

Московский государственный университет
имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики
Кафедра математической кибернетики

Курсовая работа студента 518 группы
Гордеева Михаила Михайловича

Тема курсовой работы:
«О свойствах поляризованных полиномов
периодических функций k -значной логики»

Научный руководитель –
доцент, к.ф.-м.н. С. Н. Селезнева

Москва, 2016

Оглавление

Введение	3
Основные определения	5
Постановка задачи	10
Результаты	11
Заключение	16
Приложение	17
Программа на Sage	17
Программа на Haskell	19
Список литературы	22

Введение

Одним из стандартных способов задания функций k -значной логики являются поляризованные полиномиальные формы (ППФ), которые также называются обобщенными формами Риды-Мюллера, или каноническими поляризованными полиномами. В ППФ каждая переменная имеет определенную поляризацию. Длиной полиномиальной формы называется число попарно различных слагаемых в ней. Длиной функции f в классе ППФ называется наименьшая длина среди длин всех поляризованных полиномиальных форм, реализующих f . Функция Шеннона $L_k^K(n)$ длины определяется как наибольшая длина среди всех функций k -значной логики в классе K от n переменных, если K опущено, то подразумевается класс ППФ. Практическое применение ППФ нашли при построении программируемых логических матриц (ПЛМ) [1, 2], сложность ПЛМ напрямую зависит от длины ППФ, по которой она построена. Поэтому в ряде работ исследуется сложность ППФ различных функций [3–9].

В 1993 В. П. Супрун [3] получил первые оценки функции Шеннона для функций алгебры логики :

$$L_2(n) \geq C_n^{\lfloor \frac{n}{2} \rfloor},$$

$$L_2(n) < 3 \cdot 2^{n-1},$$

где $[a]$ обозначает целую часть a .

Точное значение функции Шеннона для функций алгебры логики в 1995 г. было найдено Н. А. Перязевым [4] :

$$L_2(n) = \left\lceil \frac{2^{n+1}}{3} \right\rceil.$$

Функции k -значных логик являются естественным обобщением функций алгебры логики. Для функций k -значной логики верхняя оценка функции Шеннона была получена в 2002 г. С. Н. Селезневой [5] :

$$L_k(n) < \frac{k(k-1)}{k(k-1)+1} k^n,$$

в 2015 году она была улучшена [12]

$$L_k(n) < \frac{k(k-1)-1}{k(k-1)} k^n.$$

При построении ПЛМ рассматривают и другие полиномиальные формы. Например класс обобщенных полиномиальных форм. В классе обобщенных полиномиальных форм, в отличие от класса поляризованных полиномиальных форм, переменные могут иметь различную поляризацию в разных слагаемых. В статье К. Д. Кириченко [6], опубликованной в 2005 г., был предложен метод построения обобщенных полиномиальных форм из которого следует $L_2^{\text{О.П.}}(n) = O(\frac{2^n}{n})$.

В работах [7, 10] получено, что $L_k^{\text{О.П.}}(n) = O(\frac{k^n}{n})$.

В 2012 г. Н. К. Маркеловым была получена нижняя оценка функции Шеннона для функции трехзначной логики в классе поляризованных полиномов [8]:

$$L_3(n) \geq \left\lceil \frac{3}{4} 3^n \right\rceil,$$

в [11] эта оценка была достигнута на симметрических функциях.

Основные определения

Пусть $k \geq 2$ – натуральное число, $E_k = \{0, 1, \dots, k-1\}$. Весом набора $\alpha = (a_1, \dots, a_n) \in E_k^n$ назовем число $|\alpha| = \sum_{i=1}^n a_i$. Моном $\prod_{a_i \neq 0} x_i^{a_i}$ назовем соответствующим набору $\alpha = (a_1, \dots, a_n) \in E_k^n$ и обозначим через K_α . По определению положим, что константа 1 соответствует набору из всех нулей. Функцией k -значной логики называется отображение $f^{(n)} : E_k^n \rightarrow E_k$, $n = 0, 1, \dots$. Множество всех функций k -значной логики обозначим через P_k , множество всех функций k -значной логики, зависящих от переменных x_1, \dots, x_n , обозначим через P_k^n . Функция $j_i(x) = \begin{cases} 1, & \text{если } x = i; \\ 0, & \text{если } x \neq i. \end{cases}$

Если k – простое число, то каждая функция k -значной логики $f(x_1, \dots, x_n)$ может быть однозначно задана формулой вида

$$f(x_1, \dots, x_n) = \sum_{\alpha \in E_k^n : c_f(\alpha) \neq 0} c_f(\alpha) K_\alpha,$$

где $c_f(\alpha) \in E_k$ – коэффициенты, $\alpha \in E_k^n$, и операции сложения и умножения рассматриваются по модулю k . Это представление функций k -значной логики называется ее полиномом по модулю k . При простых k однозначно определенный полином по модулю k для функции k -значной логики f будем обозначать через $P(f)$.

