ISSN 1818-1015
eISSN 2313-5417

2023 Volume 30 No 2

MODELING AND ANALYSIS OF INFORMATION SYSTEMS

SCIENTIFIC JOURNAL

Start date of publication — 1999
Published quarterly

FOUNDER
P.G. Demidov Yaroslavl State University

EDITORIAL OFFICE
14 Sovetskaya str., Yaroslavl 150003, Russian Federation

Website: http://mais-journal.ru
E-mail: mais@uniyar.ac.ru
Phone: +7 (4852) 79-77-73

2023 Том 30 № 2

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

НАУЧНЫЙ ЖУРНАЛ

Издается с 1999 года
Выходит 4 раза в год

УЧРЕДИТЕЛЬ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Ярославский государственный университет им. П. Г. Демидова»

РЕДАКЦИЯ
ул. Советская, 14, Ярославль, 150003, Российская Федерация
Website: http://mais-journal.ru
E-mail: mais@uniyar.ac.ru
Телефон: +7 (4852) 79-77-73

Свидетельство о регистрации СМИ ПИ № ФС77-66186 от 20.06.2016 выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Подписной индекс в каталоге «Урал-Пресс» – 31907. Технический редактор, компьютерная вёрстка – К. В. Лагутина. Подписано в печать 13.06.2023. Дата выхода в свет 30.06.2023. Формат 200×265 мм. Объем 86 с. Тираж 32 экз. Свободная цена. Заказ 23079. Издатель и его адрес: Ярославский государственный университет им. П. Г. Демидова; ул. Советская, 14, Ярославль, 150003, Россия. Типография и ее адрес: ООО «Филигрань»; ул. Свободы, 91, Ярославль, 150049, Россия.
Содержание предназначено для детей старше 12 лет.

Contents

Содержание

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 2, 2023
journal homepage: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

The Zhegalkin Polynomial of Multiseat Sole Sufficient Operator

L. Y. Bystrov[1], E. V. Kuzmin[1]

DOI: 10.18255/1818-1015-2023-2-106-127

[1]P. G. Demidov Yaroslavl State University, 14 Sovetskaya, Yaroslavl 150003, Russia.

MSC2020: 06E30
Research article
Full text in Russian

Received February 27, 2023
After revision May 15, 2023
Accepted May 17, 2023

Among functionally complete sets of Boolean functions, sole sufficient operators are of particular interest. They have a wide range of applicability and are not limited to the two-seat case. In this paper, the conditions, imposed on the Zhegalkin polynomial coefficients, are formulated. The conditions are necessary and sufficient for the polynomial to correspond to a sole sufficient operator. The polynomial representation of constant-preserving Boolean functions is considered. It is shown that the properties of monotone and linearity do not require special consideration in describing a sole sufficient operator. The concept of a dual remainder polynomial is introduced. The value of it allows one to determine the self-duality of a Boolean function. It is proved that the preserving 0 and 1 or preserving neither 0 nor 1 Boolean function is self-dual if and only if the dual remainder of its corresponding Zhegalkin polynomial is equal to 0 for any sets of function variable values. Based on this fact, a system of leading coefficients is obtained. The solution of the system made it possible to formulate the criterion for the self-duality of the Boolean function represented by the Zhegalkin polynomial. It imposes necessary and sufficient conditions on the polynomial coefficients. Thus, it is shown that Zhegalkin polynomials are a rather convenient tool for studying precomplete classes of Boolean functions.

Keywords: Zhegalkin polynomial; sole sufficient operator; Sheffer function; precomplete classes; constant-preserving Boolean functions; self-dual Boolean functions; dual remainder polynomial; leading coefficient

INFORMATION ABOUT THE AUTHORS

Leonid Y. Bystrov | orcid.org/0000-0002-0610-5466. E-mail: bystrovl0306@mail.ru
Undergraduate Student, Chair of Theoretical Informatics.

Egor V. Kuzmin (corresponding author) | orcid.org/0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru
Head of the Chair of Theoretical Informatics, Doctor of Science.

Funding: Yaroslavl State University (project VIP-016).

For citation: L. Y. Bystrov and E. V. Kuzmin, “The Zhegalkin Polynomial of Multiseat Sole Sufficient Operator”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 106-127, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 2, 2023
сайт журнала: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Полином Жегалкина многоместного самодостаточного оператора

Л. Ю. Быстров[1], Е. В. Кузьмин[1]

DOI: 10.18255/1818-1015-2023-2-106-127

[1]Ярославский государственный университет им. П. Г. Демидова, ул. Советская, 14, Ярославль, 150000, Россия.

УДК 510.6
Научная статья
Полный текст на русском языке

Получена 27 февраля 2023 г.
После доработки 15 мая 2023 г.
Принята к публикации 17 мая 2023 г.

Среди полных систем булевых функций особый интерес представляют самодостаточные операторы. Они обладают широкой областью применимости и не ограничиваются двухместным случаем. В данной работе формулируются условия, накладываемые на коэффициенты полинома Жегалкина, необходимые и достаточные для того, чтобы полином соответствовал самодостаточному оператору. Рассмотрено полиномиальное представление булевых функций, сохраняющих константу. Показано, что свойства монотонности и линейности не требуют специального рассмотрения при описании самодостаточного оператора. Вводится понятие полинома двойственного остатка, значение которого позволяет определить самодвойственность булевой функции. Доказано, что сохраняющая 0 и 1 или не сохраняющая ни 0, ни 1 булева функция является самодвойственной тогда и только тогда, когда двойственный остаток соответствующего ей полинома Жегалкина равен 0 для любых наборов значений переменных функции. На основании этого факта получена система ведущих коэффициентов. Решение данной системы позволило сформулировать критерий самодвойственности булевой функции, представленной полиномом Жегалкина, накладывающий необходимые и достаточные условия на коэффициенты полинома. Таким образом, показано, что полиномы Жегалкина являются достаточно удобным инструментом при исследовании предполных классов булевых функций.

Ключевые слова: полином Жегалкина; самодостаточный оператор; функция Шеффера; предполные классы; булевы функции; сохраняющие константу; самодвойственные булевы функции; полином двойственного остатка; ведущий коэффициент

ИНФОРМАЦИЯ ОБ АВТОРАХ

Леонид Юрьевич Быстров	orcid.org/0000-0002-0610-5466. E-mail: bystrovl0306@mail.ru студент, кафедра теоретической информатики.
Егор Владимирович Кузьмин (автор для корреспонденции)	orcid.org/0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru заведующий кафедрой теоретической информатики, доктор физ.-мат. наук.

Финансирование: ЯрГУ (проект VIP-016).

Для цитирования: L. Y. Bystrov and E. V. Kuzmin, “The Zhegalkin Polynomial of Multiseat Sole Sufficient Operator”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 106-127, 2023.

Введение

Набор булевых функций, через суперпозицию которых можно записать любую булеву функцию, называется полной системой булевых функций [1]. Для определения полноты системы булевых функций используется Теорема Поста о полноте, опирающаяся на понятие предполных классов.

Булева функция, сама по себе образующая полную систему булевых функций, называется самодостаточным оператором (sole sufficient operator), или функцией Шеффера [2]. Наиболее известными из самодостаточных операторов являются двухместные функции: штрих Шеффера и стрелка Пирса. Функции Шеффера обладают широкой областью применимости [3].

Использование самодостаточных операторов не ограничено только лишь двухместным случаем. Например, самодостаточным является трехместный (тернарный) оператор А. А. Маркова [4].

Любая булева функция может быть представлена в виде полинома Жегалкина. В статье [5] С. Н. Селезневой было показано, что для систем булевых функций, заданных полиномами Жегалкина, существует алгоритм определения полноты системы с полиномиальной сложностью.

В данной работе исследуется общий вид полинома Жегалкина, реализующего собой самодостаточный многоместный оператор. Сформулирован и доказан ряд утверждений о свойствах булевых функций, представленных в виде полинома Жегалкина, относительно их принадлежности к предполным классам. Особое внимание уделено свойству самодвойственности. Сформулирован критерий самодвойственности булевой функции, заданной полиномом Жегалкина. Наконец, получен общий вид полинома Жегалкина многоместного самодостаточного оператора.

1. Основные понятия

Замыканием $[A]$ называется множество всех булевых функций, представимых в виде суперпозиции функций множества A. Множество A является *функционально замкнутым классом*, если $[A] = A$. Множество называется *полной системой булевых функций*, если любая булева функция может быть записана в виде суперпозиции через функции этой системы [1].

Класс R булевых функций называется *предполным*, если он не полон, но для любой булевой функции f, не принадлежащей этому классу, множество $R \cup \{f\}$ является полным [6]. Эмиль Пост показал, что этому условию удовлетворяют следующие классы [7]: класс S самодвойственных функций, класс M монотонных функций, класс L линейных функций, класс T_0 функций, сохраняющих 0, и класс T_1 функций, сохраняющих 1.

Булева функция $f(x_1, \dots, x_n)$ *сохраняет константу* 0, если $f(0, \dots, 0) = 0$.

Булева функция $f(x_1, \dots, x_n)$ *сохраняет константу* 1, если $f(1, \dots, 1) = 1$.

Функция $f(x_1, \dots, x_n)$ называется *линейной*, если она представима полиномом Жегалкина не выше первой степени [8], т. е. если существуют такие константы $a_i \in \{0, 1\}$, $i \in \overline{0, n}$, что

$$f(x_1, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus \dots \oplus a_n x_n.$$

Для двух наборов значений переменных $\tilde{\alpha} = (\alpha_1, \dots, \alpha_n)$ и $\tilde{\beta} = (\beta_1, \dots, \beta_n)$ выполнено *отношение предшествования* $\tilde{\alpha} \preceq \tilde{\beta}$, если

$$\alpha_1 \leqslant \beta_1, \dots, \alpha_n \leqslant \beta_n.$$

Функция $f(x_1, \dots, x_n)$ называется *монотонной*, если для любых двух наборов $\tilde{\alpha}$ и $\tilde{\beta}$, таких что $\tilde{\alpha} \preceq \tilde{\beta}$, имеет место неравенство

$$f(\tilde{\alpha}) \leqslant f(\tilde{\beta}).$$

Булева функция f^*, $f^*(x_1, \dots, x_n) = \overline{f}(\overline{x_1}, \dots, \overline{x_n})$, называется *двойственной* функцией к функции $f(x_1, \dots, x_n)$. Функция f является *самодвойственной*, если $f = f^*$. Для самодвойственной функции имеет место тождество [1]

$$\overline{f}(\overline{x_1}, \dots, \overline{x_n}) = f(x_1, \dots, x_n).$$

Булева функция называется *антисамодвойственной*, или *самосопряженной*, если выполняется условие [9]

$$f(x_1, \dots, x_n) = f(\overline{x_1}, \dots, \overline{x_n}).$$

Теорема 1. (**Теорема Поста о полноте [1]**) *Для того чтобы система булевых функций была полной, необходимо и достаточно, чтобы она целиком не содержалась ни в одном из классов T_0, T_1, S, M и L.*

Любая булева функция может быть записана в виде формулы, представляющей собой сумму по модулю 2 слагаемых вида $x_{i_1} x_{i_2} \cdots x_{i_s}$ и, быть может, константы 1. Эта формула носит название *полинома Жегалкина* [10]. Полином Жегалкина P булевой функции f от n переменных в общем виде выглядит следующим образом:

$$P(x_1, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n \oplus a_{1,2} x_1 x_2 \oplus a_{1,3} x_1 x_3 \oplus \dots \oplus a_{1,\dots,n} x_1 \cdots x_n, \text{ где } a_0, \dots, a_{1,\dots,n} \in \{0, 1\}.$$

2. Свойства сохранения 0 и 1 в полиномах Жегалкина

Чтобы получить самодостаточный оператор, то есть полную систему булевых функций, состоящую только из одной функции, мы будем поэтапно преобразовывать полином общего вида $P(x_1, \dots, x_n)$, соответствующий булевой функции $f(x_1, \dots, x_n)$, так, чтобы исключить функцию f из каждого из предполных классов.

Лемма 1. *Булева функция f сохраняет 0 тогда и только тогда, когда соответствующий ей полином Жегалкина P имеет нулевой свободный коэффициент:*

$$f \in T_0 \iff a_0 = 0.$$

Доказательство. Функция f, представленная полиномом P, будет сохранять 0, если на нулевом наборе переменных она принимает значение 0. Следовательно, имеют место равенства

$$P(0, \dots, 0) = a_0 = 0.$$

Обратно. Если в полиноме P свободный коэффициент a_0 равен 0, то на нулевом наборе значений переменных полином P будет равен 0. Следовательно, $f(0, \dots, 0) = 0$. □

Лемма 2. *Булева функция f сохраняет 1 тогда и только тогда, когда в соответствующем ей полиноме Жегалкина P число единичных коэффициентов нечётно:*

$$f \in T_1 \iff a_0 \oplus \bigoplus_{\substack{1 \leqslant i_1 < \dots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\dots,i_k} = 1.$$

Доказательство. Для выполнения условия $f \in T_1$ необходимо, чтобы на единичном входном наборе булева функция f принимала значение 1. Рассмотрим для f полином P на этом наборе:

$$P(1, \dots, 1) = a_0 \oplus a_1 \oplus a_2 \oplus \dots \oplus a_n \oplus a_{1,2} \oplus a_{1,3} \oplus \dots \oplus a_{1,\dots,n} = 1; \quad \Longrightarrow \quad P(1, \dots, 1) = \underbrace{1 \oplus \dots \oplus 1}_{\text{нечёт. кол-во}};$$

$$a_0 \oplus \bigoplus_{\substack{1 \leqslant i_1 < \dots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\dots,i_k} = 1.$$

Обратно. Соответствующий функции f полином P с нечётным количеством ненулевых коэффициентов на единичном наборе значений переменных будет равен 1. Действительно,

$$f(1, \dots, 1) = P(1, \dots, 1) = a_0 \oplus a_1 \oplus a_2 \oplus \dots \oplus a_n \oplus a_{1,2} \oplus a_{1,3} \oplus \dots \oplus a_{1,\dots,n} = \underbrace{1 \oplus \dots \oplus 1}_{\text{нечёт. кол-во}} = 1.$$

□

Леммы 1 и 2 показывают, как свойства сохранения 0 и 1 булевой функции f переносятся на соответствующий ей полином P. Самодостаточный оператор не обладает ни одним из этих свойств.

Полином Жегалкина, соответствующий самодостаточному n-местному оператору обозначим через $\hat{P}(x_1, \dots, x_n)$.

Лемма 3. *Для полинома $\hat{P}(x_1, \dots, x_n)$ выполняется:*

1) $a_0 = 1$,
2) $\bigoplus\limits_{\substack{1 \leqslant i_1 < \dots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\dots,i_k} = 1.$

Доказательство. Самодостаточный оператор не сохраняет 0. Следовательно, по Лемме 1 в полиноме $\hat{P}$ свободный коэффициент равен 1, т. е. $a_0 = 1$.

Самодостаточный оператор не сохраняет 1. Следовательно, по Лемме 2 число единичных коэффициентов полинома $\hat{P}$ чётно:

$$a_0 \oplus \bigoplus_{\substack{1 \leqslant i_1 < \dots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\dots,i_k} = 0.$$

Из того факта, что $a_0 = 1$, получаем

$$\bigoplus_{\substack{1 \leqslant i_1 < \dots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\dots,i_k} = 1.$$

□

3. О свойствах монотонности и линейности

Лемма 4. *Булева функция $f(x_1, \dots, x_n)$, не сохраняющая ни 0, ни 1, не является монотонной.*

Доказательство. Предположим противное. Пусть булева функция f, не сохраняющая ни 0, ни 1, является монотонной. Тогда для нулевого и единичного наборов значений переменных $x_1, \dots, x_n$ должно быть справедливо неравенство $f(0, \dots, 0) \preceq f(1, \dots, 1)$. Но поскольку функция f не сохраняет ни 0, ни 1, имеем $f(0, \dots, 0) = 1$ и $f(1, \dots, 1) = 0$. Получили, что $1 \preceq 0$. Пришли к противоречию. Следовательно, рассматриваемая функция f не является монотонной. □

Булева функция, соответствующая полиному $\hat{P}$, не сохраняет ни 0, ни 1. Из Леммы 4 следует, что эта функция не обладает свойством монотонности.

На данном этапе нерассмотренными предполными классами остались класс S самодвойственных функций и класс L линейных функций.

Существует теорема, которая утверждает, что *любая булева функция, не принадлежащая одновременно классам T_0, T_1 и S, не является линейной* (не принадлежит классу L) [11]. Поэтому нет необходимости рассматривать класс L отдельно от класса S. Если исключить функцию f, соответствующую полиному $\hat{P}$, из класса S, то это будет означать и исключение её из класса L.

4. Самодвойственность в полиномах Жегалкина

Перейдём к рассмотрению свойства самодвойственности в полиномах Жегалкина.

4.1. Самодвойственность и полином двойственного остатка

Самодвойственная функция по определению должна либо одновременно сохранять 0 и 1, либо не сохранять ни 0, ни 1.

Определение 1. Пусть P — полином Жегалкина булевой функции f,

$$P(x_1, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n \oplus a_{1,2} x_1 x_2 \oplus a_{1,3} x_1 x_3 \oplus \dots \oplus a_{1,\dots,n} x_1 \cdots x_n, \text{ где } a_0, \dots, a_{1,\dots,n} \in \{0, 1\}.$$

Полином N вида

$$N(x_1, \dots, x_n) = a_{1,2}(x_1 \oplus x_2) \oplus a_{1,3}(x_1 \oplus x_3) \oplus \dots \oplus a_{(n-1),n}(x_{n-1} \oplus x_n) \oplus a_{1,2,3}(x_1 x_2 \oplus x_1 x_3 \oplus x_2 x_3 \oplus x_1 \oplus x_2 \oplus x_3) \oplus$$

$$\oplus \dots \oplus a_{1,\dots,n}(x_1 x_2 \cdots x_{n-1} \oplus x_1 x_2 \cdots x_{n-2} x_n \oplus \dots \oplus x_2 x_3 \cdots x_n \oplus \dots \oplus x_1 \oplus x_2 \oplus \dots \oplus x_n).$$

назовём *полиномом двойственного остатка полинома P*, или *двойственным остатком P*.

Теорема 2. *Пусть булева функция f сохраняет 0 и 1 или не сохраняет ни 0, ни 1. Функция f является самодвойственной тогда и только тогда, когда двойственный остаток N соответствующего ей полинома Жегалкина P равен 0 для любых наборов значений переменных $x_1, x_2, \dots, x_n$.*

Доказательство. Доказательство теоремы разбивается на 2 случая: 1) функция f сохраняет 0 и 1, 2) функция f не сохраняет ни 0, ни 1.

1. Функция f сохраняет и 0, и 1. По Лемме 1 в полиноме P свободный коэффициент a_0 равен 0, а по Лемме 2 сумма коэффициентов полинома нечётна:

$$\bigoplus_{\substack{1 \leqslant i_1 < \dots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\dots,i_k} = 1. \tag{1}$$

Полином P имеет вид

$$P(x_1, \dots, x_n) = a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n \oplus a_{1,2} x_1 x_2 \oplus a_{1,3} x_1 x_3 \oplus \dots \oplus a_{1,\dots,n} x_1 \cdots x_n.$$

Условие самодвойственности функции f может быть переписано в виде

$$\overline{f}(x_1, \dots, x_n) = f(\overline{x_1}, \dots, \overline{x_n}) \iff \overline{P}(x_1, \dots, x_n) = P(\overline{x_1}, \dots, \overline{x_n}). \tag{2}$$

Функции отрицания $\overline{x}$ соответствует полином Жегалкина $x \oplus 1$, поэтому $P(\overline{x_1}, \dots, \overline{x_n})$ можно записать следующим образом:

$$P(\overline{x_1}, \dots, \overline{x_n}) = a_1(x_1 \oplus 1) \oplus a_2(x_2 \oplus 1) \oplus \dots \oplus a_n(x_n \oplus 1) \oplus a_{1,2}(x_1 \oplus 1)(x_2 \oplus 1) \oplus$$

$$\oplus a_{1,3}(x_1 \oplus 1)(x_3 \oplus 1) \oplus \dots \oplus a_{1,\dots,n}(x_1 \oplus 1)(x_2 \oplus 1) \cdots (x_n \oplus 1).$$

Раскроем скобки и сгруппируем полученные слагаемые:

$$P(\overline{x_1}, \dots, \overline{x_n}) = a_1 \oplus a_2 \oplus \dots \oplus a_n \oplus a_{1,2} \oplus a_{1,3} \oplus \dots \oplus a_{1,\dots,n} \oplus$$

$$\oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n \oplus a_{1,2} x_1 x_2 \oplus a_{1,3} x_1 x_3 \oplus \dots \oplus a_{1,\dots,n} x_1 \cdots x_n \oplus$$

$$\oplus a_{1,2}(x_1 \oplus x_2) \oplus a_{1,3}(x_1 \oplus x_3) \oplus \dots \oplus a_{1,\dots,n}(x_1 x_2 \cdots x_{n-1} \oplus x_1 x_2 \cdots x_{n-2} x_n \oplus \dots \oplus x_2 x_3 \cdots x_n \oplus \dots \oplus x_1 \oplus x_2 \oplus \dots \oplus x_n).$$

Мы разделили полином $P(\overline{x_1}, \dots, \overline{x_n})$ на 3 части: сумма коэффициентов полинома, сам полином $P(x_1, \dots, x_n)$ и его двойственный остаток $N(x_1, \dots, x_n)$. Воспользуемся свойством (1) полинома P, чтобы заменить сумму коэффициентов полинома на 1:

$$P(\overline{x_1}, \dots, \overline{x_n}) = \underbrace{a_1 \oplus a_2 \dots \oplus a_n \oplus a_{1,2} \oplus a_{1,3} \oplus \dots \oplus a_{1,\dots,n}}_{1} \oplus$$

$$\oplus \underbrace{a_1x_1 \oplus a_2x_2 \oplus \ldots \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus \ldots \oplus a_{1,\ldots,n}x_1 \cdots x_n}_{P(x_1,\ldots,x_n)} \oplus$$

$$\oplus \underbrace{a_{1,2}(x_1 \oplus x_2) \oplus a_{1,3}(x_1 \oplus x_3) \oplus \ldots \oplus a_{1,\ldots,n}(x_1x_2 \cdots x_{n-1} \oplus x_1x_2 \cdots x_{n-2}x_n \oplus \ldots \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n)}_{N(x_1,\ldots,x_n)}.$$

В результате полином $P(\overline{x_1}, \ldots, \overline{x_n})$ приобретёт вид

$$P(\overline{x_1}, \ldots, \overline{x_n}) = 1 \oplus P(x_1, \ldots, x_n) \oplus N(x_1, \ldots, x_n) = \overline{P}(x_1, \ldots, x_n) \oplus N(x_1, \ldots, x_n).$$

При $N = 0$ выполнено условие самодвойственности (2):

$$P(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}) = \overline{P}(x_1, \ldots, x_n).$$

Таким образом, если функция f является самодвойственной, то двойственный остаток N соответствующего ей полинома P должен быть равен 0 для любых наборов переменных $x_1, x_2, \ldots, x_n$, и наоборот: если двойственный остаток N равен 0 для любых наборов переменных $x_1, x_2, \ldots, x_n$, то выполнено условие самодвойственности.

2. Функция f не сохраняет ни 0, ни 1. По Лемме 1 в полиноме P свободный коэффициент a_0 равен 1, а по Лемме 2 сумма коэффициентов полинома чётна. Следовательно, сумма всех коэффициентов полинома без a_0 нечётна:

$$\bigoplus_{\substack{1 \leqslant i_1 < \ldots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\ldots,i_k} = 1. \tag{3}$$

Полином P имеет вид

$$P(x_1, \ldots, x_n) = 1 \oplus a_1x_1 \oplus a_2x_2 \oplus \ldots \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus \ldots \oplus a_{1,\ldots,n}x_1 \cdots x_n.$$

$P(\overline{x_1}, \ldots, \overline{x_n})$ можно записать как

$$P(\overline{x_1}, \ldots, \overline{x_n}) = 1 \oplus a_1(x_1 \oplus 1) \oplus a_2(x_2 \oplus 1) \oplus \ldots \oplus a_n(x_n \oplus 1) \oplus a_{1,2}(x_1 \oplus 1)(x_2 \oplus 1) \oplus$$

$$\oplus a_{1,3}(x_1 \oplus 1)(x_3 \oplus 1) \oplus \ldots \oplus a_{1,\ldots,n}(x_1 \oplus 1)(x_2 \oplus 1) \cdots (x_n \oplus 1).$$

Раскроем скобки и сгруппируем полученные слагаемые:

$$P(\overline{x_1}, \ldots, \overline{x_n}) = a_1 \oplus a_2 \oplus \ldots \oplus a_n \oplus a_{1,2} \oplus a_{1,3} \oplus \ldots \oplus a_{1,\ldots,n} \oplus$$

$$\oplus 1 \oplus a_1x_1 \oplus a_2x_2 \oplus \ldots \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus \ldots \oplus a_{1,\ldots,n}x_1 \cdots x_n \oplus$$

$$\oplus a_{1,2}(x_1 \oplus x_2) \oplus a_{1,3}(x_1 \oplus x_3) \oplus \ldots \oplus a_{1,\ldots,n}(x_1x_2 \cdots x_{n-1} \oplus x_1x_2 \cdots x_{n-2}x_n \oplus \ldots \oplus x_2x_3 \cdots x_n \oplus \ldots \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n).$$

Аналогично случаю 1 мы разделили полином на 3 части. Используя свойство (3), получим

$$P(\overline{x_1}, \ldots, \overline{x_n}) = \underbrace{a_1 \oplus a_2 \oplus \ldots \oplus a_n \oplus a_{1,2} \oplus a_{1,3} \oplus \ldots \oplus a_{1,\ldots,n}}_{1} \oplus$$

$$\oplus \underbrace{1 \oplus a_1x_1 \oplus a_2x_2 \oplus \ldots \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus \ldots \oplus a_{1,\ldots,n}x_1 \cdots x_n}_{P(x_1,\ldots,x_n)} \oplus$$

$$\oplus \underbrace{a_{1,2}(x_1 \oplus x_2) \oplus a_{1,3}(x_1 \oplus x_3) \oplus \ldots \oplus a_{1,\ldots,n}(x_1x_2 \cdots x_{n-1} \oplus x_1x_2 \cdots x_{n-2}x_n \oplus \ldots \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_n)}_{N(x_1,\ldots,x_n)}.$$

В результате $P(\overline{x_1}, \dots, \overline{x_n})$ приобретёт вид

$$P(\overline{x_1}, \dots, \overline{x_n}) = 1 \oplus P(x_1, \dots, x_n) \oplus N(x_1, \dots, x_n) = \overline{P}(x_1, \dots, x_n) \oplus N(x_1, \dots, x_n).$$

При $N = 0$ выполнено условие самодвойственности (2):

$$P(\overline{x_1}, \overline{x_2}, \dots, \overline{x_n}) = \overline{P}(x_1, \dots, x_n).$$

□

Из Теоремы 2 следует, что самодвойственность в полиномах Жегалкина определяется двойственным остатком. Если двойственный остаток равен 0 для любых наборов значений переменных $x_1, x_2, \dots, x_n$, то можно говорить о самодвойственности функции. Проверка данного условия путём последовательного рассмотрения двойственного остатка на каждом из наборов выглядит трудоёмкой задачей. Поэтому результат Теоремы 2 скорее теоретический, чем практический. Решению этой проблемы посвящён следующий раздел.

4.2. Система ведущих коэффициентов

Обозначим через $P_{i_1,\dots,i_k}$ полином Жегалкина $P'(x_{i_1}, \dots, x_{i_k})$, в котором все коэффициенты, кроме старшего и свободного, единичные:

$$P_{i_1} = P'(x_{i_1}) = 0,$$

$$P_{i_1,\dots,i_k} = P'(x_{i_1}, \dots, x_{i_k}) = x_{i_1} \oplus x_{i_2} \oplus \dots \oplus x_{i_k} \oplus x_{i_1}x_{i_2} \oplus x_{i_1}x_{i_3} \oplus \dots \oplus x_{i_1}x_{i_2} \cdots x_{i_{k-1}} \oplus x_{i_1}x_{i_2} \cdots x_{i_{k-2}}x_{i_k} \oplus \dots \oplus x_{i_2}x_{i_3} \cdots x_{i_k}.$$

Пример 1. Построим полином $P_{1,2,3}$.

Полином Жегалкина $P(x_1, x_2, x_3)$ в общем виде имеет следующее представление:

$$P(x_1, x_2, x_3) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus a_{2,3}x_2x_3 \oplus a_{1,2,3}x_1x_2x_3.$$

В полиноме $P_{1,2,3}$ по определению старший и свободный коэффициенты равны 0: $a_0 = a_{1,2,3} = 0$. Все остальные коэффициенты равны 1. Полином $P_{1,2,3} = P'(x_1, x_2, x_3)$ записывается следующим образом:

$$P_{1,2,3} = x_1 \oplus x_2 \oplus x_3 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3.$$

Двойственный остаток N полинома P можно переписать через полиномы $P_{i_1,\dots,i_k}$:

$$N(x_1, \dots, x_n) = a_{1,2}(x_1 \oplus x_2) \oplus a_{1,3}(x_1 \oplus x_3) \oplus \dots \oplus a_{1,\dots,n}(x_1x_2 \cdots x_{n-1} \oplus x_1x_2 \cdots x_{n-2}x_n \oplus \dots \oplus x_1 \oplus x_2 \oplus \dots \oplus x_n) \iff$$

$$\iff N(x_1, \dots, x_n) = a_{1,2}P_{1,2} \oplus a_{1,3}P_{1,3} \oplus \dots \oplus a_{1,\dots,n}P_{1,\dots,n}.$$

Лемма 5. *Число слагаемых полинома $P_{i_1,\dots,i_k}$ чётно.*

Доказательство. Число слагаемых полинома $P_{i_1,\dots,i_k}$ можно найти через сумму числа сочетаний, где каждому сочетанию C_k^i соответствует число способов сформировать конъюнкцию i-ой степени из k переменных:

$$C_k^1 + C_k^2 + \dots + C_k^{k-1} = 2^k - 2.$$

□

Теорема 3. *Функция, соответствующая полиному Жегалкина $P_{i_1,\dots,i_k}$, на любом наборе значений переменных, кроме нулевого и единичного, равна 1.*

Доказательство. На нулевом наборе все слагаемые полинома $P_{i_1,\dots,i_k}$ равны 0. Значит, $P'(0,\dots,0) = 0$. На единичном наборе все слагаемые полинома $P_{i_1,\dots,i_k}$ равны 1. По Лемме 5 число всех слагаемых полинома $P_{i_1,\dots,i_k}$ чётно. Следовательно, $P'(1,\dots,1) = 0$.

Теперь рассмотрим произвольный набор значений переменных $\tilde{\alpha} = (\alpha_{i_1},\dots,\alpha_{i_k})$, для которого выполняется условие, что $\exists t, p \in \overline{1,k},\ t \neq p\colon\ \alpha_{i_t} = 1,\ \alpha_{i_p} = 0$.

В полиноме $P_{i_1,\dots,i_k}$ конъюнкции, содержащие переменную x_{i_p}, равны 0 для всех найденных p. Рассмотрим полином $P_{i_1,\dots,i_k}$ без этих конъюнкций (считаем их равными 0). Полином содержит только конъюнкции с переменными x_{i_t} для найденных t. Будем считать, что таких t было найдено m штук, где $m < k$:

$$P_{i_1,\dots,i_k}(\tilde{\alpha}) = P'(\alpha_{i_{t_1}},\dots,\alpha_{i_{t_m}}) \oplus \alpha_{i_{t_1}} \cdots \alpha_{i_{t_m}} = P_{i_{t_1},\dots,i_{t_m}} \oplus \alpha_{i_{t_1}} \cdots \alpha_{i_{t_m}}.$$

По Лемме 5 число слагаемых $P_{i_{t_1},\dots,i_{t_m}}$ чётно. Все слагаемые $P_{i_{t_1},\dots,i_{t_m}}$ на наборе $\tilde{\alpha}$ равны 1. Следовательно, $P_{i_{t_1},\dots,i_{t_m}} = 0$.

Все члены конъюнкции $\alpha_{i_{t_1}} \cdots \alpha_{i_{t_m}}$ равны 1. Следовательно, сама конъюнкция равна 1. Таким образом, полином $P_{i_1,\dots,i_k}$ на наборе $\tilde{\alpha}$ равен 1.

□

Определение 2. Коэффициент $a_{i_1,\dots,i_k}$ полинома двойственного остатка $N(x_1,\dots,x_n)$ назовём *ведущим на наборе* $\tilde{\alpha} = (\alpha_1,\dots,\alpha_n)$, $k \leqslant n$, если соответствующий этому коэффициенту полином $P_{i_1,\dots,i_k}$ на наборе $\tilde{\alpha}$ принимает значение 1.

Теорема 2 утверждает, что самодвойственность функции f определяется двойственным остатком N соответствующего функции f полинома P. Если для всех наборов переменных $x_1, x_2, \dots, x_n$ двойственный остаток N равен 0, то сохраняющая 0 и 1 или не сохраняющая ни 0, ни 1 функция f является самодвойственной. Значение полинома N на некотором наборе переменных $(\alpha_1,\dots,\alpha_n)$ определяется ведущими на этом наборе коэффициентами.

Равенство нулю двойственного остатка N для любого набора переменных $x_1, x_2, \dots, x_n$ означает, что вектор значений коэффициентов $a_{1,2}, a_{1,3}, \dots, a_{1,\dots,n}$ является решением системы:

$$\begin{cases} N(0,0,0,\dots,0,0) = 0, \\ N(1,0,0,\dots,0,0) = 0, \\ N(0,1,0,\dots,0,0) = 0, \\ N(1,1,0,\dots,0,0) = 0, \\ \dots \\ N(1,0,1,\dots,1,1) = 0, \\ N(0,1,1,\dots,1,1) = 0, \\ N(1,1,1,\dots,1,1) = 0. \end{cases} \tag{4}$$

Для того чтобы определить, является ли коэффициент $a_{i_1,\dots,i_k}$ ведущим на наборе $\tilde{\alpha}$, необходимо вычислить значение полинома $P_{i_1,\dots,i_k}$ на этом наборе $\tilde{\alpha}$. Следующая теорема позволяет находить ведущие коэффициенты без использования полиномов $P_{i_1,\dots,i_k}$.

Теорема 4. *Коэффициент $a_{i_1,\dots,i_k}$ полинома двойственного остатка $N(x_1,\dots,x_n)$ является ведущим на наборе $\tilde{\alpha} = (\alpha_1,\dots,\alpha_n)$, если $\exists\ t, p \in \overline{1,k},\ t \neq p\ :\ \alpha_{i_t} = 1$ и $\alpha_{i_p} = 0$.*

Коэффициент $a_{i_1,\dots,i_k}$ не является ведущим на наборе $\tilde{\alpha}$, если для $\forall t \in \overline{1,k}$ выполнено $\alpha_{i_t} = 1$ или если для $\forall p \in \overline{1,k}$ выполнено $\alpha_{i_p} = 0$.

Доказательство. Если $\forall t \in \overline{1,k}$ имеем $\alpha_{i_t} = 1$, то для полинома $P_{i_1,\dots,i_k}$ набор $\tilde{\alpha}$ является единичным. Из Теоремы 3 следует, что $P_{i_1,\dots,i_k} = 0$. По определению $a_{i_1,\dots,i_k}$ не является ведущим коэффициентом.

Если $\forall p \in \overline{1,k}$ имеем $\alpha_{i_p} = 0$, то для полинома $P_{i_1,\dots,i_k}$ набор $\tilde{\alpha}$ является нулевым. Из Теоремы 3 следует, что $P_{i_1,\dots,i_k} = 0$. Таким образом, $a_{i_1,\dots,i_k}$ — не ведущий коэффициент.

Теперь пусть $\exists\ t, p \in \overline{1,k},\ t \neq p : \alpha_{i_t} = 1$ и $\alpha_{i_p} = 0$. Для полинома $P_{i_1,\dots,i_k}$ набор $\tilde{\alpha}$ не является ни единичным, ни нулевым. Следовательно, по Теореме 3 полином $P_{i_1,\dots,i_k}$ на наборе $\tilde{\alpha}$ равен 1. Получили, что $a_{i_1,\dots,i_k}$ — ведущий коэффициент.

□

Теорема 4 позволяет переписать систему (4) через ведущие коэффициенты. Перед тем как это сделать, систему (4) можно упростить, используя следующую теорему:

Теорема 5. *Полином двойственного остатка N любого полинома Жегалкина P обладает свойством антисамодвойственности. Причём если коэффициент $a_{i_1,\dots,i_k}$ был ведущим на наборе значений переменных $(\alpha_1,\dots,\alpha_n)$, то он является ведущим и на противоположном наборе $(\overline{\alpha_1},\dots,\overline{\alpha_n})$.*

Доказательство. Рассмотрим $N(\overline{x_1},\dots,\overline{x_n})$:

$$N(\overline{x_1},\dots,\overline{x_n}) = a_{1,2}P'(\overline{x_1},\overline{x_2}) \oplus a_{1,3}P'(\overline{x_1},\overline{x_3}) \oplus \dots \oplus a_{1,\dots,n}P'(\overline{x_1},\dots,\overline{x_n}).$$

Из Теоремы 3 следует, что полином $P_{i_1,\dots,i_k}$, где $k \leqslant n$, равен 0 на нулевом и единичном, то есть противоположном к нулевому, наборах переменных. По той же Теореме значение полинома $P_{i_1,\dots,i_k}$ на любом другом наборе и противоположном к нему равно 1. Следовательно, $P_{i_1,\dots,i_k}$ обладает свойством антисамодвойственности:

$$P'(\overline{x_{i_1}},\dots,\overline{x_{i_k}}) = P'(x_{i_1},\dots,x_{i_k}) = P_{i_1,\dots,i_k}.$$

Полином двойственного остатка N также обладает свойством антисамодвойственности. Причём все ведущие коэффициенты на некотором наборе переменных $\tilde{\alpha}$ остаются ведущими и на противоположном к $\tilde{\alpha}$ наборе:

$$N(\overline{x_1},\dots,\overline{x_n}) = a_{1,2}P_{1,2} \oplus a_{1,3}P_{1,3} \oplus \dots \oplus a_{1,\dots,n}P_{1,\dots,n} = N(x_1,\dots,x_n).$$

□

Из Теоремы 5 следует, что система (4) содержит одинаковые уравнения. Так как N на любом наборе значений переменных $\tilde{\alpha}$ имеет такой же вид как на противоположном к $\tilde{\alpha}$ наборе, в системе (4) достаточно рассмотреть только половину всех входных наборов. Следующая теорема показывает, что выбор таковой половины можно проводить относительно любой переменной $x_1, x_2, \dots, x_n$.

