORMATIONS.** The paper is dedicated to some issues of using perfectly balanced Boolean functions as filtering functions and to some weaknesses in corresponding cryptographic primitives.
Keywords: *perfectly balanced functions, barriers of Boolean functions, filtering generator, cryptography.*

Devyanin P. N. **SECURITY ANALYSIS OF SYSTEMS WITH SIMPLE TRAJECTORIES OF FUNCTIONING WITHIN THE BASE ROLE DP-MODEL.** In the paper, we consider computer systems with only simple trajectories of functioning and with the any number of the cooperated user sessions that do not get access owning to each other by using the information flows by memory to essences functionally associated with the user sessions. For these systems, the conditions for transfering access rights and realizing information flows by memory are formulated and proved within the base role DP-model.
Keywords: *computer security, base role DP-model.*

Prokopyev S. E. **USING TEMPORARY LOGICS AND MODEL CHECKERS FOR DYNAMIC CONTROL ABNORMAL DEVIATIONS OF THE SYSTEM.** We propose to apply the temporary logics and model checkers for dynamic control of "abnormal" deviations of a system by approximating its "normal" features with the temporary logic formulas. Also, we propose an exhaustive blind search algorithm for discovering regularities which can be expressed with the help of the temporal logics.
Keywords: *temporary logics, model checkers, control of system deviations, blind search of regularities.*

Stephantsov D. A., Filimonov A. Y. **INTEGRATING SECURITY POLICIES WITH COMPUTER SYSTEMS BY MEANS OF AOP WITH APPLICATION TO APACHE FTP SERVER.** Recommendations for integrating security policies with computer systems by means of aspect oriented programming are given. It is shown how to integrate the role-based access control policy with the Apache Ftp Server by following this recommendations.
Keywords: *computer systems, security policy, aspect oriented programming, Apache Ftp Server.*

Salii V. N. **FRAME OF AN AUTOMATON.** The frame of an automaton is the partially ordered set of its strongly connected subsets together with the relation of inverse attainability. Some properties of frames are established related to basic algebraic constructions such as subautomata, homomorphisms, and congruences.
Keywords: *automaton, frame of an automaton, subautomaton, homomorphism, congruence, ordered set.*

Skobelev V. V., Skobelev V. G. **ANALYSIS OF NON-LINEAR AUTOMATA WITH DELAY 2 OVER A FINITE RING.** For invertible one-dimensional automata with delay 2 over the ring $\boldsymbol{Z}_{p^k} = (\mathbb{Z}_{p^k}, \oplus, \circ)$, the structure of the transition graph is investigated,

the sets of equivalent states are characterized, the problems of the parametric identification and of the initial state identification are solved, the sets of fixed points of mappings realized by initial automata are characterized.
Keywords: *nonlinear automata, finite rings, simmetric stream ciphers, system of equations over finite rings.*

Ignatiev A. S., Semenov A. A. **ALGORITHMS USING ROBDD AS A BASE FOR BOOLEAN CONSTRAINTS.** In the paper, we study algorithmic properties of ROBDD considered in the role of Boolean constraints in the hybrid (SAT + ROBDD) logical derivation. We suggest ROBDD-analogs for the basic algorithmic procedures used in DPLL-derivation such as variable assignment, unit clause, clause learning, and the techniques of delayed computations. A new algorithm intended for ROBDD reordering is proposed. Computational complexity of all the considered algorithms is provided.
Keywords: *logical equations, binary decision diagrams, hybrid logical derivation.*

Abrosimov M. B. **MINIMAL EDGE EXTENSIONS OF SOME PRECOMPLETE GRAPHS.** We consider the minimal edge k-extensions of precomplete graphs — graphs in which there is a vertex adjacent to all other vertices. The lemma about the marginal value of k when a precomplete graph can have a minimal edge k-extension is proved. The full description of all the minimal edge k-extensions of precomplete graphs being joins of a complete graph and an empty graph, a chain or a cycle is given.
Keywords: *precomplete graph, minimal edge extension, fault tolerance.*

Fomichev V. M. **PROPERTIES OF PATHS IN GRAPHS AND MULTIGRAPHS.** The length of the shortest full path in a strongly connected orgraph with n vertices is estimated. The exponent of the vertex incedent matrix of the graph with self-loop is estimated too. A polynomial estimate is obtained for the subexponent of the system of vertex incedent matrices of n-vertex graphs whose union is a strongly connected graph. These results can be used to investigate essential variables of the coordinate functions which define compositions of transformations of the set of finite words.
Keywords: *full path, shortest path, exponent, subexponent.*

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте vestnik.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также и правила подготовки рукописей статей в журнал.

Тематика публикаций журнала:

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надежности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Дискретные модели реальных процессов*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и ее приложениям*