Лемма 6.

$$\forall i,j \in \overline{1,n} : \forall(\alpha_1,\alpha_2,\dots,\alpha_{i-1},\alpha_{i+1},\dots,\alpha_n)\ \exists\ (\beta_1,\beta_2,\dots,\beta_{j-1},\beta_{j+1},\dots,\beta_n):$$

$$N(\alpha_1,\alpha_2,\dots,\alpha_{i-1},0,\alpha_{i+1},\dots,\alpha_n) = N(\beta_1,\beta_2,\dots,\beta_{j-1},0,\beta_{j+1},\dots,\beta_n).$$

Доказательство. Если $i = j$, то равенство выполняется при $\beta_1 = \alpha_1, \dots, \beta_{i-1} = \alpha_{i-1}, \beta_{i+1} = \alpha_{i+1}, \dots, \beta_n = \alpha_n$:

$$N(\alpha_1,\alpha_2,\dots,\alpha_{i-1},0,\alpha_{i+1},\dots,\alpha_n) = N(\alpha_1,\alpha_2,\dots,\alpha_{i-1},0,\alpha_{i+1},\dots,\alpha_n).$$

Пусть $i \neq j$. Не ограничивая общности рассуждений, положим, что $i < j$. Если $\alpha_j = 0$, то равенство выполняется при $\beta_1 = \alpha_1, \beta_2 = \alpha_2, \dots, \beta_{i-1} = \alpha_{i-1}, \beta_i = 0, \beta_{i+1} = \alpha_{i+1}, \dots, \beta_n = \alpha_n$:

$$N(\alpha_1,\alpha_2,\dots,\alpha_{i-1},0,\alpha_{i+1},\dots,\alpha_{j-1},0,\alpha_{j+1},\dots,\alpha_n) = N(\alpha_1,\alpha_2,\dots,\alpha_{i-1},0,\alpha_{i+1},\dots,\alpha_{j-1},0,\alpha_{j+1},\dots,\alpha_n).$$

Если $\alpha_j = 1$, то $\overline{\alpha_j} = 0$. Из Теоремы 5 получаем

$$N(\alpha_1, \alpha_2, \dots, \alpha_{i-1}, 0, \alpha_{i+1}, \dots, \alpha_{j-1}, 1, \alpha_{j+1}, \dots, \alpha_n) = N(\overline{\alpha_1}, \overline{\alpha_2}, \dots, \overline{\alpha_{i-1}}, 1, \overline{\alpha_{i+1}}, \dots, \overline{\alpha_{j-1}}, 0, \overline{\alpha_{j+1}}, \dots, \overline{\alpha_n}).$$

Утверждение Леммы выполняется, если в качестве β выбрать следующие числа: $\beta_1 = \overline{\alpha_1}, \beta_2 = \overline{\alpha_2}, \dots, \beta_{i-1} = \overline{\alpha_{i-1}}, \beta_i = 1, \beta_{i+1} = \overline{\alpha_{i+1}}, \dots, \beta_n = \overline{\alpha_n}$. □

Благодаря Лемме 6 для исключения повторяющихся уравнений в системе (4) достаточно рассмотреть наборы, где одна переменная равна 0. Пусть этой переменной будет x_n. Сделаем ещё одно упрощение: так как на нулевом наборе значений переменных никакой коэффициент не является ведущим, исключим этот набор из системы (4). Получим систему вида:

$$\begin{cases} N(1, 0, 0, \dots, 0, 0) = 0, \\ N(0, 1, 0, \dots, 0, 0) = 0, \\ N(1, 1, 0, \dots, 0, 0) = 0, \\ \dots \\ N(0, 1, 1, \dots, 1, 0) = 0, \\ N(1, 1, 1, \dots, 1, 0) = 0. \end{cases} \tag{5}$$

Теперь, используя Теорему 4, каждое уравнение системы (5) можно переписать через сумму ведущих коэффициентов:

$$\begin{cases} a_{1,2} \oplus a_{1,3} \oplus \dots \oplus a_{1,n} \oplus \dots \oplus a_{1,3,4,\dots,n} \oplus a_{1,2,3,\dots,n} = 0, \\ a_{1,2} \oplus a_{2,3} \oplus \dots \oplus a_{2,n} \oplus \dots \oplus a_{2,3,4,\dots,n} \oplus a_{1,2,3,\dots,n} = 0, \\ a_{1,3} \oplus a_{1,4} \oplus \dots \oplus a_{1,n} \oplus a_{2,3} \oplus a_{2,4} \oplus \dots \oplus a_{2,n} \oplus \dots \oplus a_{2,3,4,\dots,n} \oplus a_{1,2,3,\dots,n} = 0, \\ \dots\dots\dots \\ a_{1,2} \oplus a_{1,3} \oplus \dots \oplus a_{1,(n-1)} \oplus a_{2,n} \oplus a_{3,n} \oplus a_{(n-1),n} \oplus \dots \oplus a_{2,3,4,\dots,n} \oplus a_{1,2,3,\dots,n} = 0, \\ a_{1,n} \oplus a_{2,n} \oplus \dots \oplus a_{(n-1),n} \oplus \dots \oplus a_{2,3,4,\dots,n} \oplus a_{1,2,3,\dots,n} = 0. \end{cases} \tag{6}$$

4.3. Критерий самодвойственности

Система (6) представляет собой систему линейных булевых уравнений. Чтобы решить полученную систему, приведём её к ступенчатому виду, используя метод, близкий к классическому методу Гаусса для решения систем линейных алгебраических уравнений [12]. Сначала для каждого уравнения системы (6) построим прямые суммы $S_d^{i_1,i_2,\dots,i_k}$, а затем обратные суммы $S_r^{i_1,i_2,\dots,i_k}$, $k \in \overline{1, n-1}$.

Замечание. С этого момента в индексах коэффициентов, наборов переменных и сумм могут использоваться операции над множествами, чтобы показать, что расширение индекса может происходить добавлением не только последовательно идущих чисел, но любых чисел, отсутствующих в исходном индексе.

Обозначим через $\alpha^{i_1,i_2,\dots,i_k}$ набор значений переменных $x_1, x_2, \dots, x_n$, в котором $x_{i_1} = 1$, $x_{i_2} = 1, \dots, x_{i_k} = 1$, а все остальные переменные в наборе равны 0.

Через $S_d^{i_1,i_2,\dots,i_k}$ обозначим сумму

$$N(\alpha^{i_1}) \oplus N(\alpha^{i_2}) \oplus \dots \oplus N(\alpha^{i_k}) \oplus N(\alpha^{i_1,i_2}) \oplus N(\alpha^{i_1,i_3}) \oplus \dots \oplus N(\alpha^{i_{k-1},i_k}) \oplus \dots \oplus N(\alpha^{i_1,i_2,\dots,i_{k-1}}) \oplus$$

$$\oplus N(\alpha^{i_1,i_2,\dots,i_{k-2},i_k}) \oplus \dots \oplus N(\alpha^{i_2,i_3,\dots,i_k}) \oplus N(\alpha^{i_1,i_2,\dots,i_k}),$$

где N — полином двойственного остатка, $k \in \overline{1, n-1}$.

Через $S_r^{i_1,i_2,\ldots,i_k}$, $k \in \overline{1,n-1}$, обозначим сумму

$$S_r^{1,2,\ldots,(n-1)} = S_d^{1,2,\ldots,(n-1)},$$

$$S_r^{i_1,i_2,\ldots,i_k} = S_d^{i_1,i_2,\ldots,i_k} \oplus \bigoplus_{\substack{1\leqslant i_{k+1}<i_{k+2}<\ldots<i_{k+l}\leqslant n-1 \\ \{i_{k+1},i_{k+2},\ldots,i_{k+l}\}\cap\{i_1,i_2,\ldots,i_k\}=\varnothing \\ l\in\overline{1,n-1-k}}} S_r^{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},\ldots,i_{k+l}\}},\ k \in \overline{1,n-2}.$$

Последняя запись означает, что при формировании суммы $S_r^{i_1,i_2,\ldots,i_k}$ учитываются суммы S_r большего порядка, степень которых образуется из чисел i_1, i_2, …, i_k и чисел 1, 2, …, n, не вошедших в множество $\{i_1, i_2, \ldots, i_k\}$

Пример 2. Решим систему (6) для n, равного 4, используя прямые и обратные суммы S_d и S_r.

$$\begin{cases} N(1,0,0,0) = 0, \\ N(0,1,0,0) = 0, \\ N(1,1,0,0) = 0, \\ N(0,0,1,0) = 0, \\ N(1,0,1,0) = 0, \\ N(0,1,1,0) = 0, \\ N(1,1,1,0) = 0. \end{cases} \Leftrightarrow \begin{cases} a_{1,2} \oplus a_{1,3} \oplus a_{1,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,2} \oplus a_{2,3} \oplus a_{2,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,3} \oplus a_{1,4} \oplus a_{2,3} \oplus a_{2,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,3,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,3} \oplus a_{2,3} \oplus a_{3,4} \oplus a_{1,2,3} \oplus a_{1,3,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,2} \oplus a_{1,4} \oplus a_{2,3} \oplus a_{3,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,3,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,2} \oplus a_{2,4} \oplus a_{1,3} \oplus a_{3,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,3,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,4} \oplus a_{2,4} \oplus a_{3,4} \oplus a_{1,2,4} \oplus a_{1,3,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0. \end{cases}$$

Образуем прямые суммы S_d:

$$S_d^1 = N(\alpha^1),$$
$$S_d^2 = N(\alpha^2),$$
$$S_d^{1,2} = N(\alpha^1) \oplus N(\alpha^2) \oplus N(\alpha^{1,2}),$$
$$S_d^3 = N(\alpha^3),$$
$$S^{1,3} = N(\alpha^1) \oplus N(\alpha^3) \oplus N(\alpha^{1,3}),$$
$$S_d^{2,3} = N(\alpha^2) \oplus N(\alpha^3) \oplus N(\alpha^{2,3}),$$
$$S_d^{1,2,3} = N(\alpha^1) \oplus N(\alpha^2) \oplus N(\alpha^3) \oplus N(\alpha^{1,2}) \oplus N(\alpha^{1,3}) \oplus N(\alpha^{2,3}) \oplus N(\alpha^{1,2,3}).$$

Преобразуем систему через суммирование строк системы по правилам прямых сумм S_d, описанным выше:

$$\begin{cases} S_d^1 = 0, \\ S_d^2 = 0, \\ S_d^{1,2} = 0, \\ S_d^3 = 0, \\ S_d^{1,3} = 0, \\ S_d^{2,3} = 0, \\ S_d^{1,2,3} = 0. \end{cases} \Leftrightarrow \begin{cases} a_{1,2} \oplus a_{1,3} \oplus a_{1,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,2} \oplus a_{2,3} \oplus a_{2,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,3} \oplus a_{2,3} \oplus a_{3,4} \oplus a_{1,2,3} \oplus a_{1,3,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,2,3} \oplus a_{1,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,2,3} \oplus a_{2,3,4} \oplus a_{1,2,3,4} = 0, \\ a_{1,2,3,4} = 0. \end{cases}$$

Теперь образуем обратные суммы S_r:

$$S_r^{1,2,3} = S_d^{1,2,3} = a_{1,2,3,4},$$

$$S_r^{2,3} = S_d^{2,3} \oplus S_r^{1,2,3} = a_{1,2,3} \oplus a_{2,3,4} \oplus a_{1,2,3,4} \oplus a_{1,2,3,4} = a_{1,2,3} \oplus a_{2,3,4},$$

$$S_r^{1,3} = S_d^{1,3} \oplus S_d^{1,2,3} = a_{1,2,3} \oplus a_{1,3,4} \oplus a_{1,2,3,4} \oplus a_{1,2,3,4} = a_{1,2,3} \oplus a_{1,3,4},$$

$$S_r^3 = S_d^3 \oplus S_r^{1,3} \oplus S_r^{2,3} \oplus S_r^{1,2,3} = a_{1,3} \oplus a_{2,3} \oplus a_{3,4} \oplus a_{1,2,3} \oplus a_{1,3,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} \oplus a_{1,2,3} \oplus a_{1,3,4} \oplus a_{1,2,3} \oplus a_{2,3,4} \oplus a_{1,2,3,4} =$$

$$= a_{1,3} \oplus a_{2,3} \oplus a_{3,4} \oplus a_{1,2,3},$$

$$S_r^{1,2} = S_d^{1,2} \oplus S_r^{1,2,3} = a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,2,3,4} \oplus a_{1,2,3,4} = a_{1,2,3} \oplus a_{1,2,4},$$

$$S_r^2 = S_d^2 \oplus S_r^{1,2} \oplus S_r^{2,3} \oplus S_r^{1,2,3} = a_{1,2} \oplus a_{2,3} \oplus a_{2,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{2,3,4} \oplus a_{1,2,3,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,2,3} \oplus a_{2,3,4} \oplus a_{1,2,3,4} =$$

$$= a_{1,2} \oplus a_{2,3} \oplus a_{2,4} \oplus a_{1,2,3},$$

$$S_r^1 = S_d^1 \oplus S_r^{1,2} \oplus S_r^{1,3} \oplus S_r^{1,2,3} = a_{1,2} \oplus a_{1,3} \oplus a_{1,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,3,4} \oplus a_{1,2,3,4} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus a_{1,2,3} \oplus a_{1,3,4} \oplus a_{1,2,3,4} =$$

$$= a_{1,2} \oplus a_{1,3} \oplus a_{1,4} \oplus a_{1,2,3}.$$

На месте каждой прямой суммы S_d в системе сформируем сумму S_r по правилам, указанным выше:

$$\begin{cases} S_d^{1,2,3} = 0, \\ S_d^{2,3} = 0, \\ S_d^{1,3} = 0, \\ S_d^3 = 0, \\ S_d^{1,2} = 0, \\ S_d^2 = 0, \\ S_d^1 = 0. \end{cases} \Leftrightarrow \begin{cases} S_r^{1,2,3} = 0, \\ S_r^{2,3} = 0, \\ S_r^{1,3} = 0, \\ S_r^3 = 0, \\ S_r^{1,2} = 0, \\ S_r^2 = 0, \\ S_r^1 = 0. \end{cases} \Leftrightarrow \begin{cases} a_{1,2,3,4} = 0, \\ a_{1,2,3} \oplus a_{2,3,4} = 0, \\ a_{1,2,3} \oplus a_{1,3,4} = 0, \\ a_{1,3} \oplus a_{2,3} \oplus a_{3,4} \oplus a_{1,2,3} = 0, \\ a_{1,2,3} \oplus a_{1,2,4} = 0, \\ a_{1,2} \oplus a_{2,3} \oplus a_{2,4} \oplus a_{1,2,3} = 0, \\ a_{1,2} \oplus a_{1,3} \oplus a_{1,4} \oplus a_{1,2,3} = 0. \end{cases}$$

В качестве свободных переменных в векторе решения возьмём $a_{1,2}$, $a_{1,3}$, $a_{2,3}$, $a_{1,2,3}$. Тогда решение системы можно выписать следующим образом:

$$a_{1,2,3,4} = 0,$$

$$a_{2,3,4} = a_{1,3,4} = a_{1,2,4} = a_{1,2,3},$$

$$a_{3,4} = a_{1,3} \oplus a_{2,3} \oplus a_{1,2,3},$$

$$a_{2,4} = a_{1,2} \oplus a_{2,3} \oplus a_{1,2,3},$$

$$a_{1,4} = a_{1,2} \oplus a_{1,3} \oplus a_{1,2,3}.$$

Определение 3. Коэффициент $a_{j_1,j_2,\ldots,j_m}$, $m \leqslant n$, полинома двойственного остатка $N(x_1, \ldots, x_n)$ назовём *ведущим на сумме* $S_d^{i_1,i_2,\ldots,i_k}$ ($S_r^{i_1,i_2,\ldots,i_k}$), $k \in \overline{1, n-1}$, если он является ведущим коэффициентом на нечётном числе наборов соответствующих слагаемых суммы $S_d^{i_1,i_2,\ldots,i_k}$ (на нечётном числе слагаемых суммы $S_r^{i_1,i_2,\ldots,i_k}$).

Использование чётности в определении ведущего на сумме коэффициента естественно возникает из того факта, что чётное количество равных слагаемых в сумме по модулю 2 даёт 0. Соответственно, чтобы коэффициент присутствовал в сумме, необходимо, чтобы он входил в нечётное число слагаемых.

Число слагаемых суммы $S_d^{i_1,i_2,\ldots,i_k}$ может быть довольно большим. При определении ведущего на сумме коэффициента перебор всех слагаемых суммы выглядит трудоёмкой задачей. Сократим этот перебор. Для этого сумму $S_d^{i_1,i_2,\ldots,i_k}$ разобьём на 2 части. Первая часть S_1 будет содержать слагаемые, соответствующие наборам, где переменная x_{i_1} равна 0, а вторая S_2 — слагаемые, соответствующие наборам, где переменная x_{i_1} равна 1. Подобное разбиение можно проводить относительно любой переменной $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$, но для определённости была выбрана переменная x_{i_1}.

Определение 4. Набор $\tilde{\beta} = (\beta_1, \beta_2, \dots, \beta_n)$ назовём *парным* к набору $\tilde{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_n)$ *относительно переменной* x_k, $k \in \overline{1,n}$, если $\tilde{\beta}$ отличается от $\tilde{\alpha}$ только значением переменной x_k

$$\beta_1 = \alpha_1, \beta_2 = \alpha_2, \dots, \beta_{k-1} = \alpha_{k-1}, \beta_k = \overline{\alpha_k}, \beta_{k+1} = \alpha_{k+1}, \dots, \beta_n = \alpha_n.$$

Далее, будем рассматривать парные наборы относительно только переменной x_{i_1} ввиду того, что разбиение суммы $S_d^{i_1,i_2,\dots,i_k}$ на S_1 и S_2 было проведено относительно этой переменной.

Лемма 7. *Для выбранного числа $i_1 \in \{j_1, j_2, \dots, j_m\}$ коэффициент $a_{j_1,j_2,\dots,j_m}$, $m \leqslant n$, является ведущим на сумме $S_d^{i_1,i_2,\dots,i_k}$ тогда и только тогда, когда он является ведущим на нечётном числе наборов, для которых*

$$x_{j_1} = 1, x_{j_2} = 1, \dots, x_{i_1} = 0, \dots, x_{j_m} = 1 \tag{7}$$

или

$$x_{j_1} = 0, x_{j_2} = 0, \dots, x_{i_1} = 1, \dots, x_{j_m} = 0 \tag{8}$$

и значения прочих переменных подобраны таким образом, чтобы полученные наборы соответствовали слагаемым суммы $S_d^{i_1,i_2,\dots,i_k}$.

Доказательство. Покажем, что из нечётности числа наборов вида (7) и (8), соответствующих слагаемым суммы $S_d^{i_1,i_2,\dots,i_k}$, следует, что коэффициент $a_{j_1,j_2,\dots,j_m}$ является ведущим на сумме $S_d^{i_1,i_2,\dots,i_k}$.

Если некоторый коэффициент $a_{j_1,j_2,\dots,j_m}$ являлся ведущим на каком-то наборе слагаемого суммы S_1 и на парном к нему наборе слагаемого суммы S_2, то он является ведущим на двух наборах суммы $S_d^{i_1,i_2,\dots,i_k}$. Соответственно, число всех парных наборов слагаемых сумм S_1 и S_2, на которых коэффициент $a_{j_1,j_2,\dots,j_m}$ будет ведущим, является чётным. Поэтому для проверки чётности числа вообще всех наборов в сумме $S_d^{i_1,i_2,\dots,i_k}$, на которых рассматриваемый коэффициент является ведущим, будет достаточно рассмотреть только наборы специального вида, то есть такие наборы, на которых некоторый коэффициент является ведущим только в одной из сумм S_1 или S_2 и не является ведущим на парных к ним наборах в другой сумме. Так как число парных наборов всегда чётное, необходимо определить чётность числа наборов специального вида. Если их количество будет нечётным, то сумма нечётного и чётного чисел даст нечётное число и коэффициент $a_{j_1,j_2,\dots,j_m}$ будет ведущим на сумме $S_d^{i_1,i_2,\dots,i_k}$. И наоборот: если количество специальных наборов будет чётным, то вместе с чётным числом парных наборов они дадут в сумме чётное число, и коэффициент $a_{j_1,j_2,\dots,j_m}$ не будет ведущим на сумме.

По Теореме 4 коэффициент $a_{j_1,j_2,\dots,j_m}$, $m \leqslant n$, является ведущим на наборе $\tilde{\alpha}$, если $\exists t, p \in \overline{1,n}, t \neq p$: $\alpha_{j_t} = 1$ и $\alpha_{j_p} = 0$. То есть он является ведущим на наборе $\tilde{\alpha}$, если среди значений переменных $x_{j_1}, x_{j_2}, \dots, x_{j_m}$ в наборе есть хотя бы один ноль и хотя бы одна единица. Соответственно, коэффициент $a_{j_1,j_2,\dots,j_m}$ не является ведущим на наборе $\tilde{\alpha}$, когда все значения переменных $x_{j_1}, x_{j_2}, \dots, x_{j_m}$ в этом наборе равны 1 или равны 0. Такие наборы будут парными относительно переменной x_{i_1} для наборов соответствующих слагаемых S_1 вида $x_{j_1} = 1, x_{j_2} = 1, \dots, x_{i_1} = 0, \dots, x_{j_m} = 1$ и наборов соответствующих слагаемых S_2 вида $x_{j_1} = 0, x_{j_2} = 0, \dots, x_{i_1} = 1, \dots, x_{j_m} = 0$. Как можно видеть, коэффициент $a_{j_1,j_2,\dots,j_m}$ по Теореме 4 является ведущим на наборах вида (7) и (8). Если таких наборов нечётное число, то коэффициент $a_{j_1,j_2,\dots,j_m}$ является ведущим на сумме $S_d^{i_1,i_2,\dots,i_k}$, в противном случае — не является ведущим.

Докажем утверждение Леммы в обратную сторону. Пусть коэффициент $a_{j_1,j_2,\dots,j_m}$ является ведущим на сумме $S_d^{i_1,i_2,\dots,i_k}$. По определению это означает, что коэффициент $a_{j_1,j_2,\dots,j_m}$ является ведущим на нечётном числе наборов соответствующих слагаемых суммы $S_d^{i_1,i_2,\dots,i_k}$. Среди этих наборов присутствуют парные наборы слагаемых сумм S_1 и S_2, на которых коэффициент $a_{j_1,j_2,\dots,j_m}$ является ведущим. Таких наборов всегда чётное число. Поэтому, исключив их из общего нечётного числа наборов, для которых коэффициент $a_{j_1,j_2,\dots,j_m}$ ведущий, получим нечётное число оставшихся наборов.

Такими наборами будут наборы, на которых коэффициент $a_{j_1,j_2,\ldots,j_m}$ является ведущим только в одной из сумм S_1 и S_2 и не является ведущим на наборах, парных к данным. Как было показано ранее (в первой части доказательства), такие наборы имеют специальный вид (7) и (8). Следовательно, число наборов вида (7) и (8) в сумме нечётно. □

Используя Лемму 7, докажем следующую теорему.

Теорема 6. *На сумме $S_d^{i_1,i_2,\ldots,i_k}$ только коэффициенты вида $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},i_{k+2},\ldots,i_{k+l}\}}, \forall l \in \overline{1,n-k}$, являются ведущими.*

Доказательство. Запись $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},i_{k+2},\ldots,i_{k+l}\}}$ означает, что в индексации коэффициента $a_{\{i_1,i_2,\ldots,i_k\}}$ мы добавляем числа $1, 2, \ldots, n$, отличные от $i_1, i_2, \ldots, i_k$, в количестве от 1 до l.

Рассмотрим случай при $k = 1$: $S_d^{i_1} = N(\alpha^{i_1})$. Сумма $S_d^{i_1}$ состоит только из одного слагаемого. На наборе, соответствующем этому слагаемому, $x_{i_1} = 1$ и $\forall j \in \overline{2,n} : x_{i_j} = 0$. По Теореме 4 ведущими на наборе α^{i_1} будут только те коэффициенты, которые содержат i_1, т. е. $a_{\{i_1\}\cup\{i_2,\ldots,i_{1+l}\}}, \forall l \in \overline{1,n-1}$.

При $k > 1$ требуется рассмотреть 4 случая:

1) Коэффициент $a_{i_1,i_2,\ldots,i_k}$ не является ведущим на сумме $S_d^{i_1,i_2,\ldots,i_k}$.
2) Коэффициенты вида $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},i_{k+2},\ldots,i_{k+l}\}}, \forall l \in \overline{1,n-k}$, являются ведущими на сумме $S_d^{i_1,i_2,\ldots,i_k}$.
3) Коэффициенты, полученные из $a_{i_1,i_2,\ldots,i_k}$ исключением из индекса коэффициента одного или более чисел $i_1, i_2, \ldots, i_k$, не являются ведущими на сумме $S_d^{i_1,i_2,\ldots,i_k}$.
4) Коэффициенты, полученные из $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},i_{k+2},\ldots,i_{k+l}\}}, \forall l \in \overline{1,n-k}$, исключением из индекса коэффициента одного или более чисел $i_1, i_2, \ldots, i_k$, не являются ведущими на сумме $S_d^{i_1,i_2,\ldots,i_k}$.

В каждом конкретном случае необходимо посчитать количество наборов вида (7) и (8) Леммы 7. Если полученное число будет нечётным, то коэффициент является ведущим на сумме, иначе коэффициент не будет являться ведущим.

1. Рассмотрим коэффициент $a_{i_1,i_2,\ldots,i_k}$. Такой коэффициент является ведущим на единственном наборе вида (8) — наборе α^{i_1}. Этот набор соответствует слагаемому $N(\alpha^{i_1})$ суммы $S_d^{i_1,i_2,\ldots,i_k}$:

$$S_d^{i_1,i_2,\ldots,i_k} = N(\alpha^{i_1}) \oplus N(\alpha^{i_2}) \oplus \ldots \oplus N(\alpha^{i_k}) \oplus N(\alpha^{i_1,i_2}) \oplus N(\alpha^{i_1,i_3}) \oplus \ldots \oplus N(\alpha^{i_{k-1},i_k}) \oplus \ldots \oplus N(\alpha^{i_1,i_2,\ldots,i_{k-1}}) \oplus$$

$$\oplus N(\alpha^{i_1,i_2,\ldots,i_{k-2},i_k}) \oplus \ldots \oplus N(\alpha^{i_2,i_3,\ldots,i_k}) \oplus N(\alpha^{i_1,i_2,\ldots,i_k}).$$

Набором специального вида (7) в этом случае является единственный набор $\alpha^{i_2,i_3,\ldots,i_k}$. Он соответствует слагаемому $N(\alpha^{i_2,i_3,\ldots,i_k})$.

Таким образом, коэффициент $a_{i_1,i_2,\ldots,i_k}$ является ведущим на двух наборах специального вида. Следовательно, по Лемме 7 коэффициент $a_{i_1,i_2,\ldots,i_k}$ не является ведущим на сумме $S_d^{i_1,i_2,\ldots,i_k}$.

2. Теперь рассмотрим коэффициенты вида $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},i_{k+2},\ldots,i_{k+l}\}}, \forall l \in \overline{1,n-k}$. Слагаемым суммы $S_d^{i_1,i_2,\ldots,i_k}$ соответствуют наборы, в которых только переменные $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ могут принимать единичные значения. Следовательно, значения переменных $x_{i_{k+1}}, x_{i_{k+2}}, \ldots, x_{i_{k+l}}$ на любых наборах в сумме $S_d^{i_1,i_2,\ldots,i_k}$ равны 0. Так как $l > 0$, то в любом наборе слагаемых суммы $S_d^{i_1,i_2,\ldots,i_k}$ всегда найдётся хотя бы один ноль среди значений переменных $x_{i_{k+1}}, x_{i_{k+2}}, \ldots, x_{i_{k+l}}$. Но в наборах специального вида (7) среди переменных $x_{i_1}, x_{i_2}, \ldots, x_{i_{k+l}}$ только переменная x_{i_1} должна быть равна 0. Следовательно, наборов вида (7) для рассматриваемых коэффициентов не будет.

К наборам вида (8) будет относиться только один набор α^{i_1}. Следовательно, поскольку набор специального вида всего один, то коэффициенты, имеющие вид $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},i_{k+2},\ldots,i_{k+l}\}}, \forall l \in \overline{1,n-k}$, по Лемме 7 являются ведущими на сумме $S_d^{i_1,i_2,\ldots,i_k}$.

3. В коэффициенте $a_{i_1,i_2,\ldots,i_k}$ исключим из индекса некоторое количество чисел i_1, i_2, ..., i_k. Будем считать, что число i_1 не было исключено. В противном случае разбиение на S_1 и S_2 проведём относительно любой другой переменной x_{i_j}, где i_j — одно из оставшихся в индексе чисел i_2, i_3, ..., i_k.

Из индекса может быть исключено не более $(k-2)$ чисел. Только одно число остаться в индексе не может, так как коэффициенты вида a_{i_1} не являются коэффициентами полинома двойственного остатка N.

Пусть в индексе коэффициента среди чисел i_1, i_2, ..., i_k осталось m-ое количество чисел, включая i_1. К наборам специального вида (7) будут относиться такие наборы, где значение переменной x_{i_1} равно 0 и значения всех прочих $(m-1)$ переменных равны 1, а к наборам вида (8) — наборы, где значение переменной x_{i_1} равно 1 и значения всех прочих $(m-1)$ переменных равны 0. Число наборов каждого из видов (7) и (8) равно количеству способов выбрать значения $(k-m)$ переменных, номера которых были исключены из индекса коэффициента, то есть равно чётному числу

$$2^{k-m}.$$

По Лемме 7 коэффициенты для случая 3 не являются ведущими на сумме $S_d^{i_1,i_2,\ldots,i_k}$.

4. В коэффициентах вида $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},i_{k+2},\ldots,i_{k+l}\}}, \forall l \in \overline{1,n-k}$, исключим из индекса некоторое количество чисел i_1, i_2, ..., i_k. Как и в случае 3, будем считать, что число i_1 не было исключено. Если были исключены все числа i_1, i_2, ..., i_k, то полученный коэффициент не будет являться ведущим ни на одном наборе слагаемых суммы $S_d^{i_1,i_2,\ldots,i_k}$. Следовательно, он не будет ведущим и на самой сумме.

Пусть в индексе коэффициента среди чисел i_1, i_2, ..., i_k осталось m-ое количество чисел, включая i_1. В данном случае к наборам специального вида типа (8) будут относиться такие наборы, в которых значение переменной x_{i_1} равно 1 и значения всех прочих $(m-1)$ переменных равны 0. Их число равно количеству способов выбрать значения $(k-m)$ переменных, номера которых были исключены из индекса коэффициента, то есть равно чётному числу

$$2^{k-m}.$$

Наборов специального вида (7) в данном случае не будет, так как значения $x_{i_{k+1}}$, $x_{i_{k+2}}$, $x_{i_{k+l}}$ на любом наборе суммы $S_d^{i_1,i_2,\ldots,i_k}$ равны 0, а на наборах вида (7) они должны быть равны 1.

Таким образом, коэффициенты, полученные из $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},i_{k+2},\ldots,i_{k+l}\}}, \forall l \in \overline{0,n-k}$, исключением одного или более чисел i_1, i_2, ..., i_k по Лемме 7 не являются ведущими на сумме $S_d^{i_1,i_2,\ldots,i_k}$. □

Используя Теорему 6, преобразуем систему ведущих коэффициентов (6) следующим образом: к каждому элементу системы $N(\alpha^{i_1,i_2,\ldots,i_k})$ мы прибавим (сложением по модулю 2) элементы, входящие в сумму $S_d^{i_1,i_2,\ldots,i_k}$, исключая тот элемент, к которому осуществляется прибавление, так как он уже принадлежит сумме $S_d^{i_1,i_2,\ldots,i_k}$. Таким образом, получим систему вида

$$\begin{cases} S_d^1 = 0, \\ S_d^2 = 0, \\ S_d^{1,2} = 0, \\ \ldots \\ S_d^{2,3,\ldots,(n-1)} = 0, \\ S_d^{1,2,\ldots,(n-1)} = 0. \end{cases} \Leftrightarrow \begin{cases} a_{1,2} \oplus a_{1,3} \oplus \ldots \oplus a_{1,n} \oplus a_{1,2,3} \oplus \ldots \oplus a_{1,3,4,\ldots,n} \oplus a_{1,2,\ldots,n} = 0, \\ a_{1,2} \oplus a_{2,3} \oplus \ldots \oplus a_{2,n} \oplus a_{1,2,3} \oplus \ldots \oplus a_{2,3,\ldots,n} \oplus a_{1,2,\ldots,n} = 0, \\ a_{1,2,3} \oplus a_{1,2,4} \oplus \ldots \oplus a_{1,2,n} \oplus a_{1,2,3,4} \oplus \ldots \oplus a_{1,2,4,5,\ldots,n} \oplus a_{1,2,\ldots,n} = 0, \\ \ldots \\ a_{1,2,\ldots,(n-1)} \oplus a_{2,3,\ldots,n} \oplus a_{1,2,\ldots,n} = 0, \\ a_{1,2,\ldots,n} = 0. \end{cases} \tag{9}$$

Теорема 7. *На сумме $S_r^{i_1,i_2,\ldots,i_k}$ ведущими являются только коэффициенты вида $a_{\{i_1,i_2,\ldots,i_k\}\cup\{i_{k+1},\ldots,i_{k+l}\}}$, $l \in \overline{1,n-k}$, где при $l>1$ для $\forall j \in \overline{k+1,k+l}$ выполнено $i_j \neq n$.*

Доказательство. Запись «$l>1$ для $\forall j \in \overline{k+1,k+l}$ выполнено $i_j \neq n$» означает, что число n может содержаться в индексе коэффициента только при $l=1$.

Доказательство проведём индукцией по k. В качестве базы индукции рассмотрим $k = n - 1$. Сумма $S_r^{i_1,i_2,\ldots,i_{n-1}}$ в этом случае имеет вид

$$S_r^{i_1,i_2,\ldots,i_{n-1}} = S_r^{1,2,\ldots,n-1} = S_d^{1,2,\ldots,n-1}.$$

Коэффициент вида $a_{\{i_1,i_2,\ldots,i_{n-1}\}\cup\{i_n,\ldots,i_{n-1+l}\}}$ всего один: $a_{1,2,\ldots,n}$, при $l = 1$. По Теореме 6 на сумме $S_d^{1,2,\ldots,n-1}$ только этот коэффициент является ведущим.

Пусть для $m \geqslant k$ на сумме $S_r^{i_1,i_2,\ldots,i_m}$ только коэффициенты вида $a_{\{i_1,i_2,\ldots,i_m\}\cup\{i_{m+1},\ldots,i_{m+l}\}}$ являются ведущими, $l \in \overline{1, n-m}$, где при $l > 1$ для $\forall j \in \overline{m+1, m+l}$ выполнено $i_j \neq n$.

Рассмотрим сумму $S_r^{i_1,i_2,\ldots,i_{m-1}}$:

$$S_r^{i_1,i_2,\ldots,i_{m-1}} = S_d^{i_1,i_2,\ldots,i_{m-1}} \oplus \bigoplus_{\substack{1\leqslant i_m<i_{m+1}<\ldots<i_{m-1+l}\leqslant n-1 \\ \{i_m,i_{m+1},\ldots,i_{m-1+l}\}\cap\{i_1,i_2,\ldots,i_{m-1}\}=\varnothing \\ l\in\overline{1,n-m}}} S_r^{\{i_1,i_2,\ldots,i_{m-1}\}\cup\{i_m,\ldots,i_{m-1+l}\}}. \tag{10}$$

По Теореме 6 на сумме $S_d^{i_1,i_2,\ldots,i_{m-1}}$ только коэффициенты вида $a_{\{i_1,i_2,\ldots,i_{m-1}\}\cup\{i_m,i_{m+1},\ldots,i_{m-1+l}\}}$ являются ведущими, для $\forall l \in \overline{1, n-m+1}$. На суммах S_d большего порядка ведущие коэффициенты также должны удовлетворять такому виду, поэтому нет необходимости отдельно рассматривать коэффициенты, не соответствующие описанному в Теореме 6 виду.

Коэффициенты, содержащие число n в своём индексе, $a_{\{i_1,i_2,\ldots,i_{m-1},\ldots,i_{m-1+l}\}\cup\{n\}}$, $l \in \overline{1, n-m}$, по предположению индукции являются ведущими только на суммах вида $S_r^{\{i_1,i_2,\ldots,i_{m-1+l}\}}$ при фиксированном l, причём каждому такому коэффициенту соответствует только одна S_r сумма. Также по Теореме 6 коэффициенты такого вида являются ведущими на сумме $S_d^{i_1,i_2,\ldots,i_{m-1}}$. Итак, среди сумм формулы (10) рассматриваемые коэффициенты будут ведущими только на двух суммах $S_d^{i_1,i_2,\ldots,i_{m-1}}$ и $S_r^{\{i_1,i_2,\ldots,i_{m-1+l}\}}$ (чётное число), то есть не будут ведущими на сумме $S_r^{i_1,i_2,\ldots,i_{m-1}}$.

Коэффициенты, не содержащие число n в своём индексе, $a_{\{i_1,i_2,\ldots,i_{m-1},\ldots,i_{m-1+l}\}}$, $l \in \overline{1, n-m}$, при $l > 1$: $i_j \neq n, j \in \overline{m, m-1+l}$, по Теореме 6 являются ведущими на сумме $S_d^{i_1,i_2,\ldots,i_{m-1}}$. По предположению индукции данные коэффициенты при фиксированном l также являются ведущими на суммах вида $S_r^{\{i_1,i_2,\ldots,i_{m-1+l-p}\}}$ при значениях p от 1 до $(m-2+l)$, то есть на суммах, полученных из $S^{\{i_1,i_2,\ldots,i_{m-1+l}\}}$ исключением p чисел из степени суммы. Число таких сумм S_r равно числу способов исключить p чисел из степени суммы $S^{\{i_1,i_2,\ldots,i_{m-1+l}\}}$. Найдём это число, используя формулу числа сочетаний:

$$C_{m-1+l}^1 + C_{m-1+l}^2 + \ldots + C_{m-1+l}^{m-2+l} = 2^{m-1+l} - 2.$$

Таким образом, общее число коэффициентов вида $a_{\{i_1,i_2,\ldots,i_{m-1},\ldots,i_{m-1+l}\}}$, $l \in \overline{1, n-m}$, где при $l > 1$ для $j \in \overline{m, m-1+l}$ выполнено $i_j \neq n$, равно нечётному числу $2^{m-1+l} - 1$. Следовательно, только такие коэффициенты будут ведущими на сумме $S_r^{i_1,i_2,\ldots,i_{m-1}}$. Шаг индукции доказан. □

С помощью Теоремы 7 преобразуем систему (9), заменив каждую сумму $S_d^{i_1,i_2,\ldots,i_k}$, $k \in \overline{1, n-1}$, соответствующей суммой $S_r^{i_1,i_2,\ldots,i_k}$:

$$\begin{cases} S_r^1 = 0, \\ S_r^2 = 0, \\ S_r^{1,2} = 0, \\ \ldots \\ S_r^{2,3,\ldots,(n-1)} = 0, \\ S_r^{1,2,\ldots,(n-1)} = 0. \end{cases} \Leftrightarrow \begin{cases} a_{1,n} \oplus a_{1,2} \oplus a_{1,3} \oplus \ldots \oplus a_{1,(n-1)} \oplus a_{1,2,3} \oplus \ldots \oplus a_{1,3,4,\ldots,(n-1)} \oplus a_{1,2,\ldots,(n-1)} = 0, \\ a_{2,n} \oplus a_{1,2} \oplus a_{2,3} \oplus \ldots \oplus a_{2,(n-1)} \oplus a_{1,2,3} \oplus \ldots \oplus a_{2,3,\ldots,(n-1)} \oplus a_{1,2,\ldots,(n-1)} = 0, \\ a_{1,2,n} \oplus a_{1,2,3} \oplus a_{1,2,4} \oplus \ldots \oplus a_{1,2,(n-1)} \oplus a_{1,2,3,4} \oplus \ldots \oplus a_{1,2,4,5,\ldots,(n-1)} \oplus \\ \oplus a_{1,2,\ldots,(n-1)} = 0, \\ \ldots \\ a_{2,3,\ldots,n} \oplus a_{1,2,\ldots,(n-1)} = 0, \\ a_{1,2,\ldots,n} = 0. \end{cases} \tag{11}$$

Число всех ведущих коэффициентов равно числу способов формирования конъюнкций из переменных $x_1, x_2, \dots, x_n$:

$$C_n^2 + C_n^3 + \dots + C_n^n = 2^n - C_n^1 - C_n^0 = 2^n - n - 1.$$

Число уравнений системы (11) равно половине всех возможных наборов значений переменных x_1, x_2, ..., x_n без нулевого набора:

$$\frac{2^n}{2} - 1 = 2^{n-1} - 1. \tag{12}$$

Таким образом, решение системы (11) должно иметь $2^{n-1} - 1$ зависимых переменных. Число свободных переменных равно

$$2^n - n - 1 - 2^{n-1} + 1 = 2^{n-1} - n. \tag{13}$$

При преобразовании системы (4) в (5) мы выбрали половину наборов переменных $x_1, x_2, \dots, x_n$, положив переменную x_n равной нулю. Это было сделано для того, чтобы в решении системы (11) зависимыми ведущими коэффициентами, выраженными через свободные коэффициенты, были те, которые в своём индексе содержат число n. Посчитаем количество таких коэффициентов в системе (11). Оно равно количеству способов выбрать числа, содержащиеся в индексе ведущего коэффициента помимо n:

$$C_{n-1}^1 + C_{n-1}^2 + \dots + C_{n-1}^{n-1} = 2^{n-1} - 1.$$

Получили число, равное количеству зависимых переменных (12). Следовательно, система (11) уже приведена к виду, где в каждой её строке записано выражение зависимых ведущих коэффициентов через свободные. Полученные условия на ведущие коэффициенты являются необходимыми и достаточными для того, чтобы полином двойственного остатка N был равен 0 на любых наборах значений переменных $x_1, x_2, \dots, x_n$.

Теорема 8. (*Критерий самодвойственности*) *Булева функция $f(x_1, x_2, \dots, x_n)$ является самодвойственной тогда и только тогда, когда коэффициенты соответствующего ей полинома Жегалкина P удовлетворяют условиям:*

$$\bigoplus_{\substack{1 \leqslant i_1 < \dots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\dots,i_k} = 1,$$

$$a_{1,2,\dots,n} = 0,$$

$$a_{i_1,i_2,\dots,i_k,n} = \bigoplus_{\substack{1 \leqslant i_{k+1} < i_{k+2} < \dots < i_{k+l} \leqslant n-1 \\ \{i_{k+1},i_{k+2},\dots,i_{k+l}\} \cap \{i_1,i_2,\dots,i_k\} = \varnothing \\ l \in \overline{1,n-1-k}}} a_{\{i_1,i_2,\dots,i_k\} \cup \{i_{k+1},\dots,i_{k+l}\}},\ k \in \overline{1,n-2}.$$

Доказательство. По Теореме 2 сохраняющая 0 и 1 или не сохраняющая ни 0, ни 1 функция f является самодвойственной тогда и только тогда, когда двойственный остаток N соответствующего ей полинома P равен 0 для любых наборов переменных x_1, x_2, ..., x_n. Это условие выполняется тогда и только тогда, когда коэффициенты полинома P удовлетворяют системе (11). Выражая из системы (11) коэффициенты, содержащие число n в своём индексе, получаем все условия Теоремы, кроме первого.

Первое условие Теоремы фактически означает нечётность суммы единичных коэффициентов в полиноме Жегалкина P. При этом свободный коэффициент a_0 можно не учитывать. Докажем это.

Самодвойственная функция сохранят 0 и 1 или не сохраняет ни 0, ни 1. По Лемме 2, чтобы функция f сохраняла 1, сумма коэффициентов полинома P должна быть нечётной:

$$a_0 \oplus \bigoplus_{\substack{1 \leqslant i_1 < \dots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\dots,i_k} = 1.$$

Если $a_0 = 0$, то

$$\bigoplus_{\substack{1 \leqslant i_1 < \ldots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\ldots,i_k} = 1.$$

Если $a_0 = 1$, то функция f не сохраняет 0 по Лемме 1. Следовательно, сумма коэффициентов полинома P должна быть чётной:

$$a_0 \oplus \bigoplus_{\substack{1 \leqslant i_1 < \ldots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\ldots,i_k} = 0.$$

Из того что $a_0 = 1$, получаем

$$\bigoplus_{\substack{1 \leqslant i_1 < \ldots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\ldots,i_k} = 1.$$

□

Пример 3. Пусть булева функция $f(x_1, x_2, x_3, x_4)$ представлена в виде полинома Жегалкина P:

$$P(x_1, x_2, x_3, x_4) = 1 \oplus x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4.$$

Определим, является ли функция f самодвойственной.

Свободный коэффициент a_0 равен 1. Количество единичных коэффициентов в P, за исключением свободного коэффициента, равно 5 (нечётное число). Первое условие Теоремы 8 выполнено:

$$\bigoplus_{\substack{1 \leqslant i_1 < \ldots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\ldots,i_k} = 1.$$

По Теореме 8 коэффициенты полинома P также должны удовлетворять следующим условиям

$$a_{1,2,3,4} = 0,$$

$$a_{2,3,4} = a_{1,3,4} = a_{1,2,4} = a_{1,2,3},$$

$$a_{3,4} = a_{1,3} \oplus a_{2,3} \oplus a_{1,2,3},$$

$$a_{2,4} = a_{1,2} \oplus a_{2,3} \oplus a_{1,2,3},$$

$$a_{1,4} = a_{1,2} \oplus a_{1,3} \oplus a_{1,2,3}.$$

В полиноме P коэффициенты равны

$$a_{1,2,3,4} = a_{2,3,4} = a_{1,3,4} = a_{1,2,4} = a_{1,2,3} = a_{1,2} = a_{3,4} = 0,$$

$$a_{1,3} = a_{1,4} = a_{2,3} = a_{2,4} = 1.$$

Как можно видеть, условия Теоремы 8 выполняются. Следовательно, булева функция f является самодвойственной.

Пример 4. Определим, является ли самодвойственной следующая функция $f(x_1, x_2, x_3, x_4, x_5)$, представленная в виде полинома Жегалкина P:

$$P(x_1, x_2, \ldots, x_5) = 1 \oplus x_1 \oplus x_2 \oplus x_5 \oplus x_2x_5 \oplus x_1x_2x_5 \oplus x_3x_4x_5 \oplus x_1x_2x_3x_4 \oplus x_1x_2x_3x_5 \oplus x_1x_2x_3x_4x_5.$$

Старший коэффициент $a_{1,2,3,4,5}$ в полиноме P равен 1. Следовательно, по Теореме 8 функция f не является самодвойственной.

Пример 5. Используя Теорему 8, вычислим количество всех самодвойственных булевых функций от n переменных.

В Теореме 8 число свободных коэффициентов, через которые выражаются зависимые коэффициенты в системе (9), равно $2^{n-1} - n$ (см. формулу (13)). Количество способов выбора различных наборов значений этих коэффициентов равняется

$$2^{2^{n-1}-n}.$$

Кроме определения свободных коэффициентов для формирования самодвойственной функции в виде полинома Жегалкина, необходимо определить значения коэффициентов a_0, a_1, a_2, ..., a_n. По Теореме 8 эти коэффициенты должны иметь такие значения, чтобы общее количество единичных коэффициентов, за исключением свободного коэффициента, было нечётным.

Пусть количество единиц среди нелинейных коэффициентов (индекс содержит более одного числа) нечётно. Тогда число единиц среди линейных коэффициентов (индекс содержит одно число) должно быть чётным. Число способов выбрать единицы среди линейных коэффициентов равно

$$C_n^0 + C_n^2 + C_n^4 + \ldots + C_n^{2\left[\frac{n}{2}\right]} = 2^{n-1}.$$

Если количество единиц среди нелинейных коэффициентов чётно, то количество единиц среди линейных коэффициентов должно быть нечётным. Число способов выбрать единицы среди линейных коэффициентов в этом случае равно

$$C_n^1 + C_n^3 + C_n^5 + \ldots + C_n^{2\left[\frac{n-1}{2}\right]+1} = 2^{n-1}.$$

В обоих случаях число способов выбрать единицы среди линейных коэффициентов получилось одним и тем же.

Свободный коэффициент a_0 можно выбрать двумя способами: 0 или 1.

По правилу умножения общее число способов выбрать коэффициенты полинома Жегалкина таким образом, чтобы соответствующая полиному функция f была самодвойственной, равно

$$2^{2^{n-1}-n} \cdot 2 \cdot 2^{n-1} = 2^{2^{n-1}-n+n} = 2^{2^{n-1}}.$$

5. Общий вид самодостаточного оператора

После обширного раздела, посвящённого самодвойственности в полиномах Жегалкина, мы можем наконец описать общий вид полинома Жегалкина многоместного самодостаточного оператора.

Теорема 9. *Полином Жегалкина $\hat{P}$ является самодостаточным n-местным оператором тогда и только тогда, когда*

1) $a_0 = 1$;
2) $\bigoplus\limits_{\substack{1 \leqslant i_1 < \ldots < i_k \leqslant n \\ k \in \overline{1,n}}} a_{i_1,\ldots,i_k} = 1$;

и когда хотя бы одно из следующих условий не выполняется (*условие несамодвойственности*)

$$a_{1,2,\ldots,n} = 0,$$

$$a_{i_1,i_2,\ldots,i_k,n} = \bigoplus_{\substack{1 \leqslant i_{k+1} < i_{k+2} < \ldots < i_{k+l} \leqslant n-1 \\ \{i_{k+1},i_{k+2},\ldots,i_{k+l}\} \cap \{i_1,i_2,\ldots,i_k\} = \varnothing \\ l \in \overline{1,n-1-k}}} a_{\{i_1,i_2,\ldots,i_k\} \cup \{i_{k+1},\ldots,i_{k+l}\}}, \; k \in \overline{1,n-2}.$$

Доказательство. Условия 1 и 2 следуют из Леммы 3. Чтобы самодостаточный оператор не обладал свойством самодвойственности, не должны выполняться условия Теоремы 8. □

Пример 6. Проверим, является ли данный полином Жегалкина самодостаточным оператором:

$$P(x_1, x_2, \dots, x_5) = 1 \oplus x_1 \oplus x_2 \oplus x_5 \oplus x_2x_5 \oplus x_1x_2x_5 \oplus x_3x_4x_5 \oplus x_1x_2x_3x_4 \oplus x_1x_2x_3x_5 \oplus x_1x_2x_3x_4x_5.$$

Свободный коэффициент a_0 равен 1. Условие 1 Теоремы 9 выполнено.

Количество всех единичных коэффициентов полинома, за исключением a_0, равно 9 — нечётное число. Условие 2 Теоремы 9 выполнено.

Старший коэффициент $a_{1,2,3,4,5}$ равен 1. Следовательно, выполнено условие несамодвойственности Теоремы 9.

По Теореме 9 полином P соответствует самодостаточному 5-местному оператору.

Пример 7. Выясним, будет ли следующий полином Жегалкина самодостаточным оператором:

$$P(x_1, x_2, x_3, x_4) = 1 \oplus x_1 \oplus x_2 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4.$$

Свободный коэффициент a_0 равен 1. Условие 1 Теоремы 9 выполнено.

Сумма единичных коэффициентов полинома без a_0 равна 5 — нечётное число. Следовательно, условие 2 Теоремы 9 выполнено.

Проверим выполнение условия несамодвойственности. Чтобы полином P не обладал самодвойственностью, хотя бы одно из следующих равенств не должно выполняться:

$$a_{1,2,3,4} = 0,$$

$$a_{2,3,4} = a_{1,3,4} = a_{1,2,4} = a_{1,2,3},$$

$$a_{3,4} = a_{1,3} \oplus a_{2,3} \oplus a_{1,2,3},$$

$$a_{2,4} = a_{1,2} \oplus a_{2,3} \oplus a_{1,2,3},$$

$$a_{1,4} = a_{1,2} \oplus a_{1,3} \oplus a_{1,2,3}.$$

В полиноме P старший коэффициент $a_{1,2,3,4}$ равен 0 (в отличие от полинома из Примера 6). Для проверки несамодвойственности полинома придётся рассмотреть остальные равенства.

В полиноме P не рассмотренные ранее коэффициенты равны

$$a_{2,3,4} = a_{1,3,4} = a_{1,2,4} = a_{1,2,3} = a_{1,2} = a_{1,3} = a_{3,4} = 0,$$

$$a_{1,4} = a_{2,3} = a_{2,4} = 1.$$

Равенство, содержащее коэффициент $a_{3,4}$, не выполнено:

$$a_{3,4} \neq a_{1,3} \oplus a_{2,3} \oplus a_{1,2,3}; \quad 0 \neq 0 \oplus 1 \oplus 0; \quad 0 \neq 1.$$

Условие несамодвойственности выполнено. Следовательно, полином P является самодостаточным 4-местным оператором.

Заключение

С помощью коэффициентов полинома Жегалкина могут быть описаны различные свойства булевой функции, которую он задаёт. Например, может быть рассмотрен ряд свойств, касающихся принадлежности функции к предполным классам. В частности, интерес представляет свойство самодвойственности функции применительно к последующему описанию общего вида n-местного самодостаточного оператора.

В данной статье для определения, обладает ли свойством самодвойственности некоторая булева функция, заданная в виде полинома Жегалкина, вводится понятие полинома двойственного остатка. Показано, что сохраняющая 0 и 1 или не сохраняющая ни 0, ни 1 функция f является самодвойственной тогда и только тогда, когда двойственный остаток N соответствующего ей полинома Жегалкина P равен 0 для любых наборов значений переменных x_1, x_2, ..., x_n (Теорема 2). На основании этого условия была получена система ведущих коэффициентов (6).

Система (6) решается вариацией метода Гаусса с использованием прямых S_d и обратных S_r сумм, определение которых было дано в разделе 4.3. Полученное решение этой системы позволило сформулировать критерий самодвойственности булевой функции (Теорема 8). Критерий самодвойственности опирается на значения коэффициентов полинома Жегалкина рассматриваемой функции, описывая для них соответствующие ограничения.

Основным результатом статьи является Теорема 9, формулировка которой содержит условия (накладываемые на коэффициенты), которым должен удовлетворять полином Жегалкина, реализующий n-местный самодостаточный оператор. Доказательство Теоремы 9 базируется на критерии самодвойственности (Теорема 8) и Лемме 3, полученной на основе свойств сохранения полиномом Жегалкина констант 0 и 1.

Отметим, что представление булевой функции в виде полинома Жегалкина является достаточно удобным для описания критериев принадлежности функции к предполным классам. Кроме очевидного определения линейности функций, по виду полинома Жегалкина можно сделать вывод о наличии у функций свойств самодвойственности, а также свойств сохранения констант. Как было показано, полином Жегалкина позволяет определить необходимые и достаточные условия многоместного самодостаточного оператора.

Исходя из этого, можно заключить, что полиномиальное представление булевой функции является полезным при исследовании предполных классов, чем подчёркивается особое значение полиномов Жегалкина.

References

[1] S. V. Yablonskiy, *Introduction into discrete mathematics*, 5th ed. HSE, 2008, 384 pp.

[2] N. M. Martin, *Systems of Logic*. Cambridge University Press, 1989, 318 pp.

[3] R. L. Graham, “On n-valued functionally complete truth functions”, *The Journal of Symbolic Logic*, vol. 32, no. 2, pp. 190–195, 1967.

[4] T. C. Wesselkamper, “A sole sufficient operator”, *NDJFAM*, vol. 16, no. 1, pp. 86–88, 1975.

[5] S. N. Selezneva, “O slozhnosti raspoznavaniya polnoty mnozhestv bulevyh funkcij, realizovannyh polinomami Zhegalkina”, *DMA*, vol. 9, no. 4, pp. 24–31, 1997, in Russian.

[6] S. S. Marchenkov, *Zamknutye klassy bulevyh funkcij*. Fizmatlit, 2000, 128 pp., in Russian.

[7] V. P. Barashev and S. A. Unuchek, *Diskretnaya matematika*. RTU MIREA, 2012, 268 pp., in Russian.

[8] G. P. Gavrilov and A. A. Sapozhenko, *Zadachi i uprazhneniya po diskretnoj matematike*. Fizmatlit, 2005, 416 pp., in Russian.

[9] N. V. Nikonov, “O svyazyah i otlichiyah poluzapretov I, II-go roda i zapretov K-znachnyh funkcij”, *Forestry bulletin*, no. 1, pp. 124–133, 2006, in Russian.

[10] S. S. Marchenkov, *Osnovy teorii bulevyh funkcij*. Fizmatlit, 2014, 136 pp., in Russian.

[11] L. Y. Bystrov and V. S. Rublev, “Bulevy funkcii, ne prinadlezhashchie predpolnym klassam”, in vol. 13, in Russian, Yaroslavl: P.G. Demidov Yaroslavl State University, 2021, pp. 22–26.

[12] A. I. Kostrikin, *Vvedenie v algebpu. Chast’ 1. Osnovy algebpy*. Fizmatlit, 2000, 367 pp., in Russian.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 2, 2023
journal homepage: www.mais-journal.ru

ALGORITHMS

Recursive-Parallel Algorithm for Solving the Maximum Common Subgraph Problem

V. V. Vasilchikov[1]

DOI: 10.18255/1818-1015-2023-2-128-139

[1]P. G. Demidov Yaroslavl State University, 14 Sovetskaya, Yaroslavl 150003, Russia.

MSC2020: 68W10
Research article
Full text in Russian

Received March 22, 2023
After revision May 16, 2023
Accepted May 17, 2023

The paper proposes an algorithm for solving the problem of finding the maximum common subgraph. Both the sequential and the parallel version of the algorithm, their software implementation are described, and an experimental study of their effectiveness is carried out.
This problem is one of the most famous NP-complete problems. Its solution may be required when solving many practical problems related to the study of complex structures. We solve it in a statement when we need to find all possible isomorphisms of the found common subgraph. Due to the extremely high complexity of the problem, it is natural to want to speed up its solution by parallelizing the algorithm. To organize parallel computing, the author used the RPM_ParLib library. It allows to develop effective applications for parallel computing on a local network under the control of the runtime environment .NET Framework.The library supports a recursive-parallel programming style and provides effective work distribution and dynamic load balancing of computational modules during program execution. It can be used for applications written in any programming language supported by the .NET Framework. The purpose of the numerical experiment was to study the acceleration achieved due to the recursive-parallel organization of calculations. For the experiment, the author developed a special application in the C# language designed to generate various sets of initial data with specified parameters. The paper describes the features of the generated initial pairs of graphs, as well as the results obtained during the experiment.

Keywords: maximum common subgraph; isomorphism; parallel algorithm; recursion; .NET

INFORMATION ABOUT THE AUTHORS

Vladimir V. Vasilchikov | orcid.org/0000-0001-7882-8906. E-mail: vvv193@mail.ru
PhD.

Funding: Yaroslavl State University (project VIP-016).

For citation: V. V. Vasilchikov, “Recursive-Parallel Algorithm for Solving the Maximum Common Subgraph Problem”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 128-139, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 2, 2023
сайт журнала: www.mais-journal.ru

Рекурсивно-параллельный алгоритм поиска максимального общего подграфа

В. В. Васильчиков[1]

DOI: 10.18255/1818-1015-2023-2-128-139

[1]Ярославский государственный университет им. П. Г. Демидова, ул. Советская, 14, Ярославль, 150000, Россия.

УДК 519.688: 519.85
Научная статья
Полный текст на русском языке

Получена 22 марта 2023 г.
После доработки 16 мая 2023 г.
Принята к публикации 17 мая 2023 г.

В работе предложен алгоритм решения задачи нахождении максимального общего подграфа. Описаны последовательный и параллельный вариант алгоритма, их программная реализация и произведено экспериментальное исследование их эффективности.
Данная задача является одной из самых известных NP-полных задач. Ее решение может потребоваться при решении многих практических задач, связанных с исследованием сложных структур. Мы решаем ее в постановке, в которой требуется найти все возможные изоморфизмы найденного общего подграфа. Ввиду чрезвычайно высокой трудоемкости задачи желание ускорить ее решение за счет распараллеливания алгоритма является вполне естественным. Для организации параллельных вычислений автором использовалась библиотека RPM_ParLib, которая позволяет создавать параллельные приложения, работающие в локальной вычислительной сети под управлением среды исполнения .NET Framework. Библиотека поддерживает рекурсивно-параллельный стиль программирования и обеспечивает эффективное распределение работы и динамическую балансировку загрузки вычислительных модулей в процессе исполнения программы. Она может быть использована для приложений, написанных на любом языке программирования, поддерживаемом .NET Framework. Целью численного эксперимента было исследование ускорения, достигаемого за счет рекурсивно-параллельной организации вычислений. Для эксперимента автором было разработано специальное приложение на языке C#, предназначенное для генерации различных наборов исходных данных с заданными параметрами. В работе описаны характеристики сгенерированных исходных пар графов, а также результаты, полученные в ходе эксперимента.

Ключевые слова: максимальный общий подграф; изоморфизм; параллельный алгоритм; рекурсия; .NET

ИНФОРМАЦИЯ ОБ АВТОРАХ

Владимир Васильевич Васильчиков | orcid.org/0000-0001-7882-8906. E-mail: vvv193@mail.ru
канд. техн. наук, зав. кафедрой вычислительных и программных систем.

Финансирование: ЯрГУ (проект VIP-016).

Для цитирования: V. V. Vasilchikov, “Recursive-Parallel Algorithm for Solving the Maximum Common Subgraph Problem”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 128-139, 2023.

Введение

Задача о нахождении максимального общего подграфа (MCS — maximum common subgraph) является одной из классических NP-полных задач дискретной оптимизации [1]. Потребность в решении этой задачи возникает в самых разных предметных областях, где требуется установление идентичности структур тех или иных сложных систем. В качестве примеров можно назвать транспортные, энергетические системы, системы связи, электронные схемы, задачи сравнения молекулярных структур, системы распознавания образов, а также задачи математической химии, исследование социальных сетей и многие другие.

Многочисленные авторы, занимавшиеся исследованием данной проблемы, решали ее в самых разных постановках. Например, задача решалась для неориентированных [2] и ориентированных [3] графов. Искомый подграф может быть индуцированным подграфом (MCIS — maximum common induced subgraph), полученным путем удаления узлов, или частичным подграфом (MCPS — maximum common partial subgraph), полученным путем удаления дуг и узлов [3]. В первом случае максимум определяется по числу вершин, во втором — по числу ребер. Для второго варианта также принято использовать аббревиатуру MCES — maximum common edge subgraph. Например в [4] задача рассматривается в обеих этих постановках применительно к неориентированным графам. Также в постановке задачи могут фигурировать дополнительные атрибуты (метки, веса) в зависимости от специфики области применения.

Методы решения задачи также предлагаются самые разные. Например, в [5] предлагаемый алгоритм похож на VF2 [6], а в [7] решение основано на построении так называемого графа ассоциаций и сведения к задаче о клике [8]. Там же в том числе предложены подходы к созданию параллельных алгоритмов, предназначенных для реализации на системах самой разной архитектуры [9].

Следует отметить чрезвычайно высокую вычислительную сложность данной задачи. Она намного выше, чем у других трудоемких задач об изоморфизмах, которыми автор занимался раньше [10, 11].

В первой части автор предлагает алгоритм, пригодный для решения обеих задач (MCIS и MCES) в случае как ориентированных, так и неориентированных графов. Алгоритм построен в стилистике метода ветвей и границ. При решении мы ограничились построением только связного подграфа, максимального по числу вершин либо ребер, а более точно: всех возможных изоморфизмов таких подграфов. Отсутствие упомянутого ограничения могло бы многократно повысить трудоемкость решения.

Далее предлагается параллельный вариант этого алгоритма, основанный на парадигме рекурсивного распараллеливания [12], а также обсуждаются результаты его исследования в виде серии экспериментов с его программной реализацией.

1. Постановка задачи

В упомянутой монографии М. Гэри и Д. Джонсона [1] приводится постановка задачи в следующей формулировке. Пусть есть два графа, заданных своими множествами вершин и дуг: $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$, и положительное целое число K. Существуют ли такие подмножества $E_1^* \subseteq E_1$ и $E_2^* \subseteq E_2$, что $|E_1^*| = |E_2^*| \geq K$, а графы $G_1^* = (V_1, E_1^*)$ и $G_2^* = (V_2, E_2^*)$ изоморфны? В этой формулировке максимум определяется по числу дуг, и, кроме того, отсутствует требование связности искомого подграфа. Впрочем, такая постановка задачи отнюдь не является единственно возможной.

Мы будем ее решать в постановке MCIS, которую сформулируем следующим образом. Пусть есть два графа, заданных своими множествами вершин и дуг: $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$, и положительное целое число K. Существуют ли такие подмножества $V_1^* \subseteq V_1$ и $V_2^* \subseteq V_2$, что $|V_1^*| = |V_2^*| \geq K$, а графы $G_1^* = (V_1^*, E_1^*)$ и $G_2^* = (V_2^*, E_2^*)$ изоморфны? Здесь через E_1^* и E_2^* мы обозначили множества дуг,

связывавших в исходных графах вершины из множеств V_1^* и V_2^* соответственно. Дополнительно мы потребуем связности графов G_1^* и G_2^*.

2. Последовательный алгоритм решения задачи

Основная схема предлагаемого алгоритма одинакова для вариантов MCIS и MCES, для ориентированных и неориентированных графов. Ее основным методом является рекурсивный метод *RecursiveMatch()*, осуществляющий перебор возможных пар (n, m), где вершина n берется из первого графа, а вершина m — из второго. Пока трудно предложить иной подход для получения точного решения этой задачи, кроме полного перебора. В то же время можно придумать целый ряд проверок для оптимизации этого перебора. Собственно, в этом и заключается основная идея метода ветвей и границ.

Для возможно более быстрого решения задачи наш алгоритм должен обладать следующими характеристиками:

- упорядоченный порядок рассмотрения пар, гарантирующий отсутствие повторов;
- максимально быстрое отсечение негодных вариантов;
- минимально возможную трудоемкость вычислений при выборе очередной пары в качестве кандидата на включение в строящуюся подстановку.

Для достижения этой цели очень важен выбор структур данных для хранения информации о текущем состоянии объектов и операций с этими структурами. Мы завели для каждого из графов объект класса *GraphState*, в котором имеются следующие ключевые массивы:

- *CoreByOrder* — содержит номера вершин в порядке их рассмотрения, они не обязательно идут подряд (для первого графа), поскольку при построении ядра (*Core*) нам требуется связность строящегося подграфа;
- *RecNodeStates* — содержит состояния вершин, которое задает либо номер парной вершины из другого графа, либо специальный признак, указывающий на возможность использования вершины на данной ветви вычислений.

Блок-схема метода *RecursiveMatch()* изображена на рис. 1. Метод имеет два параметра: *CoreSize* — размер построенной части подстановки и *Depth* — глубина вложенности рекурсии. Общая схема: метод порождает две ветви: с включением вершины n в строящуюся подстановку и исключением ее из таковой. Далее мы приводим необходимые комментарии к отдельным участкам вычислений.

1. *Проверка текущей ветви вычислений на бесперспективность.* Это основной элемент метода ветвей и границ, позволяющий сразу исключить из рассмотрения ветви, которые не могут улучшить уже достигнутый результат. В нашей задаче критерий для проверки почти очевиден. Для задачи MCIS мы считаем ветвь бесперспективной при выполнении условия $CoreSize + N - Depth < BestResult$, где N — количество вершин в первом графе, *BestResult* — уже достигнутый результат. Для задачи MCES мы могли бы считать ветвь бесперспективной, если оставшихся ребер не хватает для улучшения результата. Как показали эксперименты, это очень слабый критерий, который обычно не позволяет ускорить решение задачи, однако автор пока ничего другого предложить не может. Поэтому далее мы в основном будем обсуждать задачу только в постановке MCIS.
2. *Поиск n_ind — индекса первой подходящей вершины в массиве CoreByOrder для первого графа.* Подходящей вершина считается, если она не включена еще в *Core*, не исключена из рассмотрения на данной ветви и имеет связь с текущим ядром. Далее происходит перестановка номеров вершин в массиве *CoreByOrder*, которая позволяет все рассмотренные вершины держать в массиве подряд. Пусть это вершина n. Здесь же для каждой вершины первого графа вне *Core* происходит пересчет важной числовой характеристики, используемой в пункте 3а для быстрой отбраковки неподходящих для нее пар. Трудоемкость этого пересчета составляет $O(N - Depth)$.

Fig. 1. Block diagram of serial method *RecursiveMatch(CoreSize, Depth)*

Рис. 1. Блок-схема последовательного метода *RecursiveMatch(CoreSize, Depth)*

Fig. 1. Block diagram of the serial method *RecursiveMatch(CoreSize, Depth)*

Рис. 1. Рис. 1. Блок-схема последовательного метода *RecursiveMatch(CoreSize, Depth)*

3. *Начало цикла по m — вершинам в массиве CoreByOrder для второго графа, начиная с индекса CoreSize.* Вершины с меньшими индексами к данному моменту уже включены в *Core*. В начале итерации происходит вычисление числовой характеристики, используемой в пункте 3a. Для вершин второго графа трудоемкость этого вычисления составляет $O(N - |Core|)$.
   (a) *Предварительная проверка пары* (n, m). Упомянутая числовая характеристика предназначена для возможно быстрой отбраковки пар, в которых вершины n и m не годятся для включения в текущее ядро. Мы попробовали около десятка разных вариантов такой характеристики и в итоге остановились на следующем. В случае неориентированных графов мы вычисляем сумму индексов вершин из *Core*, с которыми связана данная вершина (n или m в зависимости от графа). Здесь индекс в *Core* соответствует уровню вложенности рекурсии, на котором вершина была добавлена. В случае ориентированных графов такая характеристика вычисляется отдельно для входящих и выходящих ребер (мы для общности изложения не используем здесь термин «дуга»). Такая проверка здесь требует $O(1)$ операций, а вычисление ее (делается на ветке один раз) имеет трудоемкость $O(|Core|)$.
   (b) *Полная проверка пары* (n, m). Сводится к циклу проверки соответствия ребер двух графов при добавлении пары в ядро, имеет трудоемкость $O(|Core|)$). При положительном результате сохраняем состояние второго графа и запоминаем пару (n, m) в массивах *RecNodeStates* для обоих графов.
   (c) *Если результат лучший (или это повторение лучшего результата), запоминаем его.*
   (d) *Делаем рекурсивный вызов RecursiveMatch(CoreSize+1, Depth+1).* Ветвь «+».
   (e) *Восстанавливаем состояние второго графа.* Трудоемкость этого восстановления составляет $O(N - |Core|)$.
4. *Конец цикла по m.*
5. *Восстанавливаем состояние первого графа.* Трудоемкость восстановления составляет $O(N - Depth)$. Подготавливаем вторую ветвь вычислений, на которой вершина n не включена в подстановку. Пересчитываем состояние первого графа перед началом вычислений на этой ветви, учитывая запрет на включение вершины n в подстановку. Трудоемкость пересчета составляет $O(N - Depth)$.
6. *Выполняем рекурсивный вызов RecursiveMatch(CoreSize, Depth+1).* Ветвь «−».
7. *Восстанавливаем состояние первого графа.* Трудоемкость восстановления можно оценить как $O(N - Depth)$.
8. *Выходим из метода.*

Как можно видеть из сделанных замечаний, на каждом этапе вычислений мы свели трудоемкость вычислений к минимально возможной. Кроме того следует отметить, что в последовательном режиме вычисления на ветви «+» выгоднее выполнять раньше, чем на ветви «−». Так мы быстрее достигнем лучшего результата и следовательно раньше отбросим неперспективные ветви вычислений.

3. Программная реализация последовательного алгоритма

Для отладки и оценки качества предложенного алгоритма на языке C# были написаны две программы.

Первая предназначалась для генерации исходных данных задачи MCS, а именно: двух случайных графов с заданными характеристиками. Она позволяла выстроить пару ориентированных либо неориентированных графов заданного размера с заданной вероятностью дуги или ребра соответственно, а также пару регулярных графов заданного размера с заданной степенью вершины. В случае ориентированных графов под «регулярностью» мы понимали фиксированную сумму количества входящих и исходящих дуг. Кроме того, для отладки на начальном этапе программа

позволяла заложить в эти графы подграф заданного размера, конечно, с точностью до изоморфизма. Впрочем, обычно программа, реализующая предложенный выше алгоритм, находила решение лучшее, чем специально заложенное. На этапе проведения экспериментов возможность задания такого «гарантированного» решения уже не использовалась.

Вторая программа представляла собой собственно реализацию предложенного алгоритма. Она позволяет решать задачу как в постановке MCIS, так и в постановке MCES. Программа находит все лучшие решения в заданной постановке. В ряде случаях таких решений оказывается довольно много, поэтому при выводе в протокол их количество можно ограничить. Также программа собирает и выводит в протокол разнообразную статистику о ходе решения задачи. Мы дополнительно заложили в нее возможность свопинга графов еще до начала вычислений. Причина в том, что в случае, если рассматриваемые графы имеют разное количество вершин, алгоритм, как показали эксперименты, заметно быстрее работает, если в качестве первого выступает граф меньшего размера. Впрочем, в ходе экспериментов в качестве исходных данных мы использовали только графы одинакового размера и с одинаковыми характеристиками (вероятность ребра или степень вершины).

Как и следовало ожидать, трудоемкость данной задачи оказалась намного выше, даже по сравнению с задачей об изоморфизме граф-подграф [11]. Причина в том, что решая задачу MCS, мы не можем заранее знать размер искомого изоморфизма, а значит, вынуждены перебирать все подмножества искомых пар вершин. Количество необходимых вычислений в нашем случае растет с увеличением размера задачи намного быстрее, причем по-разному для упомянутых выше категорий исходных данных.

В процессе тестирования использовались компьютеры на базе двухъядерного процессора Intel Core i3-7100 с тактовой частотой 3.90 GHz и 8 GB оперативной памяти, работающие под управлением 64-разрядной ОС Windows 10. В ходе эксперимента мы ограничились теми исходными данными, которые позволяли на компьютере указанной конфигурации получить решение задачи в постановке MCIS за приемлемое время (не более 40 минут). Поэтому предельными размерами задач для наших экспериментов и как в последовательном, так и в параллельном варианте были следующие:

- для случайных ориентированных графов: 40 вершин, вероятность дуги 0.5;
- для случайных неориентированных графов: 25 вершин, вероятность ребра 0.5;
- для регулярных ориентированных графов: 30 вершин степени 15;
- для регулярных неориентированных графов: 25 вершин степени 10.

Из приведенных значений можно сделать вывод, что для неориентированных графов время работы алгоритма выше, чем для ориентированных, что почти очевидно, поскольку на этапе 3а (предварительная проверка пары (n, m)) отсекается больше вариантов.

Также было отмечено, что количество операций, необходимых для решения задачи, и соответственно время решения задачи для регулярных графов примерно с одинаковыми характеристиками имеет существенно больший разброс, чем для случайных графов. В ходе эксперимента для приемлемых размеров задачи оно могло различаться в 100 и более раз. Про случайные графы такого сказать нельзя, для них расхождение было не более 2–3 раз. Можно также отметить, что решение задачи для регулярных графов в среднем требует больше времени, чем для случайных, что, впрочем, по-видимому, объясняется характером связей внутри графов.

Для метода ветвей и границ ключевым является этап отсечения заведомо бесперспективных ветвей перебора (в нашем алгоритме это первый пункт). Эксперимент позволил оценить эффективность работы данного этапа для предлагаемого алгоритма в обеих постановках (MCIS и MCES). Как было отмечено выше, в случае MCES предложенная оценка оказалась слишком слабой и не позволила сколько-нибудь ускорить вычисления. Вместе с тем прогоны программы в варианте для тех же данных, что и для MCIS, позволили оценить эффективность этого отсечения для последнего,

поскольку в обоих случаях мы проходим по одним и тем же ветвям и можем вычислить, сколько в итоге их было забраковано алгоритмом MCIS. Конечно, размеры решаемых задач в этом случае еще меньше, поскольку трудоемкость для задачи MCES намного выше.

Приведем некоторые результаты таких прогонов. Они достаточно условны, поскольку имеется изрядный разброс статистики для разных исходных данных, однако позволяют сделать некоторую качественную оценку. Отметим также, что эти эксперименты проводились для меньших размеров задачи, чем было указано выше, поскольку при отключенной отбраковке неперспективных ветвей (а в задаче MCES так фактически и было) решение сильно замедляется и это ограничивает размер задачи, которую можно решить за приемлемое время.

Для случайных ориентированных графов (35 вершин, вероятность дуги 0.5) на этапе 1 отбраковывалось 45–50 % ветвей, из оставшихся вариантов на этапе 3а оставался 1 % или даже менее. Ускорение за счет обеих проверок составило примерно 200 раз.

Для случайных неориентированных графов (20 вершин, вероятность ребра 0.5) на этапе 1 отбраковывалось почти 90 % ветвей, из оставшихся вариантов на этапе 3а оставалось 5–6 %. Ускорение за счет обеих проверок составило более 150 раз.

Для регулярных ориентированных графов (28 вершин, степень вершины 14) на этапе 1 отбраковывалось примерно 45 % ветвей, из оставшихся вариантов на этапе 3а оставалось 1.5–2 %. Ускорение за счет обеих проверок составило примерно 135 раз.

Для регулярных неориентированных графов (20 вершин, степень вершины 10) на этапе 1 отбраковывалось примерно более 90 % ветвей, из оставшихся вариантов на этапе 3а оставалось 5–6 %. Ускорение за счет обеих проверок составило примерно 200 раз.

Следует отметить, что такие эксперименты мы проводили и для других (меньших) размеров задач. Они позволили сделать вывод, что с увеличением размера задачи результативность примененных проверок ощутимо возрастает, и этот факт наглядно демонстрирует эффективность применения метода ветвей и границ с предложенной выше оценкой перспективности для решения задачи в постановке MCIS.

4. Параллельный алгоритм решения задачи

Метод рекурсивного распараллеливания наиболее эффективно работает в тех случаях, когда решаемую задачу (или подзадачу) можно разделить на две подзадачи с равным или хотя бы сопоставимым объемом вычислений. Вариант с равным объемом является идеальным и, как любой идеальный вариант, в жизни не встречается. Поэтому в программные средства поддержки РП программирования [13] были заложены возможности динамической балансировки загрузки. Но если объемы двух порождаемых подзадач различаются очень сильно, от программиста требуется выстроить процесс порождения параллельных ветвей несколько иначе, чем в базовом алгоритме [12].

Предложенный выше последовательный алгоритм решения задачи распадается на две ветви («+» и «−»), объем которых отличается очень существенно. Более того, ветвь «+» представляет собой цикл с рекурсивными вызовами на отдельных итерациях. К такой схеме нас привело желание исключить повторное вычисление значений, образующих текущее состояние характеристик первого графа. Выстраивая параллельный алгоритм, представляется разумным рекурсивные вызовы ветви «−» на отдельных итерациях оставить для выполнения процессорному модулю (ПМ), породившему эту ветвь (через *H_Call()*).Тогда выполнение вычислений на ветви «+» было бы естественно оформить их через вызов *P_Call()*.

Однако правильной последовательностью оформления вызовов параллельных активаций является такая: *P_Call(), H_Call(), Wait()*. Поэтому в отличие от базовой версии алгоритма, где по указанным выше причинам ветвь «+» шла первой, мы изменили порядок следования ветвей. В результате алгоритм приобрел вид, представленный на рис. 2.

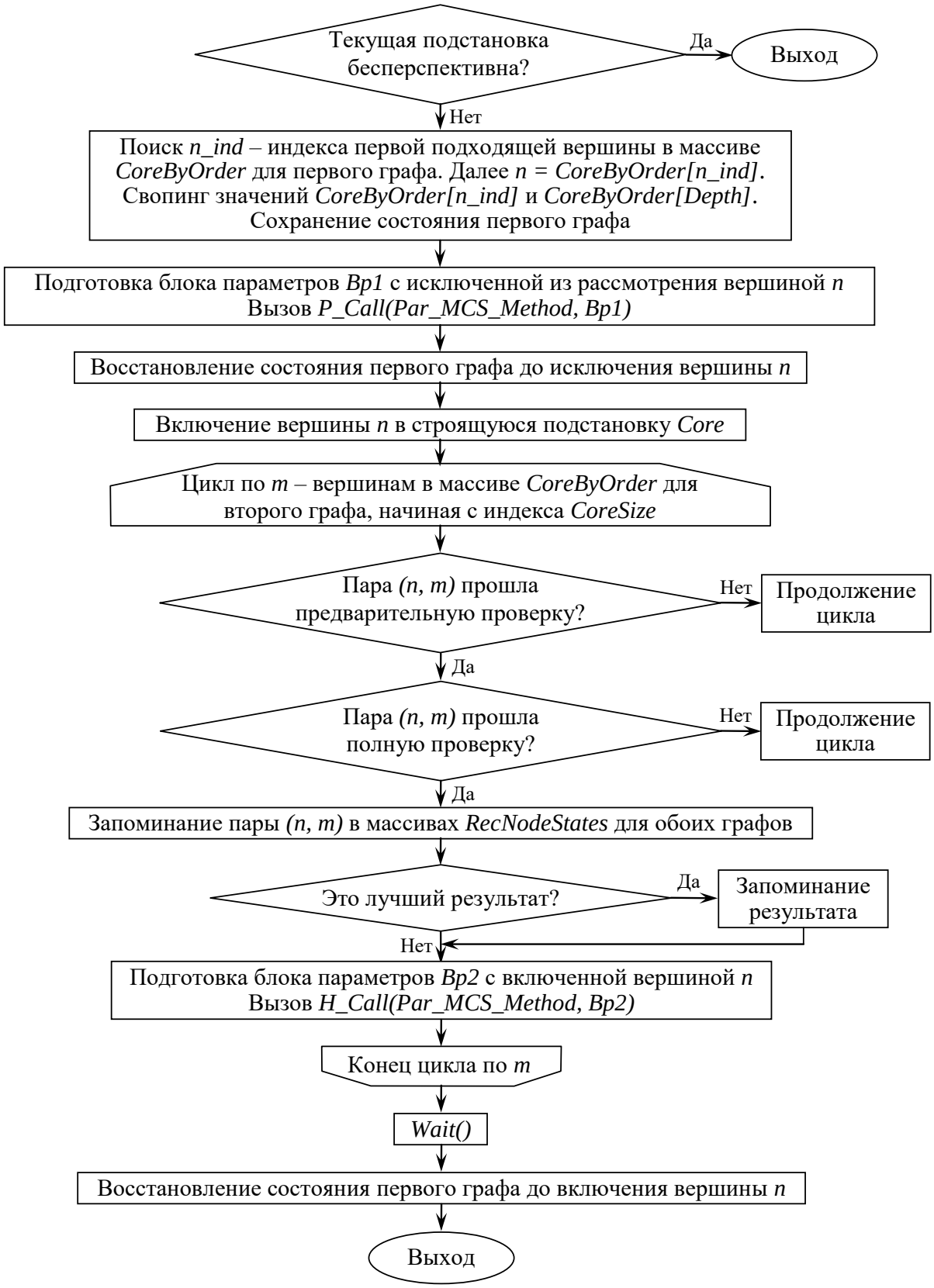


Fig. 2. Block diagram of the parallel method *RecursiveMatch(actState)*

Рис. 2. Блок-схема параллельного варианта метода *RecursiveMatch(actState)*

Table 1. The Acceleration of the RP-algorithm

Таблица 1. Ускорение РП-алгоритма

Source Data		Sequently	1 PM	2 PM	4 PM	6 PM	8 PM	12 PM	16 PM
Example 1 Branches (mln): ***518.65***	Exec. Time (s)	1827.16	1507.80	772.53	608.41	412.29	371.10	316.54	293.55
	Ratio to Seq		***1.21***	***2.37***	***3.00***	***4.43***	***4.92***	***5.77***	***6.22***
	Ratio to 1 PM		**1.00**	**1.95**	**2.48**	**3.66**	**4.06**	**4.76**	**5.14**
Example 2 Branches (mln): ***463.26***	Exec. Time (s)	1647.64	1369.98	833.04	558.49	355.37	306.41	289.03	275.55
	Ratio to Seq		***1.20***	***1.98***	***2.95***	***4.64***	***5.38***	***5.70***	***5.98***
	Ratio to 1 PM		**1.00**	**1.64**	**2.45**	**3.86**	**4.47**	**4.74**	**4.97**
Example 3 Branches (mln): ***462.06***	Exec. Time (s)	1653.02	1379.50	815.17	573.53	352.70	351.02	327.41	261.34
	Ratio to Seq		***1.20***	***2.03***	***2.88***	***4.69***	***4.71***	***5.05***	***6.33***
	Ratio to 1 PM		**1.00**	**1.69**	**2.41**	**3.91**	**3.93**	**4.21**	**5.28**
Example 4 Branches (mln): ***888.37***	Exec. Time (s)	2399.06	1769.01	789.57	566.08	423.37	357.85	284.71	280.72
	Ratio to Seq		***1.36***	***3.04***	***4.24***	***5.67***	***6.70***	***8.43***	***8.55***
	Ratio to 1 PM		**1.00**	**2.24**	**3.13**	**4.18**	**4.94**	**6.21**	**6.30**
Example 5 Branches (mln): ***884.58***	Exec. Time (s)	2391.80	1763.76	787.03	632.88	425.31	357.83	327.41	322.73
	Ratio to Seq		***1.36***	***3.04***	***3.78***	***5.62***	***6.68***	***7.31***	***7.41***
	Ratio to 1 PM		**1.00**	**2.24**	**2.79**	**4.15**	**4.93**	**5.39**	**5.47**
Example 6 Branches (mln): ***712.73***	Exec. Time (s)	1937.06	1221.53	743.62	450.25	335.93	304.79	267.13	247.84
	Ratio to Seq		***1.59***	***2.60***	***4.30***	***5.77***	***6.36***	***7.25***	***7.82***
	Ratio to 1 PM		**1.00**	**1.64**	**2.71**	**3.64**	**4.01**	**4.57**	**4.93**
Example 7 Branches (mln): ***598.41***	Exec. Time (s)	1931.40	1463.80	661.91	416.18	373.29	293.43	221.17	212.74
	Ratio to Seq		***1.32***	***2.92***	***4.64***	***5.17***	***6.58***	***8.73***	***9.08***
	Ratio to 1 PM		**1.00**	**2.21**	**3.52**	**3.92**	**4.99**	**6.62**	**6.88**
Example 8 Branches(mln): ***818.04***	Exec. Time (s)	2698.58	1995.70	860.23	503.60	378.03	375.82	322.94	285.84
	Ratio to Seq		***1.35***	***3.14***	***5.36***	***7.14***	***7.18***	***8.36***	***9.44***
	Ratio to 1 PM		**1.00**	**2.32**	**3.96**	**5.28**	**5.31**	**6.18**	**6.98**
Example 9 Branches (mln): ***1036.09***	Exec. Time (s)	3412.75	2668.96	1090.94	598.74	475.16	423.73	398.26	350.27
	Ratio to Seq		***1.28***	***3.13***	***5.70***	***7.18***	***8.05***	***8.57***	***9.74***
	Ratio to 1 PM		**1.00**	**2.45**	**4.46**	**5.62**	**6.30**	**6.70**	**7.62**
Example 10 Branches (mln): ***836.57***	Exec. Time (s)	2298.15	1662.66	768.89	658.51	446.11	370.32	353.07	285.87
	Ratio to Seq		***1.38***	***2.99***	***3.49***	***5.15***	***6.21***	***6.51***	***8.04***
	Ratio to 1 PM		**1.00**	**2.16**	**2.52**	**3.73**	**4.49**	**4.71**	**5.82**
Example 11 Branches (mln): ***702.42***	Exec. Time (s)	1976.47	1480.91	667.13	470.35	355.83	289.67	226.97	222.30
	Ratio to Seq		***1.33***	***2.96***	***4.20***	***5.55***	***6.82***	***8.71***	***8.89***
	Ratio to 1 PM		**1.00**	**2.22**	**3.15**	**4.16**	**5.11**	**6.52**	**6.66**
Example 12 Branches (mln): ***724.37***	Exec. Time (s)	2026.67	1467.65	680.11	474.15	332.51	317.74	260.46	224.02
	Ratio to Seq		***1.38***	***2.98***	***4.27***	***6.09***	***6.38***	***7.78***	***9.05***
	Ratio to 1 PM		**1.00**	**2.16**	**3.10**	**4.41**	**4.62**	**5.63**	**6.55**

Передача накопленной информации о текущем состоянии строящейся подстановки осуществляется через специальную структуру, называющуюся блоком параметров. Она содержит список уже включенных в построенную часть подстановки (*Core*) вершин в порядке их включения, а также информацию о связях каждой такой вершины с вершинами из *Core*. Эта информация используется для предварительной проверки очередной пары кандидатов на включение в *Core*. Решение о том, какую версию основного метода (последовательную или параллельную) запускать на очередном этапе вычислений, принимается на основании двух параметров: предельной глубины вложенности и предельного размера построенной части подстановки.

5. Программная реализация параллельного алгоритма и результаты экспериментов

Целью эксперимента была оценка ускорения решения задачи, достигаемого за счет использования предложенного параллельного алгоритма. В качестве исходных данных мы брали сгенерированные случайным образом пары графов с различными характеристиками. Они перечислены в разделе, посвященном описанию экспериментального исследования последовательного алгоритма. Там же указаны вычислительные характеристики компьютеров, использованных для проведения эксперимента. Пропускная способность сети равнялась 100 Mb/s.

Собранные в таблице 1 результаты демонстрируют довольно хорошую эффективность предложенного рекурсивно-параллельного алгоритма. Результаты получены на 12 парах случайных графов с различными характеристиками, заложенными при их построении, а именно:

- примеры 1–3: случайные ориентированные графы из 40 вершин, вероятность дуги 0.5;
- примеры 4–6: случайные неориентированные графы из 25 вершин, вероятность ребра 0.5;
- примеры 7–9: регулярные ориентированные графы из 30 вершин степени 15;
- примеры 10–12: регулярные неориентированные графы из 25 вершин степени 10.

Отметим, что наличие ускорения, достигаемого параллельной программой на одном компьютере, по сравнению с последовательной ее версией, объясняется использованием многопоточности.

References

[1] M. R. Garey and D. S. Johnson, “Computers and intractability: A guide to the theory of NP-completeness”, *Siam Review*, vol. 24, no. 1, pp. 90–91, 1982.

[2] R. Hoffmann, C. McCreesh, and C. Reilly, “Between subgraph isomorphism and maximum common subgraph”, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017, pp. 3907–3914.

[3] S. N. Ndiaye and C. Solnon, “CP models for maximum common subgraph problems”, *Lecture Notes in Computer Science*, vol. 6876, pp. 637–644, 2011.

[4] D. Conte, P. Foggia, and M. Vento, “Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs”, *Journal of Graph Algorithms and Applications*, vol. 11, no. 1, pp. 99–143, 2007.

[5] J. J. McGregor, “Backtrack search algorithms and the maximal common sub-graph problem”, *Software: Practice and Experience*, vol. 12, no. 1, pp. 23–34, 1982.

[6] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, “An improved algorithm for matching large graphs”, in *Proc. of the 3rd IAPR-TC-15 InternationalWorkshop on Graph-based Representations*, Citeseer, 2001, pp. 149–159.

[7] P. J. Durand, R. Pasari, J. W. Baker, and C.-c. Tsai, “An efficient algorithm for similarity analysis of molecules”, *Internet Journal of Chemistry*, vol. 2, no. 17, pp. 1–16, 1999.

[8] C. Bron and J. Kerbosch, “Algorithm 457: Finding all cliques of an undirected graph”, *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[9] A. Marcelli, S. Quer, and G. Squillero, *The maximum common subgraph problem: A portfolio approach*, 2019. arXiv: 1908.06418 [cs.DS]. [Online]. Available: http://arxiv.org/abs/1908.06418.

[10] V. V. Vasilchikov, “Parallel algorithm for solving the graph isomorphism problem”, *Modeling and analysis of information systems*, vol. 27, no. 1, pp. 86–94, 2020.

[11] V. V. Vasilchikov, “Recursive-parallel algorithm for solving the graph-subgraph isomorphism problem”, *Modeling and analysis of information systems*, vol. 29, no. 1, pp. 30–43, 2022.

[12] V. V. Vasilchikov, *Sredstva parallelnogo programmirovaniya dlya vychislitelnykh sistem s dinamicheskoy balansirovkoy zagruzki.* YarGU, Yaroslavl, 2001, in Russian.

[13] V. V. Vasilchikov, "On the recursive-parallel programming for the .NET framework", *Automatic Control and Computer Sciences*, vol. 48, pp. 575–580, 2014.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 2, 2023
journal homepage: www.mais-journal.ru

ALGORITHMS

On Simplifying Expressions with Mixed Boolean-Arithmetic

Y. V. Kosolapov[1]

DOI: 10.18255/1818-1015-2023-2-140-159

[1]Southern Federal University, 105/42 Bolshaya Sadovaya str., Rostov-on-Don, 344006, Russia.

MSC2020: 93B11
Research article
Full text in Russian

Received April 3, 2023
After revision May 17, 2023
Accepted May 17, 2023

Mixed Boolean-Arithmetic expressions (MBA-expressions) with t integer n-bit variables are often used for program obfuscations. Obfuscation consists of replacing short expressions with longer equivalent expressions that seem to take the analyst more time to explore. The paper shows that to simplify linear MBA-expressions (reduce the number of terms), a technique similar to the technique of decoding linear codes by information sets can be applied. Based on this technique, algorithms for simplifying linear MBA-expressions are constructed: an algorithm for finding an expression of minimum length and an algorithm for reducing the length of an expression. Based on the length reduction algorithm, an algorithm is constructed that allows to estimate the resistance of an MBA-expression to simplification. We experimentally estimate the dependence of the average number of terms in a linear MBA-expression returned by simplification algorithms on n, the number of decoding iterations, and the power of the set of Boolean functions, by which a linear combination with a minimum number of nonzero coefficients is sought. The results of the experiments for all considered t and n show that if before obfuscation the linear MBA-expression contained $r = 1, 2, 3$ terms, then the developed simplification algorithms with a probability close to one allow using the obfuscated version of this expression find an equivalent one with no more than r terms. This is the main difference between the information set decoding technique and the well-known techniques for simplifying linear MBA-expressions, where the goal is to reduce the number of terms to no more than 2^t. We also found that for randomly generated linear MBA-expressions with increasing n, the average number of terms in the returned expression tends to 2^t and does not differ from the average number of terms in the linear expression returned by known simplification algorithms. The results obtained, in particular, make it possible to determine t and n for which the number of terms in the simplified linear MBA-expression on average will not be less than the given one.

Keywords: code obfuscation; MBA-expressions; simplification of MBA-expressions; decoding by information sets

INFORMATION ABOUT THE AUTHORS

Yury V. Kosolapov | orcid.org/0000-0002-1491-524X. E-mail: itaim@mail.ru
Associated professor.

For citation: Y. V. Kosolapov, "On Simplifying Expressions with Mixed Boolean-Arithmetic", *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 140-159, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 2, 2023
сайт журнала: www.mais-journal.ru

ALGORITHMS

Об упрощении выражений со смешанной битовой и целочисленной арифметикой

Ю. В. Косолапов[1]

DOI: 10.18255/1818-1015-2023-2-140-159

[1]Южный федеральный университет, ул. Большая Садовая, 105/42, Ростов-на-Дону, 344006, Россия.

УДК 004.056.5
Научная статья
Полный текст на русском языке

Получена 3 апреля 2023 г.
После доработки 17 мая 2023 г.
Принята к публикации 17 мая 2023 г.

Выражения со смешанной булевой и целочисленной арифметикой (далее — MBA-выражения, от англ. Mixed Boolean-Arithmetic) от t целочисленных n-битных переменных часто находят применение при обфускации (запутывании) программного кода. Запутывание заключается в замене коротких выражений более длинными эквивалентными выражениями, на исследование которых, как представляется, аналитиком может быть затрачено больше времени. В работе показано, что для упрощения линейных MBA-выражений (сокращения количества слагаемых) может быть применена техника, аналогичная технике декодирования линейных кодов по информационным совокупностям. На основе этой техники в работе построены алгоритмы упрощения линейных MBA-выражений: алгоритм нахождения выражения с минимальным числом слагаемых и алгоритм сокращения числа слагаемых. На основе алгоритма сокращения числа слагаемых построен алгоритм, позволяющий оценить стойкость MBA-выражения к упрощению. В работе экспериментально оценена зависимость среднего числа слагаемых в линейном MBA-выражении, возвращаемом алгоритмами упрощения, от разрядности n, числа итераций декодирования и мощности набора булевых функций, по которому ищется линейная комбинация с минимальным числом ненулевых коэффициентов. Результаты экспериментов для всех рассмотренных t и n показывают, что если до обфускации линейное MBA-выражение содержало $r = 1, 2, 3$ слагаемых, то разработанные алгоритмы упрощения с вероятностью, близкой к единице, позволяют по обфусцированному варианту этого выражения найти эквивалентное с числом слагаемых не более r. В этом заключается главное отличие техники декодирования по информационным совокупностям от известных техник упрощения линейных MBA-выражений, в которых целью является сокращение числа слагаемых до не более чем 2^t. В работе также установлено, что для случайно сгенерированных линейных MBA-выражений с ростом n среднее число слагаемых в возвращаемом выражении стремится к 2^t и не отличается от среднего числа слагаемых в линейном выражении, возвращаемом известными алгоритмами упрощения. Полученные результаты, в частности, позволяют определить t и n, для которых количество слагаемых в упрощенном линейном MBA-выражении в среднем будет не менее заданного.

Ключевые слова: обфускация программного кода; MBA-выражения; упрощение MBA-выражений; декодирование по информационным совокупностям

ИНФОРМАЦИЯ ОБ АВТОРАХ

Юрий Владимирович Косолапов | orcid.org/0000-0002-1491-524X. E-mail: itaim@mail.ru
доцент, канд. техн. наук.

Для цитирования: Y. V. Kosolapov, “On Simplifying Expressions with Mixed Boolean-Arithmetic”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 140-159, 2023.

Введение

Обфускация (или запутывание) программного кода — это преобразование, сохраняющее функциональность программы, с целью затруднения исследования кода аналитиком (человеком и/или автоматическими средствами). Несмотря на то, что не существует универсального обфусцирующего преобразования, стойкого в рамках модели «виртуального черного ящика» [1], исследования в области построения и анализа практически стойких обфусцирующих преобразований продолжаются. Одним из используемых на практике подходов является замена коротких арифметических и битовых выражений более длинными выражениями, содержащими как битовые операции над целыми числами (в том числе, битовые операции $\wedge$, $\vee$, $\oplus$ над целыми числами), так и арифметические (операции + и $\times$) [2]. Далее такие выражения будем называть MBA-выражениями (аббревиатура от английского названия Mixed Boolean-Arithmetic). Эвристическое обоснование запутывающих свойств такого подхода строится, с одной стороны, на отсутствии математических правил упрощения MBA-выражений, а с другой стороны, на том, что более длинные программы менее читабельны, чем короткие [3]. В области защиты программ от исследования MBA-выражения часто применяются для построения *непрозрачных (неявных) предикатов* (*opaque predicates*) [4]. Напомним, что непрозрачным предикатом $P(x_1, \dots, x_t)$ от t переменных называется логическая функция, *тождественно* равная нулю или единице (константный предикат). Обычно переменные $x_1, \dots, x_t$ являются целочисленными. Когда значение непрозрачного предиката известно разработчику, то этот предикат может использоваться для добавления, например, условного оператора, в котором всегда будет выполняться только одна из альтернатив, известная разработчику. Непрозрачный предикат обычно строится так, чтобы по его виду было вычислительно трудно распознать в нем константный предикат [5]. В этом случае аналитику потребуется затратить ресурсы (вычислительные и временные) на анализ обеих альтернатив. В конструкции условного оператора, изображенного на рис. 1, тождественно истинный непрозрачный предикат $P_T(x_1, \dots, x_t)$ используется для добавления недостижимого кода, на анализ и понимание которого аналитиком, как представляется, будут затрачены ресурсы.

```
if P_T(x_1, ..., x_t) = 1 then
  /* Всегда выполняемый код */
else
  /* Недостижимый код */
```

Fig. 1. Using an identically true opaque predicate

Рис. 1. Использование тождественно истинного непрозрачного предиката

Как отмечено в [5], в свободно распространяемых и коммерческих средствах обфускации используется сравнительно небольшой набор непрозрачных предикатов. Поэтому они могут быть распознаны аналитиком путем сравнения исследуемого предиката из программы с предикатами из словаря. Известно, например, что некоторые оптимизаторы могут распознавать непрозрачные предикаты и удалять недостижимый код. В частности, в [6] утверждается, что многие непрозрачные предикаты не устойчивы к оптимизации, применяемой в компиляторе clang. Для расширения разнообразия предикатов, устойчивых к распознаванию по словарю и/или оптимизации, могут использоваться MBA-выражения. Например, истинность предикатов $P(x_1, \dots, x_t)$ и $P(x_1, \dots, x_t) + E(x_1, \dots, x_t)$ совпадает, если $E(x_1, \dots, x_t)$ — тождественно равное нулю MBA-выражение. MBA-выражения и сами могут использоваться в роли непрозрачных предикатов. Например, если $E_1(x_1, \dots, x_t)$ и $E_2(x_1, \dots, x_t)$ — разные тождественно равные нулю MBA-выражения, то с помощью них, например, следующим образом могут быть построены тождественно равный нулю предикат $P_F(x_1, \dots, x_n)$ и тождественно равный единице предикат $P_T(x_1, \dots, x_n)$:

$$P_F(x_1, \dots, x_t) = (E_1(x_1, \dots, x_t) \neq E_2(x_1, \dots, x_t)),$$
$$P_T(x_1, \dots, x_t) = (E_1(x_1, \dots, x_t) \geqslant E_2(x_1, \dots, x_t)).$$

С появлением обфускации на основе MBA-выражений возникла и задача деобфускации таких выражений. Целью деобфускации MBA-выражений обычно является получение по выражению $E(x_1, \dots, x_t)$ эквивалентного ему MBA-выражения $E'(x_1, \dots, x_t)$, в котором количество переходов от арифметических вычислений к битовым (или наоборот) как можно меньше. Такие переходы в [7] названы *чередованиями*. Сокращение числа чередований в MBA-выражениях в рамках задачи деобфускации мотивировано следующим фактом. При деобфускации непрозрачных предикатов часто применяются SMT-решатели (аббревиатура от английского термина Satisfiability Modulo Theories), которые могут проверить, например, тождественную истинность или тождественную ложность выражения. Результаты сравнения трех известных SMT-решателей, проведенного в [8], показали, что чем больше чередований в MBA-выражении, на основе которого построен непрозрачный предикат, тем больше времени требуется SMT-решателю для получения ответа. В частности, при десяти чередованиях время решения одного предиката на основе MBA-выражения достигало 3 000 секунд. Эти результаты также подтверждаются результатами исследования [6], где заключается, что для деобфускации MBA-выражений недостаточно только SMT-решателей. Другими словами, перед применением SMT-решателя следует упрощать MBA-выражение $E(x_1, \dots, x_t)$, так как нахождение эквивалентного ему MBA-выражения $E'(x_1, \dots, x_t)$ с меньшим числом чередований может сократить время работы SMT-решателя. Наибольшего успеха исследователи достигли при деобфускации линейных MBA-выражений (линейная комбинация битовых подвыражений) [4, 8, 9]. Отметим, что эти результаты легли в основу плагина goo MBA для дизассемблера IDA Pro.

В настоящей работе будет показано, как линейные MBA-выражения могут быть упрощены с использованием техники, применяемой при декодировании линейных кодов и дешифровании кодовых криптосистем. В ряде случае эта техника позволяет получить линейные MBA-выражения с меньшим числом чередований, чем при использовании известных техник деобфускации линейных MBA-выражений. Работа, помимо введения и заключения, содержит два раздела. В первом разделе приводится определение MBA-выражения и классификация таких выражений. Здесь же сформулирован и для полноты изложения доказан основной результат из [2], позволяющий построить множество тождественно равных нулю линейных MBA-выражений. В терминах из линейной алгебры дается определение эквивалентных линейных MBA-выражений и строятся рандомизированные алгоритмы генерации линейного MBA-выражения, эквивалентного заданному линейному MBA-выражению. Второй раздел посвящен построению и экспериментальному исследованию новых алгоритмов упрощения линейных MBA-выражений. В этом разделе приведены необходимые сведения о линейных кодах и показано, как связаны задачи декодирования линейных кодов и деобфускации линейных MBA-выражений.

1. Об обфускации MBA-выражениями

Пусть $\mathbb{R}$ — числовое кольцо. Носителем вектора $\mathbf{x} = (x_1, \dots, x_k) \in \mathbb{R}^k$ назовем множество $\operatorname{supp}(\mathbf{x}) = \{i : x_i \neq 0\}$, а весом Хэмминга $\operatorname{wt}(x)$ этого вектора — мощность его носителя: $\operatorname{wt}(x) = |\operatorname{supp}(x)|$.

1.1. MBA-выражения

Пусть $n \in \mathbb{N}$, $\mathbb{V} = \mathbb{Z}_{2^n}$. Для $B = \{0, 1\}$ и $t \in \mathbb{N}$ через $\beta : \mathbb{V} \longrightarrow B^n$ обозначим отображение, ставящее в соответствие целому числу из $\mathbb{V}$ его двоичную запись из B^n, причем для $\mathbf{a} = (a_{n-1}, \dots, a_0) \in B^n$ обратное отображение β^{-1} определим так: $\beta^{-1}(\mathbf{a}) = \sum_{i=0}^{n-1} 2^i a_i$. Под булевой функцией $f(\alpha_1, \dots, \alpha_t)$ от t переменных будем понимать произвольное отображение $f : B^t \longrightarrow B$. Тогда *битовым* выражением от t целочисленных n-битных переменных $x_1, \dots, x_t$, соответствующим булевой функции f,

называется отображение $e_f : \mathbb{V}^t \to \mathbb{V}$ вида

$$e_f(x_1, \dots, x_t) = \beta^{-1}\Big(f(\beta(x_1)_{n-1}, \dots, \beta(x_t)_{n-1}), \dots, f(\beta(x_1)_0, \dots, \beta(x_t)_0)\Big)$$
$$= \sum_{i=0}^{n-1} 2^i f(\beta(x_1)_i, \dots, \beta(x_t)_i), \tag{1}$$

где $\beta(x_j)_i$ — i-ая координата n-битного вектора $\beta(x_j)$. Из (1), в частности, вытекает, что булевой функции $f(\alpha_1, \dots, \alpha_t) \equiv 1$ соответствует битовое выражение $e_f(x_1, \dots, x_t) = \sum_{i=0}^{n-1} 2^i = 2^n - 1 = -1 \in \mathbb{V}$.

Обозначим через $\mathcal{F}_t$ множество всех булевых функций от t переменных. В соответствии с [4], *полиномиальным* MBA-выражением называется функция $E : \mathbb{V}^t \to \mathbb{V}$ от t целочисленных n-битных переменных $x_1, \dots, x_t \in \mathbb{V}$, имеющая вид

$$E(x_1, \dots, x_t) = \sum_{i \in I} a_i \prod_{f \in J_i} e_f(x_1, \dots, x_t), \tag{2}$$

где $I \subset \mathbb{N}$, $J_i \subset \mathcal{F}_t$, $a_i \in \mathbb{V} \setminus \{0\}$, $e_f(x_1, \dots, x_t)$ — битовое выражение. Операции сложения и умножения в (2) выполняются в кольце $\mathbb{V} = \mathbb{Z}_{2^n}$. Полиномиальное MBA-выражение называется *линейным*, если $|J_i| = 1$ для всех $i \in I$ в (2). Далее исследуются только линейные MBA-выражения, поэтому вместо представления (2) для таких выражений будем использовать представление

$$E(x_1, \dots, x_t) = \sum_{f \in J \subset \mathcal{F}_t} a_f e_f(x_1, \dots, x_t), \tag{3}$$

где $a_f \in \mathbb{V} \setminus \{0\}$. Множество J в (3) будем называть носителем выражения $E(x_1, \dots, x_t)$ и обозначать $\mathrm{supp}(E)$. Заметим, что количество чередований в линейном MBA-выражении, в соответствии с определением из [7], равно $|J|$. Для удобства $|J|$ будем называть *длиной* линейного MBA-выражения.

Для $n = 8$ и $t = 3$ примером полиномиального MBA-выражения является

$$E(x_1, x_2, x_3) = 7(x_1 \wedge x_2)(x_2 \vee x_3) - (x_1 \oplus x_2 \oplus x_3) + 120,$$

где $I = \{1, 2, 3\}$, $a_1 = 7$, $a_2 = -1$, $a_3 = -120$, $J_1 = \{f_1(\alpha_1, \alpha_2, \alpha_3) = \alpha_1 \wedge \alpha_2, f_2(\alpha_1, \alpha_2, \alpha_3) = \alpha_2 \vee \alpha_3\}$, $J_2 = \{f(\alpha_1, \alpha_2, \alpha_3) = \alpha_1 \oplus \alpha_2 \oplus \alpha_3\}$, $J_3 = \{f(\alpha_1, \alpha_2, \alpha_3) \equiv 1\}$. Для тех же n и t выражение вида

$$E(x_1, x_2, x_3) = 2(x_2 \vee x_3) + (x_2 \oplus x_3) \tag{4}$$

является линейным, где $J = \mathrm{supp}(E) = \{f_1(\alpha_1, \alpha_2, \alpha_3) = \alpha_2 \vee \alpha_3, f_2(\alpha_1, \alpha_2, \alpha_3) = \alpha_2 \oplus \alpha_3\}$, $a_{f_1} = 2$, $a_{f_2} = 1$.

Пусть $\{b_1, \dots, b_{2^t}\} = B^t$ — произвольным, но фиксированным образом упорядоченное множество двоичных векторов длины t. Так как $|\mathcal{F}_t| = 2^{2^t}$, то множеству $\mathcal{F}_t$ можно поставить в соответствие $2^t \times 2^{2^t}$-матрицу $\mathcal{T}_t$ из нулей и единиц, каждый столбец (столбец истинности) которой представляет собой значения соответствующей ему булевой функции на всех наборах из B^t. Для булевой функции $f \in \mathcal{F}_t$ через $\mathcal{T}_t[f]$ будем обозначать соответствующий столбец истинности. Для $N \geqslant 2^t$ и $F = \{\varphi_1, \dots, \varphi_N\} \subset \mathcal{F}_t$ через A_F обозначим $2^t \times N$-матрицу вида

$$A_F = (\mathcal{T}_t[\varphi_1] | \dots | \mathcal{T}_t[\varphi_N]), \tag{5}$$

элементы которой (0 и 1) рассматриваются как элементы кольца $\mathbb{V}$ и среди столбцов которой найдутся 2^t столбцов, образующих обратимую $2^t \times 2^t$-матрицу. Пусть $E(x_1, \dots, x_t)$ — линейное MBA-выражение вида (3), $J = \mathrm{supp}(E) \subseteq F$. Для $f \in J$ через $j(f)$ обозначим номер столбца $\mathcal{T}_t[f]$ в матрице A_F вида (5). Выражению $E(x_1, \dots, x_t)$ поставим в соответствие вектор $\mathcal{E}(E(x_1, \dots, x_t)) \in \mathbb{V}^N$ вида

$$\mathcal{E}(E(x_1, \dots, x_t)) = (\varepsilon_1, \dots, \varepsilon_N) \in \mathbb{V}^N, \varepsilon_l = \begin{cases} a_f, \text{если } l = j(f), f \in \mathrm{supp}(E), \\ 0, \text{иначе.} \end{cases}$$

Во введенных обозначениях сформулируем и докажем результат, доказанный впервые в [2].

Теорема 1. *Пусть A_F — матрица вида (5), $E(x_1, \dots, x_t)$ — MBA-выражение вида (3), $\mathrm{supp}(E) \subseteq F$. $E(x_1, \dots, x_t) \equiv 0$ тогда и только тогда, когда $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$ является решением системы*

$$A_F \mathbf{y}^T = \mathbf{0}^T = (0, \dots, 0)^T \in \mathbb{V}^{2^t}. \tag{6}$$

Доказательство. Так как матрица $A_F \in \mathbb{V}^{2^t \times N}$ представима в виде

$$A_F = \begin{pmatrix} \varphi_1(0, \dots, 0) & \dots & \varphi_N(0, \dots, 0) \\ \varphi_1(0, \dots, 1) & \dots & \varphi_N(0, \dots, 1) \\ \vdots & \ddots & \vdots \\ \varphi_1(1, \dots, 1) & \dots & \varphi_N(1, \dots, 1) \end{pmatrix},$$

то

$$A_F \mathbf{y}^T = \begin{pmatrix} \sum_{j=1}^N y_j \varphi_j(0, \dots, 0) \\ \sum_{j=1}^N y_j \varphi_j(0, \dots, 1) \\ \vdots \\ \sum_{j=1}^N y_j \varphi_j(1, \dots, 1) \end{pmatrix}.$$

Следовательно,

$$A_F \mathbf{y}^T = \mathbf{0}^T \iff \forall \boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_t) \in B^t \ : \ \sum_{j=1}^N y_j \varphi_j(\alpha_1, \dots, \alpha_t) = 0. \tag{7}$$

Поэтому, из равенства $A_F \mathbf{y}^T = \mathbf{0}^T$, учитывая (1) и (7), получаем:

$$\begin{aligned} E(x_1, \dots, x_t) &= \sum_{f \in J \subset \mathcal{P}_t} a_f e_f(x_1, \dots, x_t) = \sum_{f \in J} a_f \sum_{i=0}^{n-1} 2^i f(\beta(x_1)_i, \dots, \beta(x_t)_i) \\ &= \sum_{i=0}^{n-1} 2^i \left(\sum_{f \in J} a_f f(\beta(x_1)_i, \dots, \beta(x_t)_i) \right) \\ &= \sum_{i=0}^{n-1} 2^i \left(\sum_{j=1}^N y_j \varphi_j(\beta(x_1)_i, \dots, \beta(x_t)_i) \right) = 0 \in \mathbb{V}. \end{aligned}$$

□

1.2. О генерации эквивалентных линейных MBA-выражений

Опишем множество эквивалентных линейных MBA-выражений, используя понятия линейной алгебры. Для матрицы (5) определим группу по сложению $C_F = \{\mathbf{y} \in \mathbb{V}^N \ : \ A_F \mathbf{y}^T = \mathbf{0}^T\}$, которая является также подмодулем $\mathbb{V}$-модуля $\mathbb{V}^N$. По каждому вектору $\mathbf{y} = (y_1, \dots, y_N) \in C_F$ может быть построено тождественно равное нулю линейное MBA-выражение вида

$$\mathcal{E}^{-1}(\mathbf{y}) = \sum_{i=1}^N y_i e_{\varphi_i}(x_1, \dots, x_t) = \sum_{i \in \mathrm{supp}(\mathbf{y})} y_i e_{\varphi_i}(x_1, \dots, x_t) \equiv 0 \in \mathbb{V}. \tag{8}$$

Отсюда, в частности, вытекает, что если для $i \in \mathrm{supp}(\mathbf{y})$ элемент y_i обратим в кольце $\mathbb{V}$, то

$$e_{\varphi_i}(x_1, \dots, x_t) = -y_i^{-1} \sum_{i \in \mathrm{supp}(\mathbf{y}) \setminus \{i\}} y_i e_{\varphi_i}(x_1, \dots, x_t). \tag{9}$$

Следовательно, левая часть равенства (9) (MBA-выражение без чередований) может быть *обфусцирована* с помощью правой части этого равенства (с помощью MBA-выражения, содержащего чередования).

По группе C_F построим фактор-множество

$$\mathbb{V}^N/C_F = \left\{ C(\mathbf{b}) = C_F + \mathbf{a} \,:\, \mathbf{a} \in \mathbb{V}^N, A_F\mathbf{a}^T = \mathbf{b}^T \in \mathbb{V}^{2^t} \right\}, |\mathbb{V}^N/C_F| = |\mathbb{V}|^{2^t}.$$

Из определения фактор-классов $C(\mathbf{b})$ вытекает, для любого $\mathbf{b} \in \mathbb{V}^{2^t}$ и любых $\mathbf{y}, \mathbf{y}' \in C(\mathbf{b})$ линейные МВА-выражения $\mathcal{E}^{-1}(\mathbf{y})$ и $\mathcal{E}^{-1}(\mathbf{y}')$ являются эквивалентными:

$$\mathcal{E}^{-1}(\mathbf{y}) = \sum_{i \in \mathrm{supp}(\mathbf{y})} y_i e_{\varphi_i}(x_1, \dots, x_t) = \sum_{j \in \mathrm{supp}(\mathbf{y}')} y'_j e_{\varphi_j}(x_1, \dots, x_t) = \mathcal{E}^{-1}(\mathbf{y}'). \tag{10}$$

Также получаем, что для заданного МВА-выражения в рамках заданной матрицы A_F существует $|C_F|$ эквивалентных выражений. Отметим, что в [8] вектор $A_F\mathbf{y}^T$ назван *подписью* МВА-выражения $E(x_1, \dots, x_t)$, где $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$, и впервые доказано, что два линейных МВА-выражения эквивалентны, если равны их подписи.

К правой и левой частям равенства (10) могут быть прибавлены тождественно равные нулю МВА-выражения (не обязательно одинаковые для левой и правой части), при этом равенство сохранится. Алгоритм LinMBAGen (см. алгоритм 1) позволяет по заданному линейному МВА-выражению $E(x_1, \dots, x_t)$ найти эквивалентное ему линейное МВА-выражение заданной длины w.

Algorithm 1. LinMBAGen

Data: $E(x_1, \dots, x_t)$, $w \in [1, \dots, N]$
Result: $E'(x_1, \dots, x_t) \equiv E(x_1, \dots, x_t)$, $\mathrm{wt}(\mathcal{E}(E'(x_1, \dots, x_t))) \geqslant w$
/* Случайно выбрать тождественно равное нулю МВА-выражение */
1 Случайно выбрать $\mathbf{y} \in C_F$
/* Проверить условие на вес вектора */
2 **if** $\mathrm{wt}(\mathcal{E}(E(x_1, \dots, x_t)) + \mathbf{y}) \geqslant w$ **then**
3 **return** $\mathcal{E}^{-1}(\mathcal{E}(E(x_1, \dots, x_t)) + \mathbf{y})$
4 **else**
5 **go to** 1

Возвращаемое алгоритмом LinMBAGen выражение не содержит слагаемых с одинаковым битовым выражением. Поэтому, для того, чтобы длина выражения была как можно больше, матрица A_F должна иметь как можно больше разных столбцов. Так как для t переменных существует 2^{2^t} возможных булевых функций, то максимальная длина линейного МВА-выражения равна 2^{2^t}. Если исключить требование на неповторяемость битовых выражений, то с помощью алгоритма LinMBAGenAnyLen (см. алгоритм 2) может быть сгенерировано линейное МВА-выражение любой длины.

2. О деобфускации линейных МВА-выражений

В настоящей работе для деобфускации линейных МВА-выражений применяется техника, используемая при декодировании линейных кодов и дешифровании кодовых криптосистем. Поэтому сначала приведем необходимые сведения о линейных кодах.

2.1. Линейные коды

Напомним, что линейным $[m, k, d]_q$-кодом C над конечным полем $\mathbb{F}_q$ называется любое подпространство C размерности k пространства $\mathbb{F}_q^m$, для которого минимальный вес Хэмминга ненулевых векторов кода C равен d. Величина d называется *кодовым расстоянием* кода C. Из любых k линейно независимых векторов подпространства C может быть построена $k \times m$-матрица, называемая *порождающей* матрицей кода C. Если G — порождающая матрица кода C, то любая $(m - k) \times m$-матрица H

Algorithm 2. LinMBAGenAnyLen

Data: $E(x_1, \dots, x_t)$, $w \in \mathbb{N}$
Result: $E'(x_1, \dots, x_t) \equiv E(x_1, \dots, x_t)$, число слагаемых не менее w
1 $E'(x_1, \dots, x_t) = E(x_1, \dots, x_t)$
/* Случайно выбрать тождественно равное нулю MBA-выражение */
2 Случайно выбрать $\mathbf{y} \in C_F$
/* Проверить условие на количество слагаемых */
3 **if** $\mathrm{wt}(\mathcal{E}(E'(x_1, \dots, x_t))) + \mathrm{wt}(\mathbf{y}) \geqslant w$ **then**
4 **return** $E'(x_1, \dots, x_t) + \mathcal{E}^{-1}(\mathbf{y})$
5 **else**
6 $E'(x_1, \dots, x_t) = E'(x_1, \dots, x_t) + \mathcal{E}^{-1}(\mathbf{y})$
7 **go to** 2

ранга $m-k$, такая, что $GH^T = O$, называется *проверочной* матрицей кода C, где O — нулевая $k \times (m-k)$-матрица. Для любого вектора $\mathbf{z} \in \mathbb{F}_q^m$ вектор $\mathbf{s} = \mathbf{z}H^T$ называется *синдромом* вектора $\mathbf{z}$ относительно матрицы H. Для заданной проверочной матрицы H кода C, синдрома $\mathbf{s}$ и числа $t = \lceil (d-1)/2 \rceil$ задача нахождения вектора $\mathbf{z} \in \mathbb{F}_q^N$, такого, что $\mathrm{wt}(\mathbf{z}) \leqslant t$, называется задачей синдромного декодирования. Известно, что эта задача для случайно выбранного кода C является NP-полной [10].

2.2. Алгоритмы деобфускации линейных MBA-выражений

Предлагаемые в [8] и [4] способы упрощения линейного MBA-выражения $E(x_1, \dots, x_t)$ основаны на том, что матрица A_F содержит 2^t линейно независимых столбцов, поэтому подпись $A_F(\mathcal{E}(E(x_1, \dots, x_t)))^T$ может быть представлена как линейная комбинация не более чем 2^t столбцов матрицы A_F. Следовательно, любое линейное MBA-выражение может быть упрощено до линейного MBA-выражения длины не более 2^t. Вопрос о том, может ли быть получено MBA-выражение строго меньшей длины, чем 2^t, в работах [8] и [4] не изучался. Заметим, что отличительной особенностью работы [4] по сравнению с [8] является упрощение вычисления подписи. Именно в [4] показано, что для $E(x_1, \dots, x_t)$ можно напрямую (без разбора на составляющие битовые подвыражения) вычислить вектор подписи $\mathbf{s} = A_F\mathbf{y}^T$ на 2^t наборах значений t переменных (наименее значащие биты в этих переменных принимают все возможные значения, а все остальные биты имеют нулевое значение). В настоящей работе будет показано, что в ряде случаев можно получить упрощенное MBA-выражение с количеством слагаемых, существенно меньшим 2^t.

Задачу деобфускации линейного выражения $E(x_1, \dots, x_t)$ вида (3), как задачу поиска эквивалентного выражения с меньшим числом чередований (меньшей длиной), можно сформулировать следующим образом: найти такой вектор $\mathbf{z} \in \mathbb{V}^N$, чтобы 1) $A_F\mathbf{y}^T = A_F\mathbf{z}^T$ и 2) $\mathrm{wt}(\mathbf{z}) < \mathrm{wt}(\mathbf{y})$ для $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$. Первое условие гарантирует, что MBA-выражения являются эквивалентными, а второе означает меньшую длину MBA-выражения $\mathcal{E}^{-1}(\mathbf{z})$. Вектор $\mathbf{s} = A_F\mathbf{y}^T$ является аналогом синдрома вектора $\mathbf{y}$ относительно матрицы A_F. Отличием является то, что элементами матрицы A_F являются элементы кольца $\mathbb{V}$, в общем случае не являющегося конечным полем. Поэтому вектор $\mathbf{s}$ будем называть $\mathbb{V}$-*синдромом* вектора $\mathbf{y}$ относительно матрицы A_F. Задачу упрощения линейного MBA-выражения $E(x_1, \dots, x_t)$ сформулируем следующим образом.

Задача 1. *Для заданной $2^t \times N$-матрицы A_F и заданного линейного MBA-выражения $E(x_1, \dots, x_t)$ с $\mathbb{V}$-синдромом $\mathbf{s} = A_F \times \left(\mathcal{E}(E(x_1, \dots, x_t))\right)^T$ найти вектор $\mathbf{z} \in C_F(\mathbf{s})$ минимального веса Хэмминга.*

Рассмотрим линейное MBA-выражение $E(x_1, \dots, x_t)$ и соответствующий вектор $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$. Предположим, что существует такой вектор $\mathbf{z}$ минимального веса $\mathrm{wt}(\mathbf{z}) = r$, что $A_F\mathbf{y}^T = A_F\mathbf{z}^T$. Поиск такого вектора перебором имеет сложность $O((|\mathbb{V}|-1)^r \binom{N}{r})$. И даже для небольших значений r такой

перебор может потребовать значительных вычислительных ресурсов. Например, при $|\mathbb{V}| = 2^{32}$, $N = 15$ и $r = 3$ сложность поиска будет не менее 2^{104}.

В настоящей работе предлагается способ упрощения, закаляющийся в применении метода *декодирования по информационным совокупностям* (Information Set Decoding) [11]. Для описания этого метода нам потребуется следующее утверждение.

Утверждение 1. *Пусть $N \geqslant 2^t$, S — обратимая над $\mathbb{V}$ квадратная $2^t \times 2^t$-матрица, Q — перестановочная $N \times N$-матрица, причем $S \cdot A_F \cdot Q = [E_{2^t}|P]$, где E_{2^t} — единичная $2^t \times 2^t$-матрица. Тогда $\mathcal{L}(G) = \mathcal{C}_F$, где*

$$G = [-P^T|E_{N-2^t}]Q^T, \tag{11}$$

$\mathcal{L}(G)$ — линейная оболочка, натянутая на строки матрицы G.

Доказательство. Из равенства

$$SA_FG^T = SA_FQ[-P^T|E_{N-2^t}]^T = [E_{2^t}|P] \times [-P^T|E_{N-2^t}]^T = [-P + P] = O$$

вытекает, что $\mathcal{L}(G)$ содержится во множестве решений системы $SA_F\mathbf{y}^T = \mathbf{0}^T$, которое, в силу обратимости S, совпадает с $\mathcal{C}_F$. Следовательно, $\mathcal{L}(G) \subseteq \mathcal{C}_F$. Система уравнений $[E_{2^t}|P]\mathbf{y}^T = \mathbf{0}^T \in \mathbb{V}^{2^t}$ имеет $N - 2^t$ независимых неизвестных и 2^t зависимых. Следовательно эта система имеет $|\mathbb{V}^{N-2^t}|$ решений. Заметим, что матрица G порождает множество также из $\mathbb{V}^{N-2^t}$ векторов. Поэтому $\mathcal{L}(G) = \mathcal{C}_F$. □

Для $\mathcal{C}_F$ *информационной совокупностью* назовем множество $\tau \subset \{1, \dots, N\}$, $|\tau| = N - 2^t$, для которого столбцы матрицы G, такой, что $\mathcal{L}(G) = \mathcal{C}_F$ (например, вида (11)) с номерами из τ образуют квадратную обратимую матрицу над кольцом $\mathbb{V}$. Матрицу, образованную столбцами матрицы G с номерами из τ, будем обозначать G_τ. Вектор $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$ может быть представлен в виде

$$\mathbf{y} = \mathbf{c} + \mathbf{z}, \mathrm{wt}(\mathbf{z}) = r, \tag{12}$$

где $\mathbf{c} = \mathbf{m}G \in \mathcal{C}_F$, так как очевидно, что

$$A_F\mathbf{y}^T = A_F(\mathbf{c} + \mathbf{z})^T = A_F\mathbf{c}^T + A_F\mathbf{z}^T = A_F\mathbf{z}^T.$$

Пусть τ — информационная совокупность, причем $\tau \cap \mathrm{supp}(\mathbf{z}) = \varnothing$. Следовательно, $\mathbf{y}_\tau = \mathbf{c}_\tau = \mathbf{m}G_\tau$. Откуда $\mathbf{z} = \mathbf{y} - \mathbf{c}_\tau G_\tau^{-1}G = \mathbf{y} - \mathbf{y}_\tau G_\tau^{-1}G$. Поэтому задача нахождения вектора $\mathbf{z}$ веса r (задача 1) может быть сведена к нахождению информационной совокупности, не пересекающейся с носителем вектора $\mathbf{z}$. Этот подход и его модификации (см., например, [12]) используются в теории кодирования для исправления ошибок при помощи кодов умеренной длины, если не известны более эффективные алгоритмы; а в теории асимметричных кодовых криптосистем декодирование по информационным совокупностям является наиболее эффективным способом дешифрования сообщений без знания секретного ключа [13]. Отличительной особенностью рассматриваемого здесь алгоритма декодирования по информационным совокупностям является лишь то, что матрица G определена над коммутативным кольцом $\mathbb{V}$, а не над конечным полем $\mathbb{F}_q$. Способ нахождения вектора веса не более r за заданное количество итераций I реализован в виде алгоритма ISD (см. алгоритм 3). Сложность этого алгоритма составляет

$$O\left(\min\left\{\binom{N}{N-2^t} \cdot \binom{N-r}{N-2^t}^{-1}, I\right\} \cdot (N-2^t)^3\right) \tag{13}$$

и соответствует среднему количеству попыток нахождения информационной совокупности, не пересекающейся с носителем вектора $\mathbf{z}$ в представлении (12), умноженному на $(N - 2^t)^3$ — сложность

Algorithm 3. ISD

Data: $\mathbf{y} \in \mathbb{V}^N$, $G \in \mathbb{V}^{(N-2^t)\times N}$, $r, I \in \mathbb{N}$
Result: ($\mathbf{z} \in \mathbb{V}^N$, $\mathbf{y} - \mathbf{z} \in \mathcal{L}(G)$, $\mathrm{wt}(\mathbf{z}) \leqslant r$) или $\perp$

```
1  i = 0
2  i = i + 1
   /* проверить счетчик максимального числа итераций алгоритма; счетчик используется
      для предотвращения зацикливания                                              */
3  if i = I then
4      return ⊥
5  выбрать τ ⊂ {1, ..., N}, |τ| = N − 2^t
   /* проверить, что τ является информационной совокупностью: τ — информационная
      совокупность, если элемент det(G_τ) обратим в V                              */
6  if det(G_τ) необратим в V then
7      go to 2
8  найти вектор z = y − y_τ G_τ^{-1} G
   /* проверить условие на вес найденного вектора                                  */
9  if wt(z) ⩽ r then
10     return z
11 else
12     go to 2
```

нахождения обратной матрицы G_τ^{-1}. Например, при $|\mathbb{V}| = 2^{32}$, $N = 15$, $r = 3$, $t = 2$ сложность составляет порядка $\min\{3, I\} \cdot 11^3$.

Заметим, что в задачах криптографии и теории кодирования, где используется некоторый $[m, k, d]_q$-код, входной вектор $\mathbf{y}$ алгоритма ISD обычно такой, что вектор минимального веса с таким же синдромом единственен. Поэтому для корректного декодирования или расшифрования сообщения применяется алгоритм ISD, где $r = \lceil (d-1)/2 \rceil$. Сложность (13) алгоритма ISD не является полиномиальной и пока нет оснований предполагать, что задача 1 менее сложна, чем задача синдромного декодирования для линейного кода. Однако, при упрощении линейного MBA-выражения $E(x_1, \ldots, x_t)$ успешным упрощением можно считать не только случай решения задачи 1, но и когда в $\mathcal{C}(\mathbf{s})$ найден такой вектор $\mathbf{z}$, что $\mathrm{wt}(\mathbf{z}) < \mathrm{wt}(\mathcal{E}(E(x_1, \ldots, x_t)))$. Другими словами, под успешным упрощением можно понимать сокращение числа чередований в MBA-выражении. Поэтому задача упрощения линейного MBA-выражения может быть сформулирована так.

Задача 2. *Для заданной $2^t \times N$-матрицы A_F, заданного линейного MBA-выражения $E(x_1, \ldots, x_t)$ с $\mathbb{V}$-синдромом $\mathbf{s} = \mathcal{E}(E(x_1, \ldots, x_t)) \times A_F^T$ и заданного $I \in \mathbb{N}$ найти за I итераций такой вектор $\mathbf{z} \in C_F(\mathbf{s})$, что* $\mathrm{wt}(\mathbf{z}) < \mathrm{wt}(\mathcal{E}(E(x_1, \ldots, x_t)))$.

Алгоритм ISDMinSearch (см. алгоритм 4), предназначенный для решения задачи 2, возвращает вектор $\mathbf{z}$ с наименьшим весом среди I опробованных. Сложность алгоритма ISDMinSearch составляет

$$O\left(I \cdot (N - 2^t)^3\right). \tag{14}$$

Отметим, что алгоритмы ISD и ISDMinSearch могут быть распараллелены, так опробование разных наборов τ может выполняться независимо. Алгоритм MBASimplify (см. алгоритм 5) возвращает по линейному MBA-выражению $E(x_1, \ldots, x_t)$ эквивалентное ему линейное MBA-выражение меньшей длины и возвращает $E(x_1, \ldots, x_t)$, если при заданных входных параметрах не нашлось MBA-

Algorithm 4. ISDMinSearch

Data: $\mathbf{y} \in \mathbb{V}^N$, $G \in \mathbb{V}^{(N-2^t)\times N}$, $I \in \mathbb{N}$
Result: $\mathbf{z} \in \mathbb{V}^N$, $\mathbf{y} - \mathbf{z} \in \mathcal{L}(G)$, $\mathrm{wt}(\mathbf{z}) \longrightarrow \min$

1 $i = 0$, $mw = \mathrm{wt}(\mathbf{y})$, $\mathbf{z} = \mathbf{y}$
2 $i = i + 1$
/* проверить счетчик максимального числа итераций алгоритма */
3 **if** $i = I$ **then**
4 вернуть $\mathbf{z}$
5 выбрать $\tau \subset \{1, \dots, N\}$, $|\tau| = N - 2^t$
/* проверить, что τ является информационной совокупностью */
6 **if** $\det(G_\tau)$ *необратим в* $\mathbb{V}$ **then**
7 **go to** 2
8 найти вектор $\mathbf{x} = \mathbf{y} - \mathbf{y}_\tau G_\tau^{-1} G$
9 **if** $\mathrm{wt}(\mathbf{x}) \leqslant mw$ **then**
10 $mw = \mathrm{wt}(\mathbf{x})$, $\mathbf{z} = \mathbf{x}$
11 **else**
12 **go to** 2

выражения меньшей длины. Заметим, что если в линейном MBA-выражении $E(x_1, \dots, x_t)$ не более g слагаемых вида $a_f e_f(x_1, \dots, x_t)$, то шаг 1 алгоритма 5 имеет вычислительную сложность не более $O(g2^t)$, так как по каждому слагаемому $a_f e_f(x_1, \dots, x_t)$ прямыми вычислениями битового выражения $e_f(x_1, \dots, x_t)$ на всех 2^t наборах из B^t может быть построен столбец истинности $\mathcal{T}_t[f]$. В алгоритме MBASimplify предполагается, что выражение $E(x_1, \dots, x_t)$ не содержит слагаемых с одинаковыми битовыми выражениями вида $e_f(x_1, \dots, x_t)$. Если же такие имеются, то перед выполнением алгоритма следует привести подобные слагаемые. Так как матрица A_F строится рандомизированно, то алгоритм MBASimplify может быть выполнен несколько раз, что позволит выбрать наилучшее решение. Другим способом поиска наилучшего решения является увеличение параметров I и R, однако увеличение R может замедлить работу алгоритма ISDMinSearch, так как в этом случае растут размеры матрицы G_τ, для которой в алгоритме ISDMinSearch находится обратная матрица.

2.3. Экспериментальная оценка ISD и ISDMinSearch

В работе проведено экспериментальное исследование алгоритмов ISD и ISDMinSearch. В соответствии с (13) и (14), вычислительная сложность этих алгоритмов зависит от t — количества переменных в линейном MBA-выражении, N — количества столбцов в матрице A_F, I — количества итераций поиска. Сложность ISD также зависит от r — максимального веса искомого вектора $\mathbf{z}$ в (12). Поэтому параметры N, t, I и r выбраны так, чтобы эксперименты могли быть проведены в среде Sage Math 9.2 на компьютере x64 AMD FX(tm)-8350 Eight-Core Processor 4.00 GHz с восьмиядерным процессором и 16ГБ оперативной памяти. В качестве кольца $\mathbb{V}$ были рассмотрены $\mathbb{Z}_2$, $\mathbb{Z}_{2^2}$, $\mathbb{Z}_{2^4}$, $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$. Заметим, что кольца $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$ соответствуют типам unsigned char, unsigned short и unsigned int языков программирования C/C++, поэтому результаты для этих колец могут быть использованы, например, при обфускации или деобфускации программ на этих языках. Для случая $t = 2$ количество всех булевых функций от двух переменных равно $2^{2^2} = 16$. Поэтому в качестве матрицы A_F взята (4×15)-матрица, состоящая из всех ненулевых столбцов истинности таблицы $\mathcal{T}_2$, так как такие размеры матрицы позволяют провести вычислительный эксперимент на выбранном компьютере за приемлемое время.

Algorithm 5. MBASimplify

Data: $E(x_1, \dots, x_t)$, I, R
Result: $E'(x_1, \dots, x_t) \equiv E(x_1, \dots, x_t)$
/* Построить множество булевых функций f, соответствующих $e_f(x_1, \dots, x_t)$ в (3) */
По $E(x_1, \dots, x_t)$ построить $F \subset \mathcal{F}_t$
/* Построить матрицу A_F^0 */
for $f \in F$ **do**
    Построить столбец $\mathcal{T}_t[f]$
Из столбцов $\mathcal{T}_t[f]$, $f \in F$, построить матрицу A_F^0
/* Случайно достроить матрицу A_F^0 до A_F */
Выбрать случайно R разных $(0,1)$-столбцов, не содержащихся в A_F^0, и приписать их к матрице A_F^0, образовав $(2^t \times N)$-матрицу A_F
/* Подготовить входные данные для ISDMinSearch */
По матрице A_F построить $(N - 2^t) \times N$-матрицу G (см. утверждение 1)
$\mathbf{y} = \mathbf{0} \in \mathbb{V}^N$
for $f \in F$ **do**
    /* $\mathcal{E}(e_f(x_1, \dots, x_t))$ — вектор веса 1, в котором координата равна 1 в позиции, соответствующей столбцу $\mathcal{T}_t[f]$ в матрице A_F */
    $\mathbf{y} = \mathbf{y} + a_f \mathcal{E}(e_f(x_1, \dots, x_t))$
$\mathbf{z} = \mathsf{ISDMinSearch}(\mathbf{y}, G, I)$
return $\mathcal{E}^{-1}(\mathbf{z})$

2.3.1. Исследование алгоритма ISD

В рамках анализа ISD оценена вероятность нахождения вектора минимального веса при $t = 2, 3, 4$ и $N \approx 2^{t+2}$, а в случае $t = 3$ эта вероятность оценена в зависимости от $N \in \{16, 32, 48, 64\}$. Во всех экспериментах для $2^t \times N$-матрицы A_F полного ранга вида (5), случайно выбранной из кольца $\mathbb{Z}_n^{2^t \times N}$, используя утверждение 1, сначала находилась $(N - 2^t) \times N$-матрица G полного ранга, для которой $GA_F^T = O$. Далее из линейной оболочки $\mathcal{L}(G)$ было выбрано случайным образом сто векторов, для каждого из которых случайно и независимо от других векторов генерировался и прибавлялся вектор веса $r \in \{1, \dots, 2^t\}$. Фактически, для ста случайно сгенерированных линейных MBA-выражений с r слагаемыми был выполнен алгоритм LinMBAGen с параметром $w = 0$ (для каждого входного MBA-выражения шаг 1 алгоритма выполнялся только один раз). Для каждого из полученных векторов был выполнен алгоритм ISD с целью поиска вектора с таким же синдромом и веса не более r. Результаты экспериментов показаны в таблицах 1–6, где δ — доля входных векторов $\mathbf{y}$, для которых был найден вектор с заданным ограничением на вес, $\mathrm{wt}(\mathbf{y})$ — средний вес входного вектора, $\mathrm{wt}(\mathbf{z})$ — средний вес первого найденного алгоритмом ISD вектора, удовлетворяющего условию на вес (в эксперименте, если вектор с заданным условием за I итераций не был найден, то возвращался вектор с наименьшим найденным весом). В таблицах 1, 3 и 6 представлены результаты эксперимента, целью которого было исследование вероятности нахождения вектора минимального веса при $N/2^t \approx \mathrm{const}$ и $t = 2, 3, 4$, а в таблицах 2, 3, 4 и 5 представлены результаты исследования этой вероятности при $t = \mathrm{const}(t = 3)$ и $N = 16, 32, 48, 64$.

Из таблиц 1, 3 и 6 видно, что если вес r вектора, соответствующего MBA-выражению $E(x_1, \dots, x_t)$, принадлежит множеству $\{1, 2, 3, 2^t\}$, то с близкой к единице вероятностью алгоритмом ISD, а значит, и алгоритмом ISDMinSearch, по обфусцированному MBA-выражению LinMBAGen$(E(x_1, \dots, x_t), w)$ или LinMBAGenAnyLen$(E(x_1, \dots, x_t), w)$ для любого $w \in \mathbb{N}$ будет найден вектор веса не более r, соответствующий MBA-выражению $E'(x_1, \dots, x_t)$, тождественно равному $E(x_1, \dots, x_t)$. Для других значений r

Table 1. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 2$, $N = 2^{2^t} - 1 = 15$, $r \in \{1, \dots, 4\}$, the number of experiments for the given weight r — 100.

Таблица 1. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 2$, $N = 2^{2^t} - 1 = 15$, $r \in \{1, \dots, 4\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	7.50	0.45	1.0	11.11	0.72	1.0	14.16	0.90
2	**1.0**	**7.16**	**1.13**	**1.0**	**11.21**	**1.68**	**1.0**	**14.14**	**1.90**
3	1.0	7.39	1.71	1.0	11.15	2.43	1.0	14.20	2.94
4	1.0	7.65	1.99	1.0	11.12	3.00	1.0	14.14	3.76

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	14.94	1.00	1.0	14.99	1.00	1.0	15.0	1.00
2	**1.0**	**14.94**	**2.00**	**1.0**	**14.99**	**2.00**	**1.0**	**15.0**	**2.00**
3	1.0	14.96	2.96	1.0	14.99	2.97	1.0	15.0	3.00
4	1.0	14.93	3.88	1.0	14.99	3.94	1.0	15.0	3.94

Table 2. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 3$, $N = 16$, $r \in \{1, \dots, 8\}$, the number of experiments for the given weight r — 100.

Таблица 2. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 3$, $N = 16$, $r \in \{1, \dots, 8\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	8.07	0.49	1.0	11.98	0.81	1.00	14.94	0.86
2	1.0	8.01	1.03	1.0	11.88	1.58	1.00	15.04	1.76
3	1.0	8.17	1.63	1.0	12.12	2.43	1.00	14.96	2.71
4	1.0	7.82	2.44	1.0	11.81	3.35	0.99	14.96	3.64
5	1.0	8.01	3.08	1.0	11.99	4.12	0.94	14.94	4.76
6	1.0	8.25	3.37	1.0	12.19	5.00	1.00	14.91	5.55
7	1.0	8.09	3.87	1.0	11.74	5.66	1.00	14.81	6.56
8	1.0	8.00	3.72	1.0	12.02	5.82	1.00	14.95	7.39

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	15.93	1.00	1.00	16.0	1.00	1.00	16.0	1.00
2	1.00	15.93	1.98	1.00	16.0	2.00	1.00	16.0	2.00
3	1.00	15.95	2.96	1.00	16.0	3.00	0.99	16.0	3.04
4	1.00	15.93	3.96	0.97	16.0	4.07	0.96	16.0	4.14
5	1.00	15.94	4.97	0.93	16.0	5.19	0.93	16.0	5.21
6	0.97	15.93	5.95	0.80	16.0	6.32	0.77	16.0	6.45
7	1.00	15.98	6.98	0.87	16.0	7.10	0.73	16.0	7.25
8	1.00	15.94	7.91	1.00	16.0	7.92	1.00	16.0	7.99

вероятность успеха значительно зависит от кольца. Например, при $t = 2$ и $\mathbb{V} = \mathbb{Z}_2$ алгоритм ISD почти всегда находит вектор с заданным ограничением на вес. Но уже для кольца $\mathbb{V} = \mathbb{Z}_{2^2}$ при $r = 5$ и $r = 6$ алгоритм не находит векторы веса с заданным ограничением на вес с вероятностями соответственно 0.21 и 0.31. Для колец, начиная с $\mathbb{Z}_{2^4}$, имеются значения r, для которых алгоритм ни для одного входного вектора (при заданных ограничениях на количество итераций) не нашел вектор веса не более r. Также видно, что для каждого r с ростом мощности кольца растет и средний

Table 3. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 3$, $N = 32$, $r \in \{1, \ldots, 8\}$, the number of experiments for the given weight r — 100.

Таблица 3. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 3$, $N = 32$, $r \in \{1, \ldots, 8\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	16.30	0.51	1.00	24.11	0.82	1.00	29.93	0.93
2	1.0	16.16	1.06	1.00	24.09	1.56	1.00	29.85	1.88
3	1.0	16.14	1.97	0.99	23.97	2.49	0.98	29.92	2.87
4	1.0	16.28	2.69	1.00	24.06	3.54	0.93	29.89	4.04
5	1.0	16.40	3.31	1.00	23.90	4.53	0.99	29.90	4.77
6	1.0	16.02	4.03	1.00	23.75	5.31	1.00	29.93	5.80
7	1.0	16.38	3.85	1.00	23.84	5.84	1.00	29.78	6.64
8	1.0	16.40	3.79	1.00	24.29	6.16	1.00	29.93	7.52

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	31.80	1.00	1.00	32.0	1.00	1.00	32.0	1.00
2	1.00	31.80	2.00	1.00	32.0	1.99	1.00	32.0	1.98
3	0.99	31.78	3.02	0.97	32.0	3.11	0.97	32.0	3.15
4	0.67	31.79	5.20	0.70	32.0	5.12	0.63	32.0	5.39
5	0.36	31.81	6.86	0.42	32.0	6.69	0.18	32.0	7.42
6	0.52	31.84	6.91	0.45	32.0	7.05	0.28	32.0	7.39
7	0.99	31.86	6.99	0.63	32.0	7.35	0.61	32.0	7.37
8	1.00	31.82	7.95	1.00	32.0	7.98	1.00	32.0	7.99

Table 4. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 3$, $N = 48$, $r \in \{1, \ldots, 8\}$, the number of experiments for the given weight r — 100.

Таблица 4. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 3$, $N = 48$, $r \in \{1, \ldots, 8\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	24.52	0.51	1.00	36.20	0.79	1.00	45.08	0.91
2	1.0	24.48	1.19	1.00	36.37	1.58	0.99	45.03	1.91
3	1.0	24.29	2.04	0.98	36.37	2.58	0.91	45.12	3.28
4	1.0	24.19	2.68	1.00	36.07	3.58	0.72	45.00	4.77
5	1.0	24.31	3.23	1.00	36.41	4.49	0.94	45.15	5.09
6	1.0	24.39	3.71	1.00	36.18	4.96	1.00	45.06	5.88
7	1.0	24.21	3.64	1.00	36.34	5.79	1.00	45.00	6.76
8	1.0	24.45	4.23	1.00	36.27	6.05	1.00	45.08	7.53

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	47.87	1.00	1.00	48.0	0.98	1.00	48.0	1.00
2	0.99	47.86	2.03	1.00	48.0	1.94	1.00	48.0	2.00
3	0.73	47.86	4.26	0.75	48.0	4.10	0.79	48.0	3.93
4	0.30	47.84	6.63	0.31	48.0	6.58	0.36	48.0	6.39
5	0.26	47.85	7.15	0.20	48.0	7.32	0.14	48.0	7.47
6	0.39	47.89	7.21	0.24	48.0	7.48	0.26	48.0	7.46
7	1.00	47.84	7.00	0.63	48.0	7.36	0.57	48.0	7.42
8	1.00	47.82	7.94	1.00	48.0	7.99	1.00	48.0	7.99

Table 5. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 3$, $N = 64$, $r \in \{1, \dots, 8\}$, the number of experiments for the given weight r — 100.

Таблица 5. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 3$, $N = 64$, $r \in \{1, \dots, 8\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	31.62	0.44	1.0	48.06	0.75	1.00	60.05	0.95
2	1.0	31.68	1.09	1.0	48.00	1.63	0.99	60.00	1.97
3	1.0	31.79	1.80	1.0	47.98	2.53	0.75	60.01	3.97
4	1.0	31.69	2.80	1.0	47.85	3.57	0.66	59.99	4.98
5	1.0	31.44	3.43	1.0	48.14	4.58	0.95	60.04	5.04
6	1.0	31.79	3.85	1.0	48.29	5.19	1.00	59.79	5.87
7	1.0	31.42	3.77	1.0	48.07	5.75	1.00	59.95	6.75
8	1.0	31.74	3.81	1.0	48.09	5.95	1.00	59.97	7.49

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	63.77	0.99	1.00	64.0	1.00	1.00	64.0	1.00
2	1.00	63.74	1.98	0.99	64.0	2.04	0.99	64.0	2.06
3	0.58	63.74	4.96	0.55	64.0	5.09	0.65	64.0	4.58
4	0.15	63.76	7.29	0.26	64.0	6.80	0.24	64.0	6.86
5	0.12	63.77	7.57	0.14	64.0	7.49	0.13	64.0	7.51
6	0.33	63.76	7.29	0.23	64.0	7.05	0.20	64.0	7.58
7	1.00	63.76	6.99	0.62	64.0	7.38	0.64	64.0	7.34
8	1.00	63.76	7.96	1.00	64.0	7.99	1.00	64.0	7.99

вес возвращаемого вектора. Это свидетельствует о том, что чем больше кольцо, тем менее вероятно по линейному MBA-выражению $E(x_1, \dots, x_t)$ произвольной длины найти эквивалентное MBA-выражение длины не более r. Для примера в выделенной в таблице 1 строке при $r = 2$ средний вес найденного вектора для $\mathbb{V} = \mathbb{Z}_2$ равен 1.13, но уже, начиная с $\mathbb{V} = \mathbb{Z}_{2^8}$, средний вес равен двум.

Результаты в таблицах 2–5 показывают, что с ростом N доля δ при $t = 3$ не возрастает, а для большинства колец только убывает, при этом средний вес возвращаемого вектора с ростом N меняется несущественно (как в большую, так и в меньшую стороны). Поэтому представляется, что значение N порядка $O(2^{t+2})$ может являться достаточным для алгоритма ISD.

2.3.2. Исследование алгоритма ISDMinSearch

Для алгоритма ISDMinSearch при $t = 2, \dots, 5$ и $I = 1000$ оценен средний вес возвращаемого вектора $\mathbf{z}$, когда входной вектор $\mathbf{y}$ выбран случайно и равновероятно из $\mathbb{V}^N$. Результаты оценки среднего веса представлены в таблице 7. Также при тех же t исследована зависимость среднего веса вектора, возвращаемого алгоритмом ISDMinSearch, от количества итераций $I \in \{10, 100, 1000, 10000\}$. Именно для случайно выбранного вектора веса N алгоритм ISDMinSearch вызывался с числом итераций 10000, в ходе выполнения которого фиксировался минимальный вес после $i \in \{10, 100, 1000, 10000\}$ итераций. Всего для каждого t было случайно выбрано по 10 векторов веса N. Результаты показаны в таблице 8. Кроме этого оценена зависимость среднего веса вектора, возвращаемого алгоритмом ISDMinSearch, от параметра N. Эксперименты проведены для $t = 3$ при $N = 16, 32, 48, 64$, а результаты представлены в таблице 9.

Результаты из таблицы 7 показывают, что для случайно выбранного вектора из $\mathbb{V}^N$ с ростом мощности кольца растет и средний вес wt(z) найденного алгоритмом ISDMinSearch вектора при фиксированном числе итераций I. В таблице 7 выделены ячейки, в которых количество слагаемых

Table 6. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 4$, $N = 64$, $r \in \{1, \ldots, 16\}$, the number of experiments for the given weight r — 100.

Таблица 6. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 4$, $N = 64$, $r \in \{1, \ldots, 16\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	31.53	0.50	1.00	48.32	0.77	1.00	60.08	0.92
2	1.00	31.49	1.04	1.00	48.35	1.46	1.00	60.05	1.83
3	0.99	31.31	1.56	0.99	48.48	2.23	0.99	60.10	2.91
4	1.00	31.60	2.46	0.92	48.34	3.45	0.72	59.91	6.84
5	1.00	31.44	3.80	0.79	48.43	5.26	0.29	60.07	11.75
6	1.00	31.38	4.60	0.69	48.28	7.06	0.04	60.10	14.56
7	1.00	31.89	5.86	0.94	48.35	6.69	0.01	60.19	14.79
8	1.00	31.84	6.61	1.00	48.30	7.71	0.01	59.77	14.82
9	1.00	31.35	6.99	1.00	48.14	8.62	0.00	59.94	14.88
10	1.00	31.81	7.43	1.00	48.42	9.42	0.12	59.72	14.40
11	1.00	32.37	7.77	1.00	48.44	10.32	0.54	59.74	12.82
12	1.00	31.51	7.88	1.00	48.55	11.02	1.00	60.02	11.80
13	1.00	31.95	7.87	1.00	48.28	11.46	1.00	59.69	12.80
14	1.00	31.34	7.93	1.00	48.76	11.76	1.00	59.82	13.61
15	1.00	31.57	7.86	1.00	48.33	11.98	1.00	59.95	14.31
16	1.00	31.57	8.44	1.00	47.89	11.87	1.00	59.92	14.92

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	63.66	1.00	1.00	63.99	1.00	1.00	64.0	1.00
2	1.00	63.67	2.00	1.00	63.99	2.00	1.00	64.0	2.00
3	0.98	63.66	3.26	0.99	63.99	3.13	0.98	64.0	3.26
4	0.54	63.67	9.44	0.64	63.99	8.32	0.57	64.0	9.15
5	0.10	63.67	14.81	0.14	63.99	14.46	0.12	64.0	14.67
6	0.04	63.68	15.54	0.06	63.99	15.40	0.03	64.0	15.70
7	0.00	63.68	15.92	0.00	63.99	15.99	0.00	64.0	16.00
8	0.00	63.76	15.97	0.00	64.00	16.00	0.01	64.0	15.92
9	0.00	63.71	15.97	0.00	64.00	16.00	0.00	64.0	16.00
10	0.00	63.71	15.94	0.00	64.00	16.00	0.00	64.0	16.00
11	0.00	63.74	15.96	0.00	64.00	16.00	0.00	64.0	16.00
12	0.00	63.73	15.90	0.00	63.99	16.00	0.00	64.0	16.00
13	0.00	63.76	15.94	0.00	63.99	16.00	0.00	64.0	16.00
14	0.41	63.78	15.13	0.00	63.99	16.00	0.00	64.0	16.00
15	1.00	63.75	14.95	0.08	63.99	15.92	0.02	64.0	15.98
16	1.00	63.80	15.95	1.00	64.00	16.00	1.00	64.0	16.00

в упрощенном MBA-выражении больше 10. Этот порог выбран на основании результатов работы [8], где экспериментально установлено, что при числе чередований 10 и более время работы SMT-решателей превышает 3 000 секунд. Следовательно, по выделенным ячейкам можно определить размер кольца $\mathbb{V}$ и количество переменных t, для которых время анализа SMT-решателем случайного линейного MBA-выражения (с заданным синдромом) будет в среднем не менее 3 000 секунд. Например, если для тождественно истинного предиката P_T на рис. 1 нелинейные MBA-вы-

Table 7. Finding the low weight vector for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t \in \{2, 3, 4, 5\}$, $N = 4 \cdot 2^t$, $r \in \{1, \ldots, 2^t\}$, number of iterations 1000 of the algorithm ISDMinSearch, number of randomly chosen vectors — 100.

Таблица 7. Поиск вектора малого веса для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t \in \{2, 3, 4, 5\}$, $N = 4 \cdot 2^t$, $r \in \{1, \ldots, 2^t\}$, количество итераций 1000 алгоритма ISDMinSearch, количество случайно выбираемых векторов — 100.

t	$\mathbb{Z}_2$		$\mathbb{Z}_{2^2}$		$\mathbb{Z}_{2^4}$		$\mathbb{Z}_{2^8}$		$\mathbb{Z}_{2^{16}}$		$\mathbb{Z}_{2^{32}}$	
	wt(y)	wt(z)	wt(y)	wt(z)	wt(y)	wt(z)	wt(y)	wt(z)	wt(y)	wt(z)	wt(y)	wt(z)
2	7.39	0.93	11.29	1.79	14.18	2.91	14.92	3.84	15.0	4.00	15.0	4.0
3	16.41	1.99	24.16	3.23	30.00	5.24	31.86	6.91	32.0	7.98	32.0	8.0
4	31.81	3.47	47.99	6.74	60.01	**11.44**	63.74	**14.59**	64.0	**15.94**	64.0	**16.0**
5	61.30	**11.00**	94.20	**19.50**	120.10	**26.70**	127.70	**30.80**	128.0	**32.00**	128.0	**32.0**

Table 8. Dependence of the average weight of the vector returned by the algorithm ISDMinSearch on the number of iterations for 10 randomly chosen vectors $\mathbf{v}$ of the weight N.

Таблица 8. Зависимость среднего веса возвращаемого алгоритмом ISDMinSearch вектора от количества итераций для 10 случайно выбранных векторов $\mathbf{v}$ веса N.

$\mathbb{V}$	I $(t = 2)$				I $(t = 3)$				I $(t = 4)$				I $(t = 5)$			
	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4
$\mathbb{Z}_2$	1.1	1.0	1.0	1.0	3.4	1.8	1.8	1.8	6.4	4.0	3.6	3.6	15.6	11.2	7.4	6.4
$\mathbb{Z}_{2^2}$	2.1	1.7	1.7	1.7	6.0	3.5	3.1	3.1	11.0	8.4	6.8	5.8	21.6	17.6	15.8	15.0
$\mathbb{Z}_{2^4}$	3.4	3.0	2.9	2.9	7.1	5.8	5.3	4.8	14.5	12.7	11.4	10.3	29.2	26.4	24.6	23.4
$\mathbb{Z}_{2^8}$	4.0	3.9	3.8	3.8	7.9	7.4	6.9	6.7	15.6	15.1	14.6	14.0	31.8	31.0	30.0	29.6
$\mathbb{Z}_{2^{16}}$	4.0	4.0	4.0	4.0	10.4	8.0	8.0	7.8	20.8	16.0	16.0	15.5	32.0	32.0	32.0	32.0
$\mathbb{Z}_{2^{32}}$	4.0	4.0	4.0	4.0	12.8	8.0	8.0	8.0	16.0	16.0	16.0	16.0	32.0	32.0	32.0	32.0

Table 9. Dependence of the average weight of the vector returned by the algorithm ISDMinSearch on N for 10 randomly chosen vectors $\mathbf{v}$ of the weight N.

Таблица 9. Зависимость среднего веса возвращаемого алгоритмом ISDMinSearch вектора от N для 10 случайно выбранных векторов $\mathbf{v}$ веса N.

	I $(N = 16)$				I $(N = 32)$				I $(N = 48)$				I $(N = 64)$			
	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4
$\mathbb{Z}_2$	4.0	2.9	2.9	2.9	3.4	1.8	1.8	1.8	3.5	1.9	1.8	1.8	3.6	1.9	1.9	1.9
$\mathbb{Z}_{2^2}$	5.4	4.0	3.9	3.9	6.0	3.5	3.1	3.1	4.8	3.6	3.1	3.1	4.6	3.3	3.0	3.0
$\mathbb{Z}_{2^4}$	6.5	5.8	5.6	5.6	7.1	5.8	5.3	4.8	11.2	5.9	5.0	4.7	12.8	5.8	5.0	4.4
$\mathbb{Z}_{2^8}$	8.8	7.7	7.0	6.9	7.9	7.4	6.9	6.7	7.9	7.6	6.8	6.7	7.9	7.4	7.0	6.5
$\mathbb{Z}_{2^{16}}$	8.0	7.9	7.9	7.9	10.4	8.0	8.0	7.8	8.0	8.0	8.0	7.9	19.2	8.0	8.0	8.0
$\mathbb{Z}_{2^{32}}$	8.0	8.0	8.0	8.0	12.8	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0

ражения $E_1(x_1, \ldots, x_t)$ и $E_2(x_1, \ldots, x_t)$ содержат линейные MBA-подвыражения, выбранные случайно и равновероятно из некоторых фактор-классов $\mathbb{V}^N/C_F$, то при $t \geqslant 4$ и $\mathbb{V} \in \{\mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ можно ожидать, что время решения SMT-модулем предиката P_T будет не менее 3000 секунд. Подчеркнем, что выражения $E_1(x_1, \ldots, x_t)$ и $E_2(x_1, \ldots, x_t)$ следует выбирать нелинейными, иначе, согласно [4], тождественную истинность или ложность предикатов на основе этих выражений можно установить менее, чем за секунду, сравнивая соответствующие им подписи ($\mathbb{V}$-синдромы).

Результаты эксперимента, целью которого была оценка зависимости среднего веса возвращаемого вектора от числа итераций I, показаны в таблице 8. Анализ этих результатов показывает, что для колец $\mathbb{Z}_2$, $\mathbb{Z}_{2^2}$ и $\mathbb{Z}_{2^4}$ при $I = 1000$ наблюдается замедление в падении среднего веса возвращаемого вектора. Именно, в этом случае разница в среднем весе для $I = 1000$ и $I = 10000$ при всех рассмотренных значениях t не превышает 1.2. Представляется, что значения I больше 10000 могут с несущественной вероятностью уменьшить вес возвращаемого вектора ценой существенно боль-

шей вычислительной сложности. Для колец $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$ замедление в падении среднего веса возвращаемого вектора наблюдается уже при $I = 100$.

Из таблицы 9 видно, что при $I = 10^4$ для колец $\mathbb{Z}_2$, $\mathbb{Z}_{2^2}$, $\mathbb{Z}_{2^4}$ и $\mathbb{Z}_{2^8}$ средний вес возвращаемого вектора несущественно уменьшается с ростом N, когда входной вектор алгоритма ISDMinSearch выбирается случайно и равновероятно из $\mathbb{V}^N$; для колец $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$ с ростом I средний вес для всех рассмотренных N колеблется около $2^t = 8$. Можно предположить, что при деобфускации случайных линейных MBA-выражений (не только при $t = 3$) для небольших колец значение N следует выбирать побольше, чтобы уменьшить число слагаемых в MBA-выражении. Однако в этом случае возрастает вычислительная сложность алгоритма ISDMinSearch, так как в нем приходится находить обратные $(N - 2^t) \times (N - 2^t)$-матрицы, что требует порядка $O((N - 2^t)^3)$ операций.

Для всех рассмотренных t при $r = 1$ для колец малой мощности средний вес вектора, возвращаемого алгоритмом ISD, меньше единицы. Это означает, что в ряде случаев вес возвращаемого вектора равен нулю. Откуда следует, что некоторые обфусцируемые в экспериментах с помощью алгоритмов LinMBAGen и LinMBAGenAnyLen MBA-выражения принадлежали $\mathcal{C}_F$. Поэтому, перед обфускацией MBA-выражения $E(x_1, \ldots, x_t)$ (например, при генерации множества непрозрачных предикатов) следует проверить, что $\mathcal{E}(E(x_1, \ldots, x_t)) \notin \mathcal{C}_F$. Более общей рекомендацией может являться сравнение веса вектора, возвращаемого алгоритмом ISDMinSearch с заданным весом. Эта проверка может быть выполнена с помощью алгоритма CheckMBAComplexity (см. алгоритм 6).

Algorithm 6. CheckMBAComplexity

Data: $E(x_1, \ldots, x_t)$, I_1, I_2, r, R
Result: $T \in \{0, 1\}$, $T = 0$ — вес вектора, возвращаемого ISDMinSearch, меньше r, $T = 1$ — вес не меньше r

```
1 i = 0, T = 1
2 i = i + 1
3 if i < I_2 then
      /* Попытаться упростить MBA-выражение за I_1 итераций с использованием R случайно
         выбранных добавочных столбцов                                              */
4     z = E(MBASimplify(E(x_1, ..., x_t), I_1, R))
      /* Проверить условие на вес вектора                                           */
5     if wt(z) ⩾ r then
          /* Для очередной проверки вес возвращенного вектора не менее r; перейти
             к следующей проверке (проверка с другой достроенной матрицей A_F)     */
6         go to 2
7     else
          /* Среди I_2 случайно достроенных матриц A_F нашлась, по крайней мере, одна,
             для которой алгоритм ISDMinSearch вернул MBA-выражение с менее чем r
             слагаемыми                                                             */
8         T = 0
9 return T
```

Заключение

При всех рассмотренных t и $N = O(2^{t+2})$ предлагаемый способ упрощения линейных MBA-выражений на основе техники декодирования по информационным совокупностям позволяет существенно сократить длину выражения в случае, когда вес вектора $\mathbf{z}$ в (12) не превышает 3. В случае $\mathrm{wt}(\mathbf{z}) > 3$ для колец $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$ средний вес возвращаемого вектора практически всегда равен 2^t.

В этом же случае количество итераций поиска может быть порядка $I = 100$, так как разумное увеличение числа итераций практически не увеличивает вероятность найти вектор меньшего веса. Для меньших колец при $I \geqslant 1000$ с большой вероятностью может быть получен вектор веса меньше 2^t. Заметим, что целью известных алгоритмов упрощения (см. [8] и [4]) является получение MBA-выражения длиной порядка 2^t. Таким образом, в рамках задачи деобфускации MBA-выражений (не только линейных), предлагаемый способ может быть применен как предварительный шаг перед применением метода из [9], в котором эквивалентность MBA-выражений проверяется с помощью SMT-решателей. А так как время решения зависит от количества чередований в MBA-выражении, то предлагаемый способ позволит сократить это время. Способ упрощения линейных MBA-выражений на основе декодирования по информационным совокупностям, как представляется, можно использовать не только при деобфускации кода, но и при его *оптимизации*. Кроме того, такая оптимизация может быть полезной в среде символьного исполнения KLEE, где также актуальна задача сокращения времени выполнения запросов к SMT-решателю.

References

[1] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, “On the (im)possibility of obfuscating programs”, in *Advances in Cryptology — CRYPTO 2001*, ser. Lecture Notes in Computer Science, Springer, vol. 2139, 2001, pp. 1–18.

[2] Y. Zhou, A. Main, Y. X. Gu, and H. Johnson, “Information hiding in software with mixed boolean-arithmetic transforms”, in *Information Security Applications. WISA 2007*, ser. Lecture Notes in Computer Science, Springer, vol. 4867, 2007, pp. 61–75.

[3] S. Gulwani, O. Polozov, and R. Singh, “Program synthesis”, *Foundations and Trends in Programming Languages*, vol. 4, no. 1-2, pp. 1–119, 2017.

[4] B. Reichenwallner and P. Meerwald-Stadler, “Efficient deobfuscation of linear mixed boolean-arithmetic expressions”, in *Proceedings of the 2022 ACM Workshop on Research on offensive and defensive techniques in the context of Man At The End (MATE) attacks*, 2022, pp. 19–28.

[5] L. Zobernig, “Mathematical aspects of program obfuscation”, Ph.D. dissertation, The University of Auckland, 2020. [Online]. Available: https://researchspace.auckland.ac.nz/handle/2292/53400.

[6] P. Garba and M. Favaro, “Saturn-software deobfuscation framework based on LLVM”, in *Proceedings of the 3rd ACM Workshop on Software Protection*, 2019, pp. 27–38.

[7] N. Eyrolles, “Obfuscation with mixed boolean-arithmetic expressions: Reconstruction, analysis and simplification tools”, Ph.D. dissertation, Université Paris-Saclay, 2017, 126 pp.

[8] D. Xu, B. Liu, W. Feng, J. Ming, Q. Zheng, J. Li, and Q. Yu, “Boosting SMT solver performance on mixed-bitwise-arithmetic expressions”, in *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021, pp. 651–664.

[9] B. Liu, J. Shen, J. Ming, Q. Zheng, J. Li, and D. Xu, “MBA-Blast: Unveiling and simplifying mixed boolean-arithmetic obfuscation”, 2021, pp. 1701–1718.

[10] E. Berlekamp, R. McEliece, and H. Van Tilborg, “On the inherent intractability of certain coding problems (corresp.)”, *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[11] E. Prange, "The use of information sets in decoding cyclic codes", *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962.

[12] C. Peters, "Information-set decoding for linear codes over $\mathbb{F}_q$", in *Post-Quantum Cryptography. PQCrypto 2010*, ser. Lecture Notes in Computer Science, Springer, vol. 6061, 2010, pp. 81–94.

[13] V. Weger, N. Gassner, and J. Rosenthal, *A survey on code-based cryptography*, 2022. arXiv: 2201.07119v3 [cs.CR].

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 2, 2023
journal homepage: www.mais-journal.ru

ALGORITHMS

Algorithms for asymptotic and numerical modeling of oscillatory modes in the simplest ring of generators with asymmetric nonlinearity

S. D. Glyzin[1], E. A. Marushkina[1]

DOI: 10.18255/1818-1015-2023-2-160-169

[1]P.G. Demidov Yaroslavl State University, 14 Sovetskaya str., Yaroslavl 150003, Russia.

MSC2020: 93A30, 34C15, 37M25
Research article
Full text in Russian

Received September 12, 2022
After revision November 14, 2022
Accepted November 16, 2022

A system of three ring-connected generators with asymmetric nonlinearity and special nonlinear coupling is considered. The investigated system simulates an electrical circuit of three identical generators. Each generator is an oscillatory circuit with a nonlinear element. The volt-ampere characteristic of this element has a *S*-shaped character. The nonlinear coupling between the generators is organized in such way that it has a transmission coefficient close to one in the forward direction and close to zero in the reverse direction. First, the problem of finding solutions branching from equilibrium states is studied by asymptotic methods. And then the initial system is investigated by numerical methods. The dependence of the system dynamics on the degree of asymmetry of cubic nonlinearity describing the characteristic of a nonlinear element is studied.

Keywords: modeling; autogenerator; asymptotics; oscillatory mode; bifurcation; numerical analysis

INFORMATION ABOUT THE AUTHORS

Sergey D. Glyzin	orcid.org/0000-0002-6403-4061. E-mail: glyzin.s@gmail.com Doctor, Professor, Head of Department.
Elena A. Marushkina (corresponding author)	orcid.org/0000-0001-9183-6484. E-mail: marushkina-ea@yandex.ru Ph.D., Associate Professor, Head of Department.

Funding: This work was supported by the Russian Science Foundation (project No. 21-71-30011), https://rscf.ru/en/project/21-71-30011/.

For citation: S. D. Glyzin and E. A. Marushkina, “Algorithms for asymptotic and numerical modeling of oscillatory modes in the simplest ring of generators with asymmetric nonlinearity”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 160-169, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 2, 2023
сайт журнала: www.mais-journal.ru

ALGORITHMS

Алгоритмы асимптотического и численного построения колебательных режимов в простейшем кольце генераторов с несимметричной нелинейностью

С. Д. Глызин[1], Е. А. Марушкина[1]

DOI: 10.18255/1818-1015-2023-2-160-169

[1]Ярославский государственный университет им. П.Г. Демидова, ул. Советская, 14, Ярославль, 150003, Россия.

УДК 517.9
Научная статья
Полный текст на русском языке

Получена 12 сентября 2022 г.
После доработки 14 ноября 2022 г.
Принята к публикации 16 ноября 2022 г.

Рассматривается система из трех связанных в кольцо генераторов с несимметричной нелинейностью и специальной нелинейной связью. Исследуемая система моделирует электрическую цепь, в которой каждый из трех идентичных генераторов представляет собой колебательный контур с нелинейным элементом. Вольт-амперная характеристика этого элемента имеет S-образный характер. Нелинейная связь между генераторами организована так, что имеет близкий к единичному коэффициент передачи в прямом направлении и близкий к нулевому в обратном. Асимптотическими методами сначала изучается задача о решениях, ветвящихся от состояний равновесия, а затем численными методами исследуется исходная система. Изучена зависимость динамики системы от степени несимметричности кубической нелинейности, описывающей характеристику нелинейного элемента.

Ключевые слова: моделирование; автогенератор; асимптотика; колебательный режим; бифуркация; численный анализ

ИНФОРМАЦИЯ ОБ АВТОРАХ

Сергей Дмитриевич Глызин — orcid.org/0000-0002-6403-4061. E-mail: glyzin.s@gmail.com
зав. кафедрой компьютерных сетей, д-р физ.-мат.наук, профессор.

Елена Александровна Марушкина (автор для корреспонденции) — orcid.org/0000-0001-9183-6484. E-mail: marushkina-ea@yandex.ru
зав. кафедрой общей математики, канд. физ.-мат. наук, доцент.

Финансирование: Исследование выполнено за счет гранта Российского научного фонда № 21-71-30011, https://rscf.ru/project/21-71-30011/.

Для цитирования: S. D. Glyzin and E. A. Marushkina, “Algorithms for asymptotic and numerical modeling of oscillatory modes in the simplest ring of generators with asymmetric nonlinearity”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 160-169, 2023.

Введение

Разработка нелинейных систем электрических генераторов часто связана с необходимостью выявления у этой системы значений параметров, при которых она допускает хаотические колебания. Одним из простейших объектов такого рода является система из трех генераторов, соединенных в кольцо. В настоящей работе рассмотрена система нелинейных автогенераторов, нелинейность каждого из элементов которой обусловлена наличием так называемого туннельного диода. Для вольт-амперной характеристики этого элемента выбирается несимметричная S-образная функция. В качестве характеристики несимметричности этой функции используется квадратичная нелинейность в ее представлении. Предполагается, что связь между элементами кольца является однонаправленной. В работе проведен численный эксперимент, который позволяет установить границы изменения параметра несимметричности, при которых система генерирует периодические и хаотические колебания.

1. Модельная система

Рассматривается нелинейная система, состоящая из трех идентичных автогенераторов с однонаправленной связью (рис. 1). В этой системе L, C, R — сосредоточенные индуктивность, емкость и активное сопротивление, а E — напряжение источника питания. Вольт-амперная характеристика нелинейного элемента — туннельного диода TD — имеет вид, представленный на рис. 2. Подробный вывод модельных уравнений для такой электрической цепи можно найти в [1—3]. Математическая модель парциального генератора строится на основе законов Ома и Кирхгофа, откуда получаем:

$$C\ddot{u} + \left[\frac{1}{R} + f'(u)\right]\dot{u} + \frac{1}{L}u = \frac{E}{L}, \tag{1}$$

где u — напряжение в контуре, $f(u)$ — нелинейная характеристика туннельного диода (см. рис. 2).

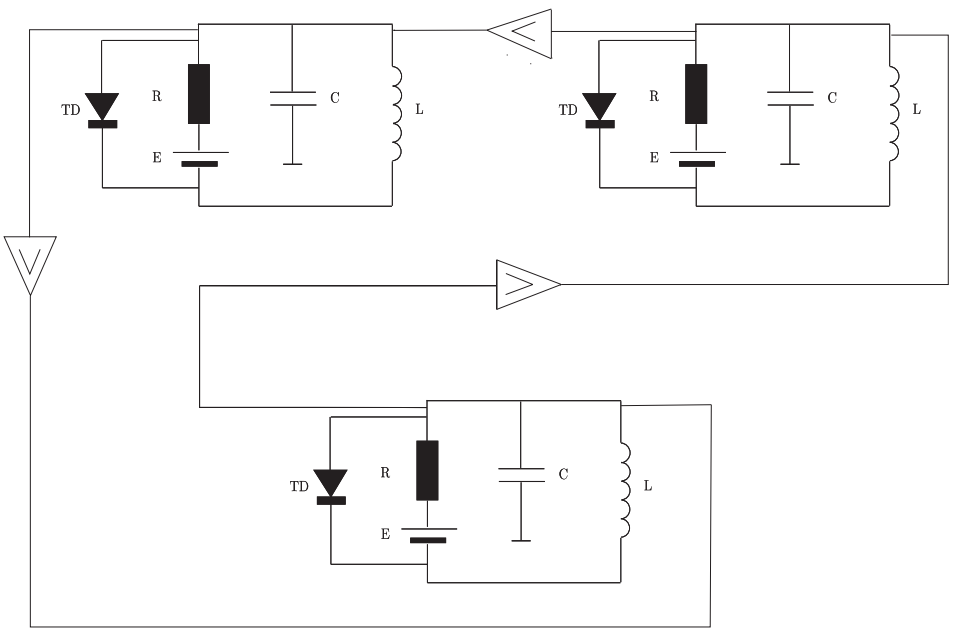


Fig. 1. A chain of three identical self-oscillators with a unidirectional connection

Рис. 1. Цепь из трех идентичных автогенераторов с однонаправленной связью

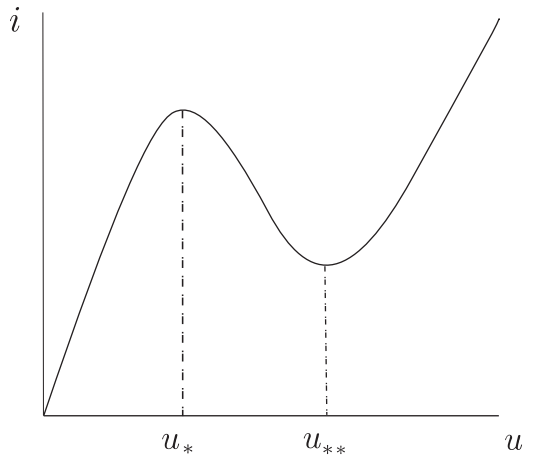


Fig. 2. Nonlinear characteristic of the tunnel diode

Рис. 2. Нелинейная характеристика туннельного диода

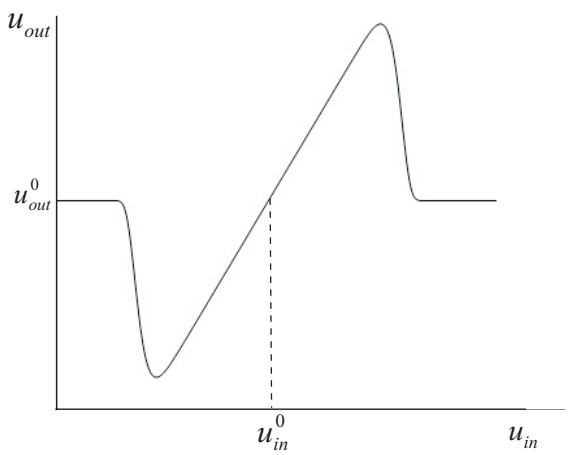


Fig. 3. Graph of the function $g(u)$

Рис. 3. График функции $g(u)$

Заметим, что парциальный осциллятор (1) имеет состояние равновесия $u = \tilde{u}_0$, где $\tilde{u}_0 = E$, которое является устойчивым (неустойчивым) при

$$\frac{1}{R} + f'(\tilde{u}_0) > 0 \quad (< 0). \tag{2}$$

Всюду ниже предполагаем, что рабочая точка $A = (\tilde{u}_0, \tilde{i}_0)$, $\tilde{i}_0 = f(\tilde{u}_0)$ лежит на падающем участке нелинейной характеристики $i = f(u)$, т. е. $f'(\tilde{u}_0) < 0$. В этом случае в силу (2) можно добиться устойчивости или неустойчивости положения равновесия $u = \tilde{u}_0$ за счет изменения только сопротивления R (при фиксированных прочих параметрах). Перейдем теперь к соответствующему кольцевому генератору, схема которого показана на рис. 1. Будем считать, что все три парциальные электрические цепи идентичны и связи между ними установлены с помощью трех одинаковых блоков. Важно отметить, что эти блоки носят принципиально нелинейный характер. А именно, каждый из них состоит из трех элементов: нелинейного преобразователя сигнала $u_{out} = g(u_{in})$, где u_{in}, u_{out} — входное и выходное напряжения соответственно, и двух буферных каскадов, реализованных на операционных усилителях. График функции $g(u)$ изображен на рис. 3. Отметим, что буферные устройства должны иметь большое входное и малое выходное сопротивления, а также обладать единичным коэффициентом передачи в прямом направлении и близким к нулевому в обратном. Практические способы реализации нелинейного блока, представленного на рис. 3, описаны в работах [4—7].

Для того чтобы вывести математическую модель интересующего нас кольцевого генератора учтем тот факт, что в силу однонаправленного характера связи парциальная система с номером j непосредственно влияет только на систему с номером $j + 1$ и, в свою очередь, сама испытывает влияние только со стороны $(j - 1)$-ой системы. Таким образом, для j-ой ячейки имеем равенства

$$i_1 = \frac{E - u_j}{R}, \quad i_3 = f(u_j), \quad i_4 = C\frac{du_j}{dt},$$
$$i_5 = -\frac{1}{L}\int \left(E - u_j - g(u_{j-1})\right) dt,$$
$$i_1 = i_2 + i_3, \quad i_2 = i_4 + i_5,$$

из которых после дифференцирования по t для парциальных напряжений u_j, $j = 1, 2, 3$ выходит система

$$C\ddot{u}_j + \left[\frac{1}{R} + f'(u_j)\right]\dot{u}_j + \frac{1}{L}\big(u_j + g(u_{j-1})\big) = \frac{E}{L}, \quad j = 1, 2, 3, \tag{3}$$

где $u_0 = u_3$.

Получившуюся систему (3) будем изучать при двух дополнительных ограничениях. Во-первых, напряжение E источников питания считаем таковым, что $E = u^0_{in} + u^0_{out}$, где $u^0_{out} = g(u^0_{in})$ (см. рис. 3). Во-вторых, предполагаем, что $u_* < u^0_{in} < u_{**}$, где u_*, u_{**} — экстремумы нелинейной характеристики туннельного диода (см. рис. 2). В этом случае система (3) имеет состояние равновесия $u_1 = u_2 = u_3 = \tilde{u}_0$, где $\tilde{u}_0 = u^0_{in}$.

Последовательность дальнейших действий такова. Сначала выполним в системе (3) замену $u_j - \tilde{u}_0 \longrightarrow u_j$, $j = 1, 2, 3$, переводящую упомянутое выше состояние равновесия в нулевое. Аппроксимация функции $f(u+\tilde{u}_0)$ обычно выполняется с помощью симметричного полинома третьей степени вида $f(u+\tilde{u}_0) = -s_0 u + s_2 u^3$, где s_0, $s_2 > 0$ (см. [1]). Представляет интерес задача о влиянии на динамику исследуемой системы квадратичного слагаемого. Ниже считаем, что $f(u + \tilde{u}_0) = -s_0 u + s_1 u^2 + s_2 u^3$. Нелинейную функцию связи $g(u+\tilde{u}_0) - u^0_{out}$ будем считать близкой к непрерывной кусочно-линейной функции. В результате после некоторых нормировок переменных u_j, $j = 1, 2, 3$, и времени интересующая нас система примет вид:

$$\ddot{u}_j + \frac{d}{dt}(\varepsilon u_j + \alpha u_j^2 + u_j^3) + u_j + \mu g(u_{j-1}) = 0, \quad j = 1, 2, 3, \tag{4}$$

где $u_0 = u_3$, $\mu > 0$, а знак параметров ε и α произволен. Что же касается функции связи $g(u)$, то теперь она задается равенством

$$g(u) = u \exp\Big(-\frac{u^n}{nb^n}\Big), \tag{5}$$

где $b > 0$ произвольно фиксировано. При увеличении параметра n функция (5) приближается к кусочно линейной функции, рассмотренной в работе [1].

Нетрудно видеть, что при достаточно малых значениях величин ε и $\mu = \nu\varepsilon$ (величину μ, как и ранее, полагаем положительной) к системе (4) применима локальная теория, изложенная в статье [1] (см. также [8—10]). При $\mu = \varepsilon = 0$ реализуется критический случай трех пар чисто мнимых корней. В связи с этим для нахождения нормальной формы системы (4) использовалась замена

$$u_j = \sqrt{\mu}u_{j1}(t, \tau) + \mu u_{j2}(t, \tau) + \mu^{3/2}u_{j3}(t, \tau) + \dots, \quad j = 1, 2, 3,$$

где $\tau = \mu t$, функции $u_{jk}(t, \tau)$ являются 2π−периодическими по t, причем

$$u_{j1} = z_j(\tau)\exp(it) + \bar{z}_j(\tau)\exp(-it), \quad j = 1, 2, 3.$$

Здесь комплексные амплитуды z_j подлежат определению в ходе выполнения алгоритма. Из условий разрешимости задач для $u_{j3}(t, \tau)$ в классе 2π−периодических по t функций на третьем шаге выполнения алгоритма получается следующая нормальная форма:

$$z_j' = -\nu\frac{z_j}{2} + \frac{i}{2}z_{j-1} + dz_j|z_j|^2, \quad j = 1, 2, 3, \tag{6}$$

где $d = -\frac{3}{2} + \frac{2\alpha^2}{3}i$. В статье [1] отыскиваются условия существования и устойчивости автомодельных циклов системы (6). Для данной работы важно отметить, что указанный результат получается и для случая $\alpha \neq 0$. Тем самым, при достаточно малых значениях параметров ε и μ показано, что система (4) с функцией связи вида (5) имеет орбитально асимптотически устойчивый цикл, ветвящийся от состояния равновесия. Это позволяет получить отправную точку для численного анализа в области, где эти значения не малы, и локальный асимптотический анализ не работает. Перейдем к обзору результатов численного эксперимента.

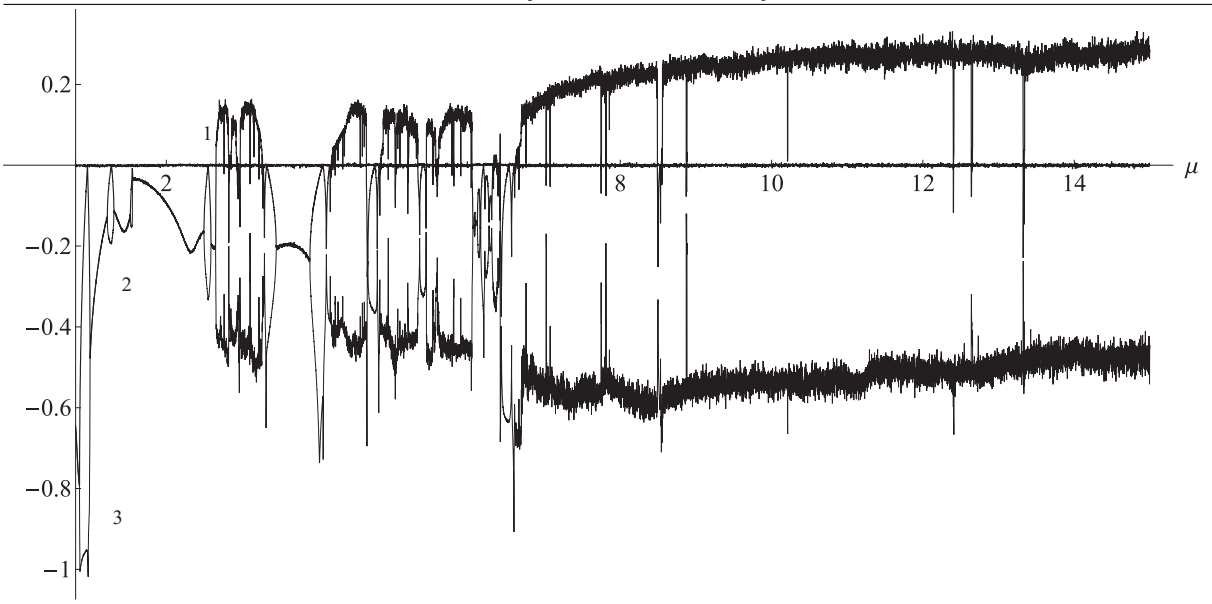


Fig. 4. Three largest Lyapunov exponents for $\alpha = 0$ and $0.8 \le \mu \le 15$

Рис. 4. Три старших ляпуновских показателя при $\alpha = 0$ и $0.8 \le \mu \le 15$

2. Численный эксперимент

Выполним численный анализ системы (4), (5) с целью нахождения ее нелокальных колебательных режимов в ситуации, когда

$$\varepsilon = 0.001, \quad b = 0.9. \tag{7}$$

Параметр α выберем трех различных видов: $\alpha = 0$ и $\alpha = \pm 0.5$. Параметр μ считаем бифуркационным и рассмотрим его изменение в пределах от 0.1 до 20.

Вместо системы (4) будем интегрировать эквивалентную систему первого порядка

$$\begin{aligned} \dot{u}_j &= v_j - \varepsilon u_j - \alpha u_j^2 - u_j^3, \\ \dot{v}_j &= -u_j - \mu g(u_{j-1}), \end{aligned} \qquad j = 1, 2, 3, \tag{8}$$

где $u_0 = u_3$. Выберем функцию связи $g(u)$ в соответствии с формулой (5) при $n = 6$.

Необходимо добавить, что при $\alpha = 0$ получившаяся система (8) инвариантна относительно замены $u_j \longrightarrow -u_j$, $v_j \longrightarrow -v_j$, $j = 1, 2, 3$, и поэтому все ее аттракторы в этом случае разбиваются на два класса: симметричные парные (переходящие друг в друга при указанной замене) и самосимметричные (остающиеся неизменными).

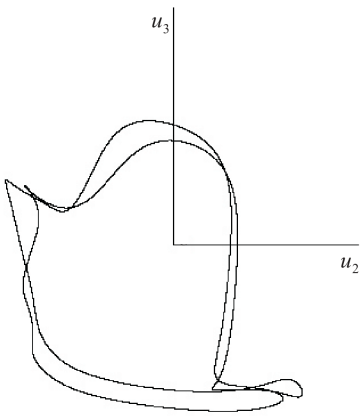
При условиях (7), $\alpha = 0$ и при увеличении μ в системе (8) наблюдаются следующие фрагменты динамики.

1) При $0.1 \le \mu < \mu_1$, где $\mu_1 \approx 0.82$, единственным ее аттрактором является устойчивый самосимметричный цикл. Добавим еще, что при достаточно малом μ существование такого цикла гарантирует изложенная в статье [1] локальная теория. Таким образом, локальный хаос с малой амплитудой колебаний здесь заведомо невозможен, поскольку единственным локальным аттрактором системы (8) является в этом случае устойчивый цикл.

2) При прохождении параметра μ через критическое значение μ_1 упомянутый выше цикл претерпевает бифуркацию типа вилки, т. е. становится неустойчивым и порождает два устойчивых симметричных цикла. Последние сохраняются на интервале (μ_1, μ_3), где $\mu_3 \approx 2.6108$, а при $\mu = \mu_3$ теряют устойчивость жестко.

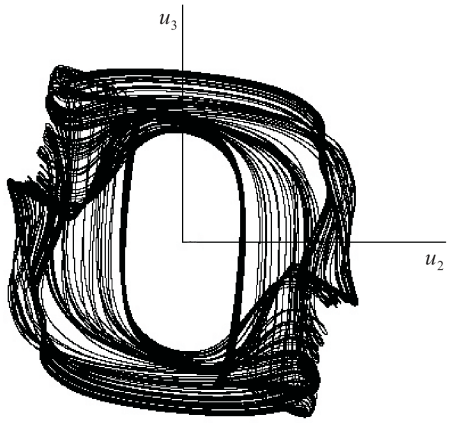
3) При $\mu = \mu_2$, где $\mu_2 \approx 2.61$, нелокально возникают два симметричных хаотических аттрактора и при $\mu_2 < \mu < \mu_3$ эти аттракторы сосуществуют с устойчивыми симметричными циклами, о которых говорилось чуть выше.

4) При $\mu \approx 2.697$ симметричные хаотические аттракторы объединяются в один самосимметричный.



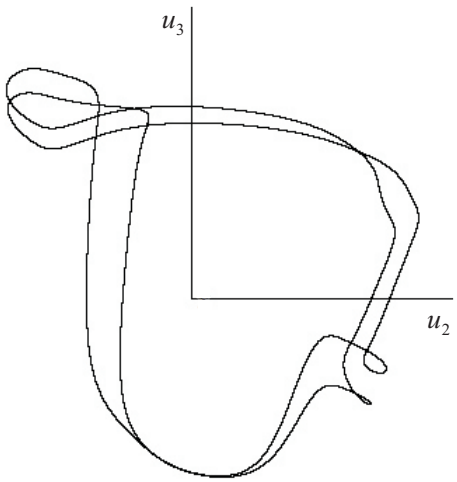
a)

Fig. 5. $\alpha = 0$ a) $\mu = 2.6$; b) $\mu = 2.8$, $\lambda_1 = 0.13$



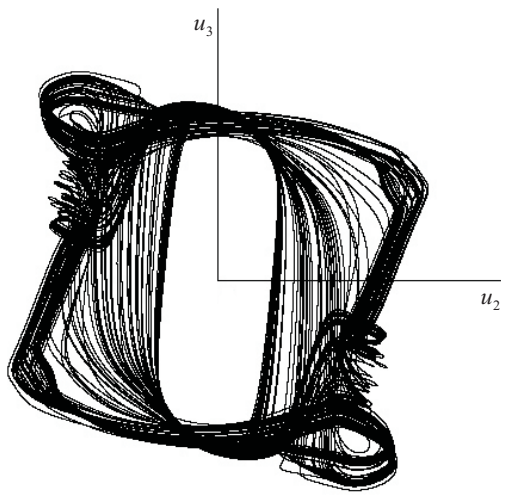
b)

Рис. 5. $\alpha = 0$ a) $\mu = 2.6$; b) $\mu = 2.8$, $\lambda_1 = 0.13$



a)

Fig. 6. $\alpha = 0$ a) $\mu = 3.8$; b) $\mu = 4.6$, $\lambda_1 = 0.14$



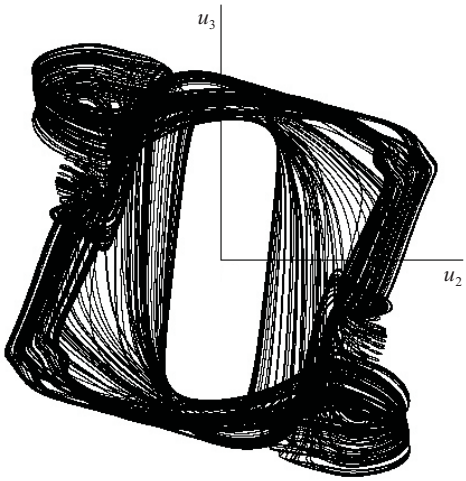
b)

Рис. 6. $\alpha = 0$ a) $\mu = 3.8$; b) $\mu = 4.6$, $\lambda_1 = 0.14$

При последующем увеличении μ не удается получить достоверного описания всех происходящих фазовых перестроек из-за наличия достаточно большого (возможно счетного) числа окон периодичности. Общее представление о динамике здесь дают лишь графики трех старших ляпуновских показателей $\lambda_1(\mu), \lambda_2(\mu), \lambda_3(\mu)$ аттрактора системы (8) (см. рис. 4), построенные на отрезке

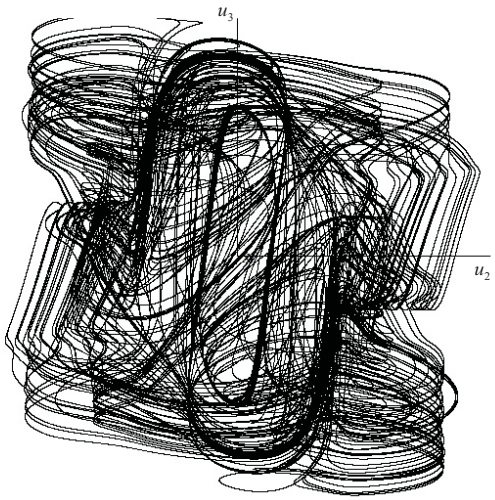
$$0.8 \le \mu \le 15 \tag{9}$$

по точкам с шагом h по μ, равным соответственно 0.001. При этом система (8) интегрировалась методом Рунге-Кутты четвертого порядка точности с постоянным шагом 0.0001 по времени (см. [11—



a)

Fig. 7. $\alpha = 0$ a) $\mu = 5.7$, $\lambda_1 = 0.12$; b) $\mu = 9.5$, $\lambda_1 = 0.245$



b)

Рис. 7. $\alpha = 0$ a) $\mu = 5.7$, $\lambda_1 = 0.12$; b) $\mu = 9.5$, $\lambda_1 = 0.245$

13]), а для оценки ляпуновских показателей использовался метод динамической перенормировки (см. [11]). Окна периодичности системы заканчиваются каскадами бифуркаций удвоения и переходом к хаосу. Для иллюстрации устойчивых режимов, возникающих в системе (8) на рисунках 5, 6 приведены проекции ее периодической и хаотической траекторий на плоскости (u_2, u_3) в окрестности правой границы ее первых двух окон периодичности.

В дополнение к приведенной на рис. 4, 5, 6 визуальной информации перечислим наиболее заметные окна периодичности, которые удалось обнаружить на отрезке $0.1 \le \mu \le 15$. К таковым относятся промежутки

$$[0.1, 2.61), \quad (2.853, 2.95), \quad (3.24, 4.582), \quad (4.67, 5.061),$$

$$(6.83, 6.96), \quad (12.235, 12.38), \quad (13.755, 13.84).$$

Общее же количество всех окон, по всей видимости, счетно. На рисунке 7 приведены проекции траекторий системы (8) в области развитого хаоса, в которой окна периодичности делаются достаточно узкими так, что их не удается различить с помощью численных методов.

Еще одна характерная особенность системы (8) заключается в том, что при некоторых μ хаотические аттракторы и устойчивые циклы в ней сосуществуют. Например, при $\mu = 4.6$ она имеет два симметричных хаотических аттрактора и три устойчивых цикла – два симметричных и один самосимметричный.

Рассмотрим теперь случай несимметричной характеристики нелинейного элемента сети и примем значение α равным 0.5. Как и в предыдущем случае для того, чтобы составить общее представление о динамике системы приводятся графики трех старших ляпуновских показателей $\lambda_1(\mu)$, $\lambda_2(\mu)$, $\lambda_3(\mu)$ аттрактора системы (8) (см. рис. 8, 9), вычисленные на отрезках

$$3.6 \le \mu \le 4.6, \quad 8 \le \mu \le 15 \tag{10}$$

В случае $\alpha = -0.5$ графики старших ляпуновских показателей построены на отрезке $2 \le \mu \le 15$ и представлены на рисунке 10.

Основное отличие случаев несимметричной характеристики нелинейного элемента от случая $\alpha = 0$ состоит в том, что пропадает инвариантность относительно замен $u_j \longrightarrow -u_j$, $v_j \longrightarrow -v_j$, $j = 1, 2, 3$,

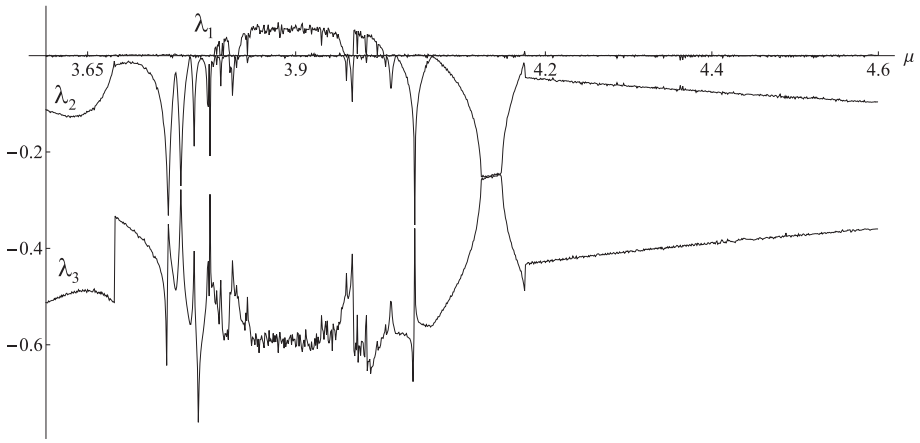


Fig. 8. Three largest Lyapunov exponents for $\alpha = 0.5$ and $3.6 \le \mu \le 4.6$

Рис. 8. Три старших ляпуновских показателя при $\alpha = 0.5$ и $3.6 \le \mu \le 4.6$

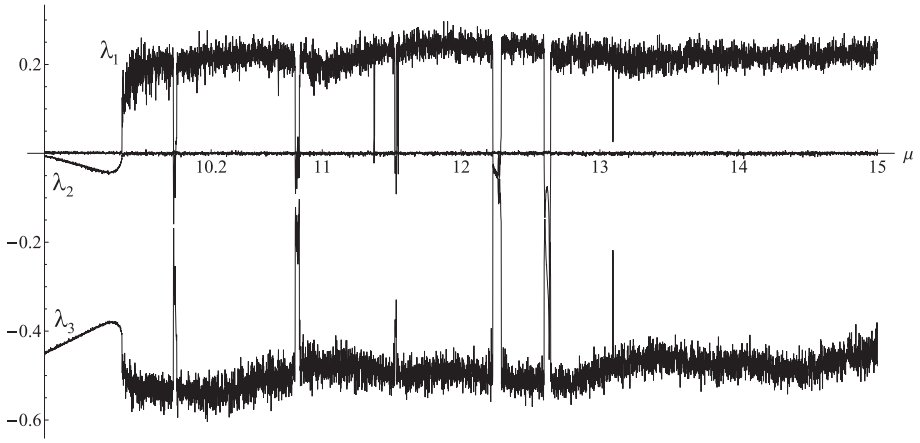


Fig. 9. Three largest Lyapunov exponents for $\alpha = 0.5$ and $8 \le \mu \le 15$

Рис. 9. Три старших ляпуновских показателя при $\alpha = 0.5$ и $8 \le \mu \le 15$

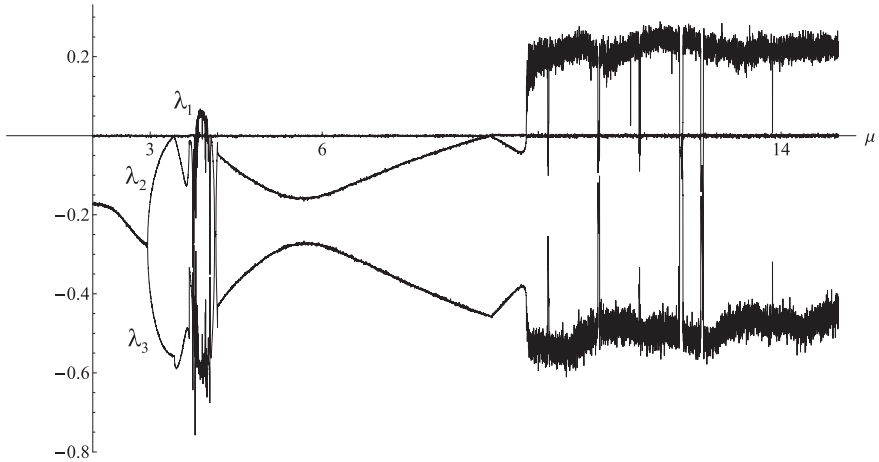


Fig. 10. Three largest Lyapunov exponents for $\alpha = -0.5$ and $2 \le \mu \le 15$

Рис. 10. Три старших ляпуновских показателя при $\alpha = -0.5$ и $2 \le \mu \le 15$

и, тем самым, отсутствуют симметричные друг другу или самосимметричные аттракторы. Следующее отличие касается размера области периодической динамики (8). Промежутки существования и устойчивости циклов этой задачи расширяются с ростом величины $|\alpha|$. Как в случае $\alpha = 0.5$, так и при $\alpha = -0.5$ имеется обширная область значений параметра μ, в которой система (8) имеет устойчивый цикл.

Заключительные замечания

На основе проделанного численного анализа, можно утверждать, что предложенный в статье [1] и развитый в данной работе способ создания хаотических генераторов достаточно универсален. Такая система может состоять из генератора периодических колебаний, описывающегося одним из уравнений (3). Нелинейный характер связи между парциальными системами в рамках локальной теории несущественен (нормальная форма остается прежней и в случае нелинейности (5) и в случае, когда в (3) функция связи линейна, т.е. $g(u) = u$). Численный анализ нелокальной динамики показал существенную зависимость наличия хаотических колебательных режимов от степени несимметричности характеристики нелинейного элемента (туннельного диода), а также важную роль нелинейной связи между парциальными системами.

References

[1] S. D. Glyzin, A. Y. Kolesov, and N. K. Rozov, “Chaos phenomena in a circle of three unidirectionally connected oscillators”, *Computational Mathematics and Mathematical Physics*, vol. 46, no. 10, pp. 1724–1736, 2006.

[2] S. D. Glyzin, A. Y. Kolesov, and N. K. Rozov, “The buffer phenomenon in ring-like chains of unidirectionally connected generators”, *Izvestiya: Mathematics*, vol. 78, no. 4, pp. 708–743, 2014.

[3] A. Y. Kolesov and N. K. Rozov, “Yavlenie bufernosti v RCLG-avtogeneratore: Teoreticheskij analiz i rezul’taty eksperimenta”, *Trudy MIAN*, vol. 233, pp. 153–207, 2001, in Russian.

[4] A. S. Dmitriev and V. Y. Kislov, *Chaotic oscillations in radiophysics and electronics*. Nauka, 1989, 280 pp., in Russian.

[5] A. S. Dmitriev, A. I. Panas, and S. O. Starkov, “Dinamicheskij haos kak paradigma sovremennyh sistem svyazi”, *Uspekhi sovremennoj radioelektroniki (Zarubezhnaya radioelektronika)*, no. 10, pp. 4–26, 1997, in Russian.

[6] A. S. Dmitriev and S. O. Starkov, “Peredacha soobshchenij s ispol’zovaniem haosa i klassicheskaya teoriya informacii”, *Uspekhi sovremennoj radioelektroniki (Zarubezhnaya radioelektronika)*, no. 11, pp. 4–32, 1998, in Russian.

[7] A. S. Dmitriev and A. I. Panas, *Dynamic chaos: novel type of information carrier for communication systems*. Fizmatlit, 2002, 252 pp., in Russian.

[8] B. D. Hassard, N. D. Kazarinoff, and Y.-H. Wan, *Theory and Applications of Hopf Bifurcation*. Cambridge University Press, 1985, 311 pp.

[9] V. V. Migulin, V. I. Medvedev, E. R. Mustel’, and V. N. Parygin, *Osnovy teorii kolebanij*. Nauka, 1988, 392 pp., in Russian.

[10] A. Y. Kolesov and N. K. Rozov, *Invariantnye tory nelinejnyh volnovyh uravnenij*. Fizmatlit, 2004, 408 pp., in Russian.

[11] D. S. Glyzin, S. D. Glyzin, A. Y. Kolesov, and N. K. Rozov, “The dynamic renormalization method for finding the maximum lyapunov exponent of a chaotic attractor”, *Differential Equations*, vol. 41, no. 2, 2005.

[12] S. P. Kuznecov, *Dinamicheskij haos: Kurs lekcij*. Fizmatlit, 2001, 296 pp., in Russian.

[13] J. Guckenheimer and P. Holmes, *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*. Springer Science & Business Media, 1983, 462 pp.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 2, 2023
journal homepage: www.mais-journal.ru

COMPUTING METHODOLOGIES AND APPLICATIONS

Signal Transition Graphs for Asynchronous Data Path Circuits

A. Kushnerov[1], S. Bystrov[2]

DOI: 10.18255/1818-1015-2023-2-170-186

[1]Independent researcher, Beer-Sheva, Israel.
[2]Independent researcher, Sochi, Russia.

MSC2020: 68W35
Research article
Full text in English

Received May 5, 2023
After revision May 29, 2023
Accepted May 31, 2023

The paper proposes a method for constructing signal transition graphs (STGs), which are directly mapped into asynchronous circuits for data processing. The advantage of the proposed method is that the resulting circuits are not only output-persistent, but also conformant to the environment. In other approaches, the environment is specified implicitly and/or inexactly and therefore they guarantee only output persistence. The conformation can be verified if both the circuit and its environment are specified by STGs. As an example, we consider a module realizing the function AND2. This module can either wait for both 1s or evaluate the function as soon as at least one 0 arrives. For each case, we draw up a separate STG (scenario) and map it into NCL gates. To provide such a mapping, we specify the behaviors of NCL gates by STG protocols. For data path, such an STG always contains alternative branches with the so-called garbage transitions at the gate inputs. The garbage transitions on a certain wire mean that the circuit is sensitive to the delay in this wire. Ignoring the garbage may lead to a violation of conformation or/and output persistence. For example, in the combinational part of the NCL circuits, the garbage appears on the inputs of NCL gates, and therefore these circuits are not delay insensitive.

Keywords: arithmetic; conformation; decomposition; delay in wires; handshake; pipeline; verification; weak causality

INFORMATION ABOUT THE AUTHORS

Alex Kushnerov | orcid.org/0000-0003-3953-1995. E-mail: kushnero@gmail.com
Independent researcher.

Sergey Bystrov
corresponding author | orcid.org/0009-0008-6525-0517. E-mail: bsa1969@yandex.ru
Independent researcher.

For citation: A. Kushnerov and S. Bystrov, “Signal transition graphs for asynchronous data path circuits”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 170-186, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 2, 2023
сайт журнала: www.mais-journal.ru

COMPUTING METHODOLOGIES AND APPLICATIONS

Графы сигнальных переходов для схем асинхронного тракта данных

А. Кушнеров[1], С. Быстров[2]

DOI: 10.18255/1818-1015-2023-2-170-186

[1]Независимый исследователь, Беэр-Шева, Израиль.
[2]Независимый исследователь, Сочи, Россия.

УДК 004.312.44
Научная статья
Полный текст на английском языке

Получена 5 мая 2023 г.
После доработки 29 мая 2023 г.
Принята к публикации 31 мая 2023 г.

В статье предлагается метод построения графов сигнальных переходов (STG), которые напрямую отображаются в схемы асинхронной обработки данных. Преимуществом предлагаемого метода является то, что полученные схемы не только неизменны по выходу (output-persistent), но и конформны внешней среде. В других подходах среда задаётся неявно и/или неточно, и поэтому они гарантируют только неизменность по выходу. Конформность можно проверить, если как схема, так и её внешняя среда заданы STG. В качестве примера мы рассматриваем модуль, реализующий функцию 2И. Этот модуль может либо ожидать лог. 1 на обоих входах, либо вычислить функцию, как только придёт хотя бы один 0. Для каждого случая мы составляем отдельный STG (сценарий) и отображаем его в элементы NCL. Чтобы обеспечить такое отображение, мы задаём поведение NCL элементов STG протоколами. Для тракта данных такой STG всегда содержит альтернативные ветви с так называемыми мусорными переключениями на входах элементов. Мусорные переключения на определенном проводе означают, что схема чувствительна к задержке в этом проводе. Игнорирование мусора может привести к нарушению конформности и/или неизменности по выходу. Например, в комбинационной части NCL схем мусор появляется на входах NCL элементов, поэтому эти схемы чувствительны к задержкам.

Ключевые слова: арифметика; верификация; декомпозиция; задержка в проводах; конформность; пайплайн; слабая причинность; хэндшейк

ИНФОРМАЦИЯ ОБ АВТОРАХ

Александр Кушнеров | orcid.org/0000-0003-3953-1995. E-mail: kushnero@gmail.com

Сергей Быстров (автор для корреспонденции) | orcid.org/0009-0008-6525-0517. E-mail: bsa1969@yandex.ru

Для цитирования: A. Kushnerov and S. Bystrov, “Signal transition graphs for asynchronous data path circuits”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 170-186, 2023.

Introduction

Asynchronous circuits do not use clock to ensure the validity of signals and operate in the mode of request-acknowledge. Like any digital circuits, they are built from logic gates. An output of a logic gate can be either in a stable or in an excited state. A stable state corresponds to the value of the Boolean function of the gate. An excited state is opposite to the value of this function. From an excited state, the gate can either switch to a new stable state or return to the previous one. The effect of returning to the previous stable state is called a hazard[1]. The goal of asynchronous design is hazard-free circuits. Input signals can also be in an excited state, but they produced by the environment, which is not realized by a circuit. Thus, to model the environment, it is more natural to use not Boolean functions, but something else. In this paper we use the event-based model called *Signal Transition Graphs (STGs)*.

In this model, the signals can be input, internal and output. The signal is excited if all the conditions for its switching to a new state are met. Any signal can stay excited for an arbitrary, but finite time. For non-input signals, i. e. for the gates obtained from STG, this means that their delay can be unbounded, but finite. The most important property of STG is output persistence. This property means that non-input signals must switch from an excited to a new stable state. The excitation can be removed only from input signals and only by switching of other input signals. Output persistence guarantees that an STG is mapped into a hazard-free circuit with arbitrary gate delays. In terms of data path design, such circuits are called delay-insensitive circuits free from gate orphans [1]. If we have one STG for the circuit and another one for the environment, we can check if the circuit does exactly what the environment expects from it to do. In other words, the circuit interface must be conformant to the environment and vice versa. The concepts of output-persistence and conformation are considered in more detail in Section 1.

Traditional approaches to designing an asynchronous data path are algebraic. Often they convert the initial combinational logic into a hazard-free one using dual-rail encoding (Table 1). Thus, each initial variable and its inversion are considered as two new signals. These signals may switch independently, but must reset into a spacer (all 0s or all 1s). Such a discipline is called a 4-phase protocol and shown in Fig. 1.

Table 1. Dual-rail encoding

a	b	a1	a0	b1	b0
0	0	0	1	0	1
0	1	0	1	1	0
1	0	1	0	0	1
1	1	1	0	1	0
spacer		0	0	0	0

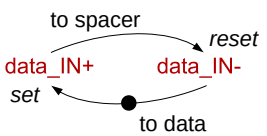


Fig. 1. A 4-phase protocol for input data

STGs are convenient for designing a control path, where the variables are single-rail. This encoding can be viewed as follows. The only value of a variable (e. g. the command “execute”) is encoded by switching a signal from 0 to 1. Switching the same signal from 1 to 0 can be interpreted as a spacer. Thus, the control is just a realization of functions in a unary (1-of-1) encoding. Let this encoding be the first in the sequence, then the second one is 1-of-2 (dual-rail), and the N-th is 1-of-N. We can use STGs for any of these encodings and thereby embed control into data. This means that we do not need to split the circuit into the control and data

[1]Traditionally, hazards are considered in terms of input changes and divided into functional and logic ones.

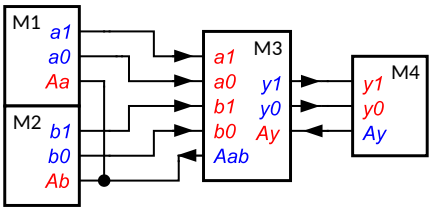


Fig. 2. Interface of the module M3 insensitive to delays in the wires

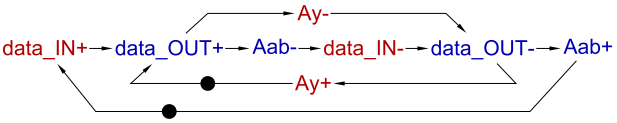


Fig. 3. Interface protocol for the module M3 in Fig. 2, which consists of input (the signals *data_IN*, *Aab*) and output (*data_OUT*, *Ay*) handshakes

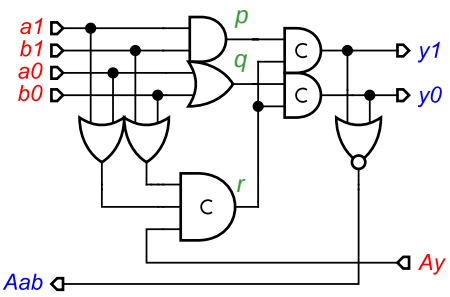


Fig. 4. Dual-rail AND circuit with embedded handshake

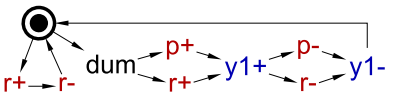


Fig. 5. Behavior of the C-element *y1* in Fig. 4. The transitions *r+*, *r-* in the short alternative branch are garbage that complicates decomposition

path. Moreover, the embedded control will allow us to build circuits according to the modular-hierarchical principle. In some sense, this principle is the opposite of RTL design. In particular, to synthesize an N-bit adder it is sufficient to construct an STG for a *single bit only*.

The 4-phase protocol in Fig. 1 is used only to exchange data. Supplementing it with a control signal, we obtain the handshake protocol, which allows us to organize the interaction between modules. Each handshake is realized by two variables as follows. The sender module sends a dual-rail variable (data) to the receiver module, which sends a unary control variable back to the sender. Let us consider the connection of modules shown in Fig. 2[2]. The module M3 receives the dual-rail variables a and b from the modules M1 and M2, and sends back an acknowledgement *Aab*. Theoretically, M1 and M2 may send data sequentially. In this case, delays in the wires may cause the sequential transitions to occur concurrently. However, the interface of each module is concurrent i. e. the module waits for input data to arrive in any order. Fig. 3 shows the interface protocol for the module M3. This is a generalized 4-phase protocol with two handshakes. The signal *data_IN* stands for a bus of input signals, which are switching concurrently. The signal *Ay* is also switching concurrently with any signal of *data_IN*. Thus, we have the handshake protocol and the concurrent interface[3]. This is what guarantees that the circuit will operate correctly with arbitrary delays of intermodule wires.

A module can realize algorithms to compute multiple functions. The modules can be connected arbitrary. The only restriction is that any ring must contain at least three modules [5]. Such a system may have some degree of concurrency and in a particular case, is a one-dimensional pipeline. The control in this system is entirely local (intermodule). No global control required.

An example of the realization of a simple module is shown in Fig. 4. This is a dual-rail AND circuit with embedded handshake. Note that the inputs of the 3-input C-element switch only once before the element fires. However, for the 2-input C-elements, this is not true. Fig. 5 shows the behavior of the C-element *y1*, which contain two alternative branches. The transition *r+* in the short branch does not cause *y1+*. In the circuit, this *r+* causes *y0+*, which initiates two concurrent processes. One is *Aab-* and reset of *a* and *b* into spacer, and the other is *Ay-*. Both of these processes are synchronized by *r-*. Thus, *r+* is eventually reset to *r-*. We will refer to such alternative branches and transitions as garbage.

[2]At the circuit level, such a structure was proposed first in [2] and formalized at the STG level in [3].

[3]The method proposed in [4] removes immediate relations between input transitions in an STG and makes them concurrent.

a) b)

Fig. 6. Decompositions of C-element verified under the environment in Fig. 5. Operating correctly (a) and incorrectly (b)

Fig. 6 shows two decompositions of the C-element [6], which have been verified under the environment in Fig. 5. The circuit in Fig. 6a is output-persistent and conformant. However, the circuit in Fig. 6b violates the conformation being still output-persistent[4].

Thus, the environment must be specified formally, otherwise we can get a wrong decomposition. In this paper, we propose a method that allows one to specify the behavior of an arbitrary module using STG. The obtained STG can be used for both verification and synthesis. To verify any previously designed circuit, this STG is used as an environment.

The synthesis is a mapping of STG into a library of logic gates. As shown in [5], the library of NCL gates is optimal. This was revealed in experiments with STGs, where the best results are obtained by "mirroring" the data phase into the spacer phase. In this case, for each alternative branch, considered separately, each signal is realized by a C-element. However, such signals can take place in different alternative branches. Hence, NCL gates are the simplest and natural mean to realize them.

1. Theoretical Background

a) Signal Transition Graphs (STGs) [8]. These graphs are used to specify the behavior of asynchronous circuits. An STG is a type of a labeled Petri net, where transitions are associated with the changes in the values of binary signals. For example, $x+$ means the switching of a signal x from 0 to 1, and $x-$ means a 1 to 0 switching. Input, internal and output signals are denoted and processed differently. The arcs in an STG capture the causal relations between the transitions. An STG may contain places with multiple incoming and outgoing arcs. If all arcs outgoing from a place, enter input transitions, such a place models a free (non-deterministic) choice made by the environment.

A signal transition is called excited if all entering it arcs, have tokens. An STG is called *output-persistent* [9] if the excitation is removed in a strictly defined way. For every non-input signal, the excitation is removed only by its firing. For every input signal, the excitation is removed either by its firing or by firing other input signal. The circuit can be converted to the so-called circuit Petri net [10], which itself is a type of STG. For verification, the circuit Petri net is combined (by parallel composition) with an STG that specifies the environment. This gives an STG of the closed system: the outputs of the circuit are the inputs for the environment STG and vice versa.

The circuit is called *conformant* if two conditions are met [10]. On one hand, the environment STG must provide only such transitions of the output signals, which the corresponding circuit Petri net can receive and still remain hazard-free. On the other hand, the circuit Petri net must provide only such transitions of the output signals, which the environment STG expects.

In this paper, we construct STGs by hand, so we need to make them readable. To this end, we use dummy transitions (*dum*) and proxy places. A dummy does not represent any real signal and is just a placeholder. A proxy place is a label for a regular place, from which an arc goes out and/or where it comes in. To verify and map STGs into circuits, as well as for verifying the obtained circuits, we use the Workcraft tool (http://workcraft.org). If an STG with dummies is used as an environment, Workcraft can verify the circuit for conformation. However, to verify the circuit or STG for output persistence, one needs to contract dummies.

[4]The problem of circuits not conformant to the environment was first considered by Izumi Kimura in [7].

At some point, we sacrifice formalities for readability and use dummies to specify weak (OR) causality. Namely, we specify a contradictory STG that violates *output determinacy* [11]. Such an STG cannot be mapped into a circuit, and if it is used as an environment, even the correct circuit will violate conformation. We propose a way around this obstacle.

b) NCL gates [12]. These gates are a special case of a generalized C (gC)-element. It is defined by the self-dependent expression $y(x, y) = S(x) \vee y\overline{R}(x)$, where $S(x)$ and $R(x)$ are set and reset functions. These functions must be orthogonal [9], that is, meet the condition $S(x)R(x) = 0$. Under this condition, the regular Boolean function $f(x, y) = S(x) \vee y\overline{R}(x)$ is monotone. A function $f(z_1, \dots, z_m)$ is called positive unate in variable z_i if $f(z_i = 1) \geqslant f(z_i = 0)$. A function positive unate in *all* variables, is called monotone. For NCL gates, the number of variables in $x = (x_1, \dots, x_n)$ is limited to $n \leqslant 4$. The NCL gates are used under the 4-phase protocol with zero spacer, hence their reset function $R(x) = \overline{(x_1 + \dots + x_n)}$, and the set functions $S(x)$ are different monotone functions.

2. Protocols for NCL gates

To guarantee output persistence, any logic gate must operate under a certain protocol. For NCL gates, we consider two types of protocols: with full and incomplete indication. The protocol with full indication realizes only *strong* (AND) causality. In this case, each transition is enabled to fire only after all of its immediate causes have fired. As shown in [5] (and outlined in Introduction), the protocols with full indication are realized best by NCL gates. In the case of *weak* (joint OR) causality [13] a transition is enabled to fire if at least one of its immediate causes has fired. Thus, there can be different variants of weak causality. The protocol with incomplete indication can realize both strong causality and all variants of weak causality.

a) Full indication. In protocols of this type each product term (conjunction) in the set function is given by an alternative branch, whose signals are synchronized in the set and reset phases. Fig. 7 shows the protocol with full indication and garbage branches for the NCL gate TH23w2 (the set function is *A+BC*). To verify this STG for output persistence, we need to contract the dummy. An equivalent STG without dummies can be obtained by translating the ProFlo expression [14]:

{B+;B-#C+;C-#A+;F+;A-;F-#(B+|C+);F+;(B-|C-);F-}

where the operators ';', '#'and '|'denote sequential composition, choice and concurrency respectively[5].

b) Incomplete indication. We can extend the protocol with full indication in different ways, each of which gives its own protocol with incomplete indication. For example, in the STG in Fig. 7 each alternative branch on the right may contain all the three input signals as shown in Fig. 8. This protocol realizes the only variant of weak causality possible for *A+BC*. In general case, each variant corresponds to a subset (from at least two conjunctions) of the set of conjunctions of the set function. For example, the set function *A+B+CD* contains four subsets: *A+B*, *A+CD*, *B+CD*, *A+B+CD*. Each of them represents a variant of weak causality.

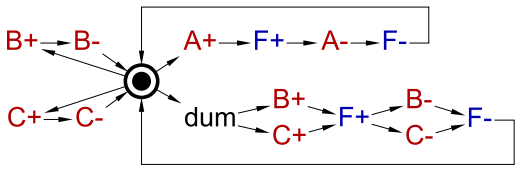


Fig. 7. Protocol with full indication and garbage branches for TH23w2 (A+BC)

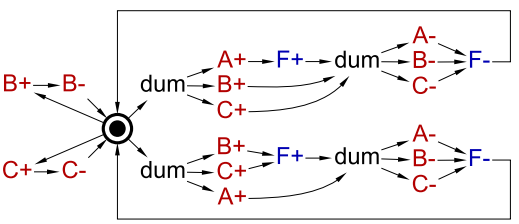


Fig. 8. Protocol with incomplete indication and garbage branches for TH23w2. Without timing constraints, the output determinacy is violated

[5]Currently, all signals in ProFlo are internal, i. e. we need to assign inputs and outputs. Otherwise, both output persistence and output determinacy are violated.

Let us consider the STG in Fig. 8 in more detail. The branches on the right contain the data phase and the spacer phase. In the data phase there are signals that are not indicated. In the upper branch this is *B+* and *C+*, and in the bottom one this is *A+*. In the spacer phase *A-*, *B-*, *C-* are indicated in both branches. On the other hand, in the upper branch *F+* is excited by *A+*, and in the bottom one — by both *B+* and *C+*. So, we have a contradiction that cannot be realized by the circuit and is a particular case of violation of output determinacy [11].

However, we presume that in the upper branch *A+* occurs earlier than either *B+* or *C+*, and in the bottom one *A+* occurs later than both *B+* and *C+*. These timing constraints are orthogonal and applied only to the input transitions. Hence, we can model them by interleaving [15]. Thus, the STG in Fig. 8 is converted into the output-determinate form. An equivalent interleaved STG without dummies is obtained by translating the ProFlo expression:

{B+;B-#C+;C-#(A+;(F+|(B+;C+))#A+;(F+|(C+;B+))#B+;A+;(F+|C+)#
B+;C+;(F+|A+)#C+;A+;(F+|B+)#C+;B+;(F+|A+));(A-|B-|C-)F-}

To guarantee output determinacy for both types of the protocols, we need to make sure that:

1) No set of immediate causes of the output signal in the data phase can be a strict subset of another set of immediate causes of the same signal in another branch (absorbed conjunction). For example, *A+BC+AB=A+BC* and therefore in Fig. 7 and Fig. 8 the branch, where *A+* and *B+* are the only immediate causes of *F+* is prohibited.

2) No garbage branch contains any set of immediate causes of the output signal in the data phase. In Fig. 7 and Fig. 8 the sets prohibited for the garbage branches are not only {A+}, {B+|C+}, but also all possible supplements: {A+|B+}, {A+|C+}, {A+|B+|C+}.

3. Proposed Method

The initial specification for the method is the truth tables of Boolean (or multiple-valued) functions[6]. Based on the truth tables, we construct an STG for the module and map this STG into a circuit. The protocols with full and incomplete indication are used as templates for the mapping. Let us demonstrate the method on the example of the AND function *y=ab*. The module realizing this function and its interface are shown in Fig. 2. According to the truth table, we represent the dual-rail variables taking the value "1", as shown in Table 2. Abbreviation "Sc." in this table means scenarios. We will introduce them later.

Table 2. Truth table of the AND function (a) and its dual-rail representation (b)

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

a)

Sc.	*as*	*bs*	*ys*
	a0	*b0*	*y0*
1.1	*a0*	*b1*	*y0*
	a1	*b0*	*y0*
1.2	*a1*	*b1*	*y1*

b)

The STG construction process consists of 6 steps (no iterations):

1. Analyze the truth table and elaborate a way to get functions. Since data is encoded by "+" transitions, it is sufficient to draw up an STG for the data phase only. This step largely determines the complexity of the resulting circuit.
2. Insert ("+") transitions of the input acknowledgement signals. According to the handshake protocol, they must precede the ("+") transitions of the functions. Since in the used NCL protocols the spacer phase is completely determined by the data phase, we can find the set functions. The number of variables in these functions must not exceed 4, otherwise decomposition is necessary.

[6]Multiple-valued (symbolic) STGs were introduced in [16]. To map such an STG into a circuit, one needs to take additional steps, which are not automated yet.

3. Decomposition. Insert ("+") transitions of internal signals before the transitions of those signals that are not realized by NCL gates. The goal is to reduce the amount of immediate causes for both the output and internal transitions.
4. Complement the STG with the spacer phase and then realize the output handshakes. The data phase is always mirrored into the spacer phase. In addition, in the protocols with incomplete indication, the signals non-indicated in the data phase, are indicated in the spacer phase. To realize the output handshakes, each "+" ("-") transition of the output signal must be an immediate cause of "-" ("+") transition of the corresponding input acknowledgement.
5. Provide the NCL protocols for all non-input signals. To this end, we need to establish new mediate relations between three sequential events: cause ["+" ("-") transition of non-input data], midterm event ["-" ("+") transition of output acknowledgement] and effect ["-" ("+") transition of input data]. In general case, these relations can be realized in several different ways. Finding the minimal realization is the classical set cover problem [5, 17]. However, there is a universal (but not always optimal) solution: to combine all the individual output acknowledgements into one, whose transitions are immediate effects of all output functions.
6. If the number of variables in the set function of the output acknowledgement exceeds 4, decompose it (as at Step 3).

From the obtained STG we can find the protocol under which any gate x operates. To this end, we need to convert all signals, except x and its immediate causes, to dummies. Contracting these dummies, we get the protocol with garbage transitions.

a) Variant 1. Using only strong causality.

Step 1. In the dual-rail representation in Table 2b *ys* depends on two variables *as* and *bs* that arrive concurrently. Although one of them can arrive earlier, we will not analyze this situation. Thus, *ys* waits for both *as* and *bs* to arrive, as shown in Fig. 9.

Step 2. The only option to insert the input acknowledgement *Ay+* is shown in Fig. 10. From this figure we find the set function *ys=as·bs·Ay*. Substituting here the variables from Table 2b, we split *ys* into *y0=Ay·(a0·b0+a0·b1+a1·b0)* and *y1=Ay·a1·b1*. Since *y0* depends on 5 variables, a decomposition is needed.

Step 3. Let us introduce Scenario 1.1 for *y0* and Scenario 1.2 for *y1*, as shown in Table 2b. Each scenario describes the behavior of the module for a certain set of input combinations. For different scenarios these sets do not overlap. To decompose *y0*, we insert *x+* into Scenario 1.1 as shown in Fig. 11. Since the NCL protocols are symmetric, any decomposition is output persistent. Now, the set functions are: *x=as·bs=a0·b0+a0·b1+a1·b0* and *y0=x·Ay*. For Scenario 1.2 the set function is the same *y1=Ay·a1·b1*.

Step 4. For the protocols with full indication, the spacer phase is a mirror reflection of the data phase. Thus, to realize the output handshakes, we establish immediate relations between *y0+*, *y1+ (y0-, y1-)* and *Ay- (Ay+)*, as shown in Fig. 12.

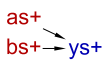


Fig. 9. STG obtained at Step 1

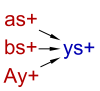


Fig. 10. STG obtained at Step 2

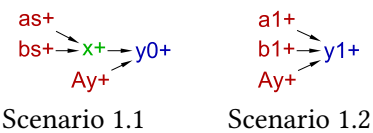


Fig. 11. STG obtained at Step 3

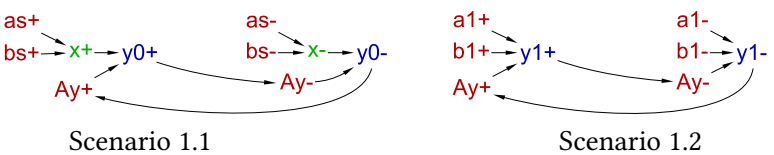


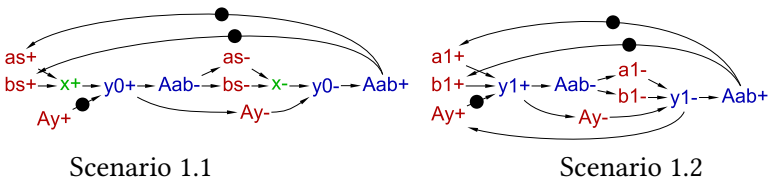
Fig. 12. STG obtained at Step 4

Table 3. All necessary mediate relations for Variant 1

Sc.	*cause*	*effect*
1.1	x+ or y0+	as-
		bs-
	y0+	as- or bs-
1.2	y1+	a1-
		b1-

Table 4. Minimal set of relations covering Table 3

Sc.	*cause*	*effect*
1.1	y0+	as-
		bs-
1.2	y1+	a1-
		b1-



Scenario 1.1 Scenario 1.2

Fig. 13. STG obtained at Step 5

Step 5. As at the previous step, to provide the NCL protocols for *x*, *y0*, *y1*, it is sufficient to establish relations between the data phase and the spacer phase. All the necessary mediate relations are shown Table 3. Minimizing the relations in Scenario 1.1, we obtain Table 4. Now, in Scenario 1.1 (1.2) the cause of *as- (a1-)* and *bs- (b1-)* is the same *y0+ (y1+)*. Therefore, all the relations for *a* and *b* must be realized via the same midterm transition *Aab-*.

Fig. 13 shows the obtained STG with mirrored new relations, which allow us to write *Aab=!(y0+y1)*.

Substituting into this STG the variables from Table 2b, we get the specification of strongly indicating AND circuit shown in Fig. 14.

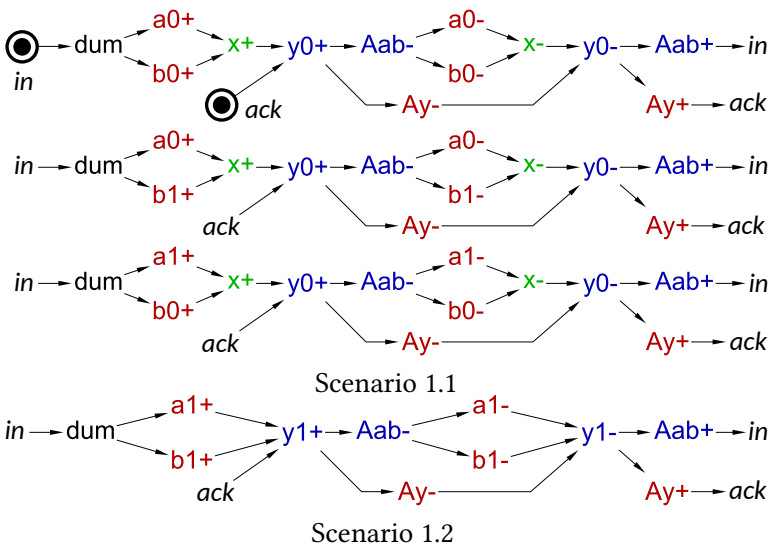


Fig. 14. STG specification of strongly indicating AND circuit with handshake

It is evident from Fig. 14 that the protocols for *x*, *y0*, *y1* are similar to the one shown in Fig. 7, i. e. are the protocols with full indication and garbage branches. We have already obtained all the equations of the gates, so the STG in Fig. 14 is mapped into the circuit shown in Fig. 15. To realize this circuit in the static CMOS, at least 46 transistors are required [15].

Let us now return to the circuit in Fig. 4 and try to embed the gates *p* and *q* into the C-elements *y1* and *y0*. To this end, we modify the STG in Fig. 14 as shown in Fig. 16. Let the signal *x* in this figure be the inversion of some signal *w*. This signal can be realized by the NCL gate TH24comp followed by a C-element.

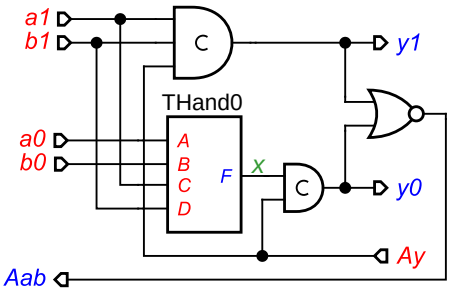


Fig. 15. AND circuit. Variant 1, obtained from the STG in Fig. 14

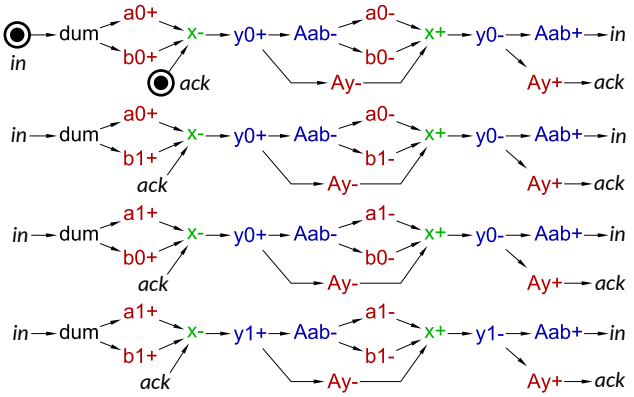


Fig. 16. Modification of the STG in Fig. 14

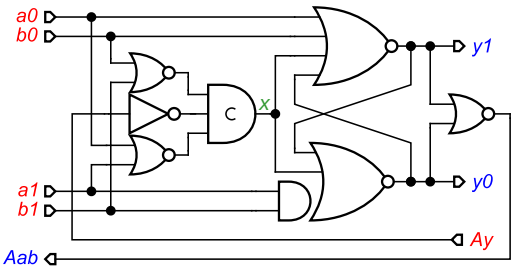


Fig. 17. SR-latch based variant of the circuit in Fig. 15

The set function of TH24comp is *S=a0·b0+a0·b1+a1·b0+a1·b1 or S=(a1+a0)·(b1+b0).* Hence, the signal *w* can be realized by the completion detector from the circuit in Fig. 4. The signals *y0*, *y1* in the STG in Fig. 16 are mapped into the SR-latch on complex gates, as shown in Fig. 17[7]. As compared to the circuit in Fig. 15, this circuit requires 42 transistors instead of 46.

b) Variant 2. Using both strong and weak causality (early propagation [19] or early evaluation [20]).

Step 1. From the dual-rail representation in Table 2b we can see that for *y0* it is sufficient to have either *a0* or *b0.* To describe this situation, we consider two cases. In the first case *a0* arrives earlier than *b0* and vice versa. This is Scenario 1.1. In the second case *a0 (b0)* switches concurrently with a don't-care term *b1 (a1).* This is Scenario 1.2. Let us denote in Scenario 1.1 the variable that arrives earlier by *N0*, and the variable that arrives later — by *Ms.* In Scenario 1.2 they arrive concurrently. For *y1* we need both *a1* and *b1.* This is Scenario 2. The variables encoded in each scenario are given in Table 5. The obtained STG is shown in Fig. 18.

Table 5. Scenarios and encoding for Variant 2

Sc.	*as*	*bs*	*N0*	*Ms*	*ys*
1.1	-	-	*a0*	*b0*	*y0*
	-	-	*b0*	*a0*	*y0*
1.2	-	-	*a0*	*b1*	*y0*
	-	-	*b0*	*a1*	*y0*
2	*a1*	*b1*	-	-	*y1*

Step 2. The only option to insert the transition *Ay+* of the input acknowledgement into both scenarios and follow the handshake protocol is shown in Fig. 19.

[7]Such circuits were first proposed in [18]. The principle of their operation is that data blocks that arm of the latch, which should remain in zero.

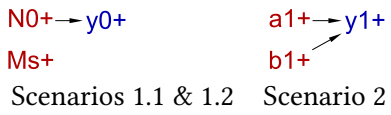


Fig. 18. STG obtained at Step 1

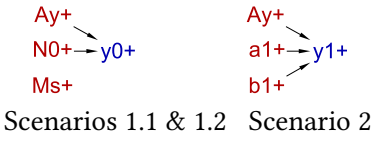


Fig. 19. STG obtained at Step 2

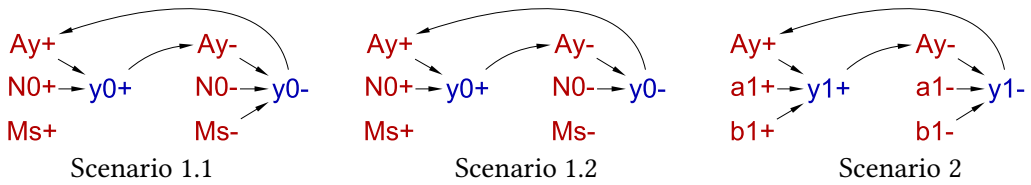


Fig. 20. STG obtained at Step 4

Substituting into Fig. 19 the variables from Table 5, we can write the set functions: *y0=Ay·N0=Ay·(a0+b0)* and *y1=Ay·a1·b1*. The number of variables in these functions does not exceed 4, therefore the decomposition (Step 3) is not needed.

Step 4. Scenario 1.1 realizes weak causality such that *Ms+* is not indicated by *y0+* in the data phase. In the spacer phase both of *N0-* and *Ms-* must be indicated by *y0-*. Scenarios 1.2 and 2 realize strong causality and therefore, the spacer phase is the mirror reflection of the data phase. To realize the output handshakes, we establish new immediate relations from *y0+ (y1+)* to *Ay-* and from *y0- (y1-)* to *Ay+* as shown in Fig. 20.

Step 5. To provide the NCL protocols for *y0* and *y1*, we establish the mediate relations given in Table 6.

Table 6. All necessary mediate relations for Variant 2

Sc.	*Com.*	*cause*	*effect*
1.1	*	y0+	N0- Ms-
	**	Ms+	N0- Ms-
1.2	*	y0+	N0-
	***	Ms+	Ms-
2	*	y1+	a1- b1-

* mirrored for spacer-data; ** only for data-spacer;
*** necessary only for sign alternation, mirrored for spacer-data.

The relations in this table cannot be minimized, so we establish them as is. Note that the effect of some relations is a nominal transition (*N0-* or *Ms-*) decoded as either *as-* or *bs-*. Hence, in such a relation, the midterm transition must be common for *a* and *b*. This is transition *Aab-*. Note also that in Fig. 20, *N0+* and *Ms+* can be decoded as the same real transition *a0+*. Therefore, we need to establish in Scenario 1.1 and 1.2 an additional relation between *N0+* and *Aab-*. The resulting STG is shown in Fig. 21.

From Fig. 21 we find the set function *Aab=N0·Ms·y0+a1·b1·y1* and decode *N0* and *Ms* using Table 5. As a result, we obtain *Aab=(a0·b0+a0·b1+a1·b0)·y0+a1·b1·y1*, which is a function of 6 variables and therefore must be decomposed.

Step 6. To decompose *Aab*, we insert three new internal signals: *a*, *b*, *y*. In Scenario 1.1 and 1.2 the signals *a* and *b* are encoded by nominal signals *N* and *M* as shown in Table 7. Fig. 22 shows the STG obtained at Step 6.

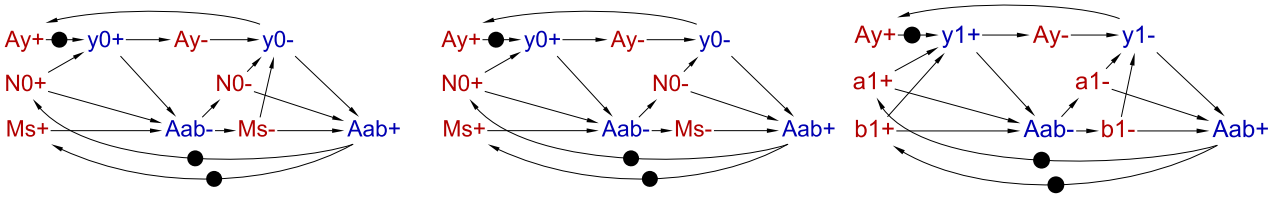


Fig. 21. STG obtained at Step 5

Table 7. Extending Table 5 by adding the columns N and M

Sc.	*as*	*bs*	*N0*	*Ms*	*N*	*M*	*ys*
1.1	-	-	*a0*	*b0*	*a*	*b*	*y0*
	-	-	*b0*	*a0*	*b*	*a*	*y0*
1.2	-	-	*a0*	*b1*	*a*	*b*	*y0*
	-	-	*b0*	*a1*	*b*	*a*	*y0*
2	*a1*	*b1*	-	-	-	-	*y1*

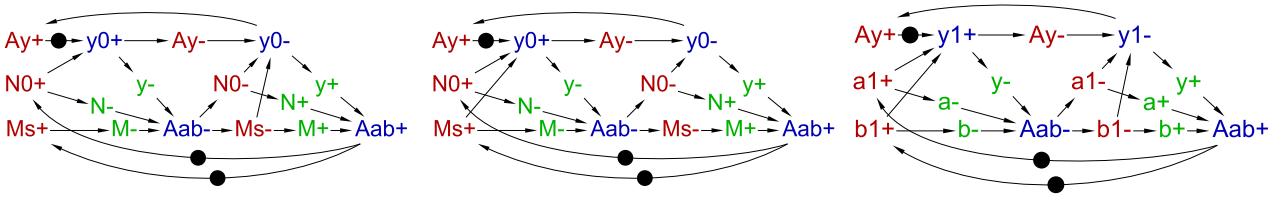


Fig. 22. STG obtained at Step 6

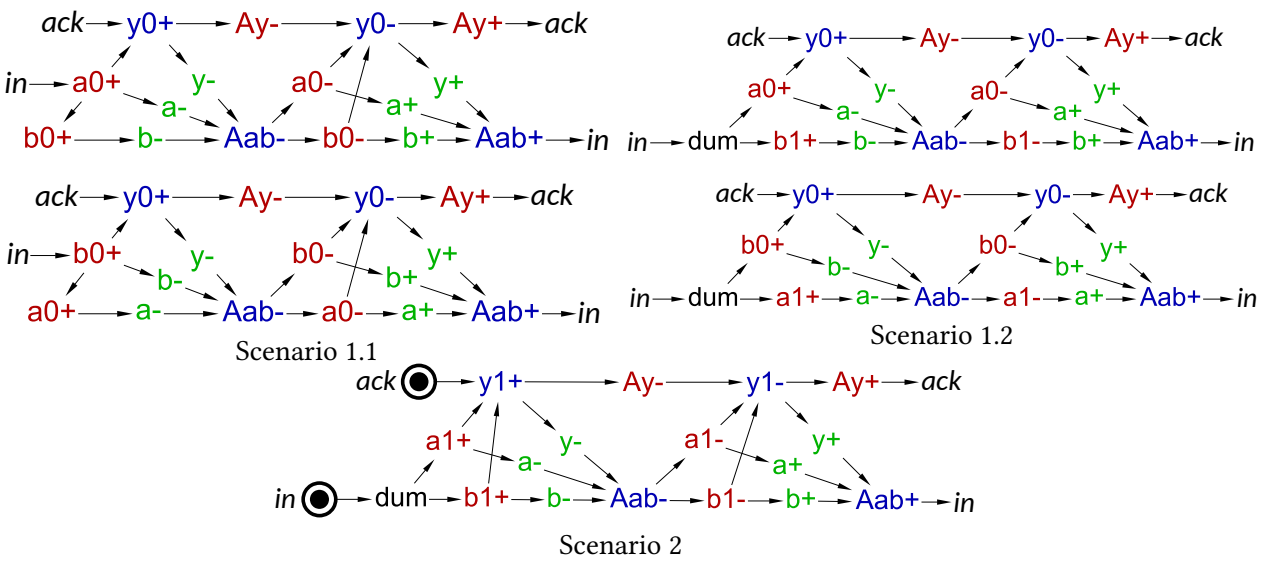


Fig. 23. STG specification of weakly indicating AND circuit with handshake

Substituting into this STG the variables from Table 7, we get the set function *Aab=a·b·y* and *a=!(a1+a0)*, *b=!(b1+b0)*, *y=!(y1+y0)*. In each equation, the number of variables does not exceed 4, so further decomposition is not needed. The above substitution gives the full STG of weakly indicating AND circuit shown in Fig. 23.

We have already obtained all the equations of the gates, so the STG in Fig. 23 is mapped into the circuit shown in Fig. 24. To realize this circuit in the static CMOS, at least 50 transistors are required [15].

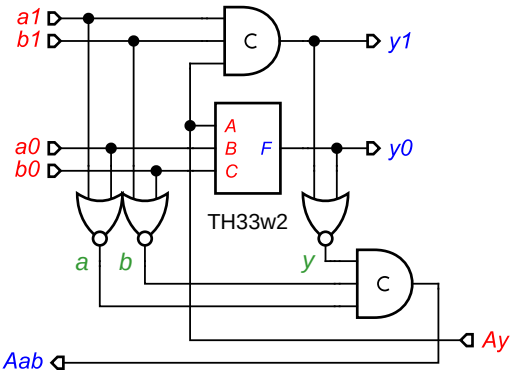


Fig. 24. AND circuit. Variant 2, obtained from the STG in Fig. 23

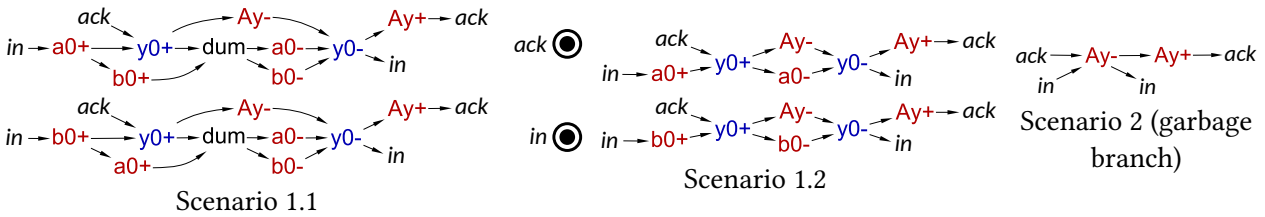


Fig. 25. Protocol with incomplete indication for *y0*

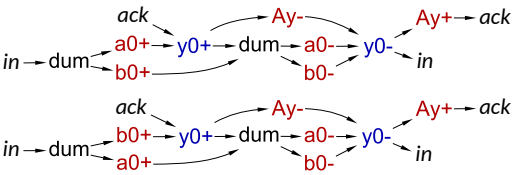


Fig. 26. Non-output-determine STG for Scenario 1.1 in Fig. 25

Let us now make sure that the behavior in Fig. 23 is mapped into the circuit in Fig. 24. To this end, we need to find a protocol under which every gate operates and compare it with the template for the corresponding NCL gate. As an example, we consider the signal *y0*, whose set function *y0=Ay·(a0+b0)*. To find the protocol for *y0*, we convert to dummies the transitions of those signals in Fig. 23, which are not causes for *y0*. Then, we contract extra dummies and obtain the STG shown in Fig. 25.

Let us consider Scenario 1.1 in Fig. 25. In the upper branch, *a0+* arrives earlier than *b0+*, and the bottom one — vice versa. Thus, we have timing constraints. Returning to the STG in Fig. 8, we specify Scenario 1.1 by non-output-determinate STG as shown in Fig. 26. The timing constraints make it output-determinate.

Thus, the obtained protocol for *y0* (Fig. 26 along with Scenarios 1.1 and 2 in Fig. 25) corresponds to the template in Fig. 8. In other words, the signal *y0* is indeed realized by the NCL gate TH33w2 as shown in the circuit in Fig. 24.

4. Related Works

Prior to this work, STGs had never been used to synthesize data path circuits at the level of gates. In terms of indication, early arithmetic circuits used only strong causality. The handshake in these circuits is realized on registers. We have shown that this is not necessary. In the so-called DIMS circuits [2, 21], dual-rail data signals are synchronized on multi-input C-elements, whose outputs are then collected by OR gates. Such circuits are two-level and therefore have large overhead in area. In contrast to DIMS, the so-called crossed implementation [18, 22] is a multilevel, purely combinational dual-rail circuit. To generate

a completion signal, all internal dual-rail signals in such a circuit are ORed and then collected by a tree of C-elements. Thus, the area occupied by the completion detector can be very large.

There is a variant of the crossed implementation [23] obtained by direct mapping of a Binary Decision Diagram (BDD). Note that transitions to spacer can be detected on the power rails, which in turn, can be used for attacks. On the other hand, logic layers in dual-rail circuits can have the opposite spacer. In particular, the corresponding variant of the cross-implementation [19] was designed especially for security purposes. To simplify the completion detection, one can use the so-called layer-wise optimization [24]. It should be taken into account that this optimization is based on relative timing, i. e. the circuit becomes sensitive to gate delays.

To optimize DIMS, it is necessary to find such C-elements that do not have garbage transitions at their inputs, as well as OR gates to which these C-elements are connected. The inputs without garbage, as well as all inputs of the corresponding OR gate, are inputs of the equivalent circuit. Its output is connected to the node where the output of the OR gate was connected. The equivalent circuit is similar to the completion detector with OR gates. Such an optimization is realized implicitly when a Multi-valued Decision Diagram (MDD) is mapped into a circuit [25].

Both DIMS and the crossed implementations can be optimized if the input signals of one gate are indicated at the output of some other gate. This principle defines the conditions of the so-called weak indication [26]. We used the same principle in the protocols with incomplete indication that realize both strong and weak causality. For these protocols, there are at least two different approaches to DIMS optimization [1, 27]. In the former approach, the optimization is deeper that allows one to convert DIMS into an NCL circuit. Then, the area or delay of this circuit is minimized. The crossed implementation for the above protocols can be optimized by a method based on solving Boolean equations [28].

A typical representative of circuits operating under the protocol with incomplete indication is the NCL full adder [29]. A simpler full adder circuit was realized on transistors [30]. However, in this circuit, the output carry rails do not return to zero. Instead, the previous data values are kept on the output capacitances. For these rails, we can either organize a 2-phase protocol or make the assumption that they are updated earlier than the sum. To design and optimize transistor circuits, similar to the one discussed above, there is a systematic approach [31].

Ad hoc algebraic techniques to embed the handshake into the NCL modules and thus to get rid of registers in the NCL pipelines are considered in [32–34]. The way to link these algebraic techniques with the presented STGs could be as follows. Let an STG is specified by the ProFlo expression and converted to a state transition diagram (using a finite state transducer). The operators of the ProFlo language on this diagram can represented by the operators of Tsirlin's algebra [22, 35].

Conclusion and Discussion

In this paper, the asynchronous data path is considered as a set of communicating dual-rail arithmetic circuits operating under the 4-phase protocol. We proposed a method for specifying such circuits by STGs. These STGs are correct by construction and are mapped into output-persistent circuits. To verify previously designed circuits, the obtained STGs can be used as an environment. The proposed method is based on the use of the protocols with full and incomplete indication. The latter is more complicated, but gives additional options for optimization. In both protocols, the data phase is mirrored into the spacer phase. If some signals are not indicated in the data phase, they are indicated in the spacer phase. This allows us to use the full potential of NCL gates.

The protocols with full indication use only strong causality. This is the 1st variant, which gives the circuits in Fig. 15 and in Fig. 17. The protocols with incomplete indication use both strong and weak causality. This is the 2nd variant, which gives the circuit in Fig. 24. This circuit requires 20 % more transistors (50 vs. 42) than the SR-latch based circuit (the 1st variant) in Fig. 17, but has a lower latency.

Note that the function AND2 is a special case of MAJ3 (carry in full adder). Moreover, the dual-rail MAJ3 is functionally complete, since we can invert the dual-rail signal by swapping the wires. A realization of MAJ3 on the SR-latch, whose arms are 5-input complex gates *!((a·b+c·(a+b))·d·e)* was proposed in [36].

Asynchronous circuits are sensitive to delays in some wires. To find such wires, we need to consider each gate separately and obtain the protocol under which it operates. In this protocol we distinguish between signals at the inputs and at the output of the gate. If all input transitions are indicated at the gate output, the corresponding input wires (from the gate to the fork) can have arbitrary delay. For each non-indicated transition there is a fork with unsafe wire. The same fork may correspond to a non-indicated transition in another protocol. We distinguish between two types of non-indicated transitions.

A non-indicated transition of the 1st type (for example, *x+*) precedes the opposite transition of the same signal (*x-*), which is followed by the output transition. A non-indicated transition of the 2nd type occurs concurrently with the output transition. For the considered protocols, this means that garbage transitions are of the 1st type, and weak causality generates transitions of the 2nd type. A circuit with unsafe wires retains all the properties (output-persistence, etc.), if the delays in these wires satisfy certain inequalities. Namely, the delay of each unsafe wire must be shorter than the delay of the corresponding adversary path [37]. Violation of this condition takes us beyond the STG model and leads to hazards. In terms of data path design, this situation is called wire orphan [1].

Acknowledgements

The authors would like to thank Dr. Victor Khomenko, who kindly answered our questions regarding the ProFlo language.

References

[1] C. Jeong and S. M. Nowick, “Optimization of robust asynchronous circuits by local input completeness relaxation”, in *IEEE Asia and South Pacific Design Automation Conference*, 2007, pp. 622–627.

[2] D. E. Muller, “Asynchronous logics and application to information processing”, *Switching Theory in Space Technology*, vol. 4, pp. 289–297, 1963.

[3] A. Yakovlev, “Designing self-timed systems”, *VLSI systems design*, no. 9, pp. 70–90, 1985.

[4] H. Saito, A. Kondratyev, J. Cortadella, L. Labagno, and A. Yakovlev, “What is the cost of delay insensitivity?”, in *IEEE/ACM International Conference on Computer-Aided Design*, 1999, pp. 316–323.

[5] S. Bystrov and A. Kushnerov, *Asynchronous data processing. Behavior analysis*, preprint, 2022. DOI: 10.13140/RG.2.2.14748.26248. [Online]. Available: https://www.researchgate.net/publication/362910934_Asynchronous_Data_Processing_Behavior_Analysis.

[6] A. Kushnerov, M. Medina, and A. Yakovlev, “Towards hazard-free multiplexer based implementation of self-timed circuits”, in *27th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2021, pp. 17–24.

[7] I. Kimura, “Extensions of asynchronous circuits and the delay problem. Part II: Spike-free extensions and the delay problem of the second kind”, *Journal of Computer and System Sciences*, vol. 5, no. 2, pp. 129–162, 1971.

[8] L. Rosenblum and A. V. Yakovlev, “Signal graphs: From self-timed to timed ones”, in *International Workshop on Timed Petri Nets*, IEEE, 1985, pp. 199–206.

[9] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, *Logic Synthesis for Asynchronous Controllers and Interfaces*. Springer Science & Business Media, 2002, 273 pp.

[10] I. Poliakov, A. Mokhov, A. Rafiev, D. Sokolov, and A. Yakovlev, “Automated verification of asynchronous circuits using circuit Petri nets”, in *14th IEEE International Symposium on Asynchronous Circuits and Systems*, 2008, pp. 161–170.

[11] V. Khomenko, M. Schaefer, and W. Vogler, "Output-determinacy and asynchronous circuit synthesis", *Fundamenta Informaticae*, vol. 88, no. 4, pp. 541–579, 2008.

[12] K. M. Fant, *Logically determined design: clockless system design with NULL convention logic*. John Wiley & Sons, 2005, 310 pp.

[13] A. Yakovlev, M. Kishinevsky, A. Kondratyev, L. Lavagno, and M. Pietkiewicz-Koutny, "On the models for asynchronous circuit behaviour with OR causality", *Formal Methods in System Design*, vol. 9, pp. 189–233, 1996.

[14] V. Khomenko, M. Koutny, and A. Yakovlev, "Slimming down Petri boxes: Compact Petri net models of control flows", in *33rd International Conference on Concurrency Theory (CONCUR 2022)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, vol. 243, 2022, 8:1–8:16.

[15] A. Kushnerov and S. Bystrov, *On minimal realization and behavior of NCL gates*, preprint, 2022. DOI: 10.13140/RG.2.2.31525.47847. [Online]. Available: https://www.researchgate.net/publication/363918158_On_Minimal_Realization_and_Behavior_of_NCL_Gates.

[16] A. Yakovlev and A. I. Petrov, "Symbolic signal transition graphs and asynchronous circuit design", Technical Report Series 395. Department of Computing Science, Tech. Rep., 1992, 40 pp.

[17] O. Coudert, "Two-level logic minimization: An overview", *Integration*, vol. 17, no. 2, pp. 97–140, 1994.

[18] V. I. Varshavsky, Ed., *Aperiodic Automata*. Nauka, 1976, 424 pp., in Russian.

[19] D. Sokolov, "Automated synthesis of asynchronous circuits using direct mapping for control and data paths", Ph.D. dissertation, University of Newcastle upon Tyne, 2006, 203 pp.

[20] J. Carmona, J. Cortadella, M. Kishinevsky, and A. Taubin, "Elastic circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1437–1455, 2009.

[21] D. Hammel, "Ideas of asynchronous feedback networks", in *Proceedings of the Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, IEEE, 1964, pp. 4–11.

[22] V. I. Varshavsky, Ed., *Self-timed control of concurrent processes. The design of aperiodic logical circuits in computers and discrete systems*. Kluwer Academic Publishers, 1990, 408 pp.

[23] T. Nanya, Y. Ueno, H. Kagotani, M. Kuwako, and A. Takamura, "TITAC: Design of a quasi-delay-insensitive microprocessor", *IEEE Design & Test of Computers*, vol. 11, no. 2, pp. 50–63, 1994.

[24] A. Mokhov, D. Sokolov, and A. Yakovlev, "Completion detection optimisation based on relative timing", in *Proceedings of the Eighteenth UK Asynchronous Forum*, 2006, pp. 73–76.

[25] B. Folco, V. Brégier, L. Fesquet, and M. Renaudin, "Technology mapping for area optimized quasi delay insensitive circuits", in *Vlsi-Soc: From Systems To Silicon. IFIP International Federation for Information Proc*, Springer, vol. 240, 2007, pp. 55–69.

[26] C. L. Seitz, "System timing", *Introduction to VLSI systems*, pp. 218–262, 1980.

[27] W. Toms and D. Edwards, "Prime indicants: A synthesis method for indicating combinational logic blocks", in *15th IEEE Symposium on Asynchronous Circuits and Systems*, 2009, pp. 139–150.

[28] L. P. Plekhanov, "Synthesis of self-timed combinational sections using the functional method", *Systems and Means of Informatics*, vol. 27, no. 2, pp. 85–97, 2017, in Russian.

[29] D. A. Duncan, G. E. Sobelman, and K. M. Fant, *Null convention adder*, US Patent 5,793,662, 1998.

[30] V. I. Varshavsky, A. Y. Kondratyev, V. A. Romanovsky, and B. S. Tsirlin, *Combinational adder*, USSR author's certificate SU1596321, 1988.

[31] Y. Zhou, "Automatic synthesis and optimisation of asynchronous data paths using partial acknowledgement", Ph.D. dissertation, University of Newcastle upon Tyne, 2008.

[32] S. C. Smith, "Design of an FPGA logic element for implementing asynchronous NULL convention logic circuits", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 6, pp. 672–683, 2007.

[33] M.-C. Chang, P.-H. Yang, and Z.-G. Pan, "Register-less NULL convention logic", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 3, pp. 314–318, 2016.

[34] M. Kim, "Null convention logic circuits for asynchronous computer architecture", Ph.D. dissertation, RMIT University, 2019, 197 pp.

[35] B. S. Tsirlin, "An algebra of asynchronous logic networks", *Cybernetics*, vol. 20, no. 1, pp. 23–29, 1984.

[36] A. I. Bukhshtab, V. I. Varshavsky, V. B. Marakhovsky, V. A. Peschansky, L. Y. Rosenblum, N. A. Starodubtsev, and B. S. Tsirlin, *Universal logic module*, USSR author's certificate SU561182, 1977.

[37] Y. Li, "Redressing timing issues for speed-independent circuits in deep sub-micron age", Ph.D. dissertation, University of Newcastle upon Tyne, 2012, 153 pp.