Определим поляризованные полиномиальные формы по модулю k . Поляризованной переменной x_i с поляризацией d , $d \in E_k$, назовем выражение вида $(x_i + d)$. Поляризованным мономом по вектору поляризации δ , $\delta = (d_1, \dots, d_n) \in E_k^n$, назовем произведение вида $(x_{i_1} + d_{i_1})^{m_1} \cdots (x_{i_r} + d_{i_r})^{m_r}$, где $1 \leq i_1 < \dots < i_r \leq n$, и $1 \leq m_1, \dots, m_r \leq k-1$. Обычный моном является мономом, поляризованным по вектору $\tilde{0} = (0, \dots, 0) \in E_k^n$.

Выражение вида $\sum_{i=1}^l c_i \cdot K_i$, где $c_i \in E_k \setminus \{0\}$ – коэффициенты, K_i – попарно различные мономы, поляризованные по вектору $\delta = (d_1, \dots, d_n) \in E_k^n$, $i = 1, \dots, l$, назовем поляризованной полиномиальной нормальной формой (ППФ) по вектору поляризации δ . Мы будем считать, что константа 0 является ППФ по произвольному вектору поляризации. Заметим, что при простых k для каждого вектора поляризации каждую функцию k -значной логики можно однозначно представить ППФ по этому вектору поляризации [5]. При простых k однозначно определенную ППФ по вектору поляризации $\delta \in E_k^n$ для функции $f \in P_k^n$ будем обозначать через $P^\delta(f)$.

Длиной $l(p)$ ППФ p назовем число попарно различных слагаемых в этой ППФ. Положим, что $l(0) = 0$. При простых k длиной функции k -значной логики в классе ППФ называется величина $l^{\text{ППФ}}(f) = \min_{\delta \in E_k^n} l(P^\delta(f))$.

Функция k -значной логики $f(x_1, \dots, x_n)$ называется симметрической, если

$$f(\pi(x_1), \dots, \pi(x_n)) = f(x_1, \dots, x_n)$$

для произвольной перестановки π на множестве переменных $\{x_1, \dots, x_n\}$. Множество всех симметрических функций k -значной логики обозначим через S_k . Симметрическая функция $f(x_1, \dots, x_n)$ называется периодической с периодом $\tau = (\tau_0 \tau_1 \dots \tau_{T-1}) \in E_k^T$, если $f(\alpha) = \tau_j$ при $|\alpha| = j \pmod{T}$ для каждого набора $\alpha \in E_k^n$. При этом число T называется длиной периода. Периодическую функцию k -значной логики $f(x_1, \dots, x_n)$ с периодом $\tau = (\tau_0 \tau_1 \dots \tau_{T-1}) \in E_k^T$ будем обозначать через $f_{(\tau_0 \tau_1 \dots \tau_{T-1})}^{(n)}$. Понятно, что такое обозначение полностью определяет эту функцию.

Введем функцию $\text{rol}_i(\alpha) \in E_k^n \times E_k \rightarrow E_k^n$, производящую циклический сдвиг вектора α влево. Пусть $\alpha = (a_1, \dots, a_n)$, тогда $\text{rol}_i(\alpha) = (a_{(1+i) \bmod T}, \dots, a_{(n+i) \bmod T})$, где T – длина вектора α .

Функция $\text{rol}_i(\alpha) \in E_k^n \times E_k \rightarrow E_k^n$ производит циклический сдвиг вектора α вправо. Пусть T – длина вектора α , тогда $\text{rol}_i(\alpha) = \text{rol}_{(-i) \bmod T}(\alpha)$.

Циклической матрицей, порожденной вектором $\beta = (b_0, b_1, \dots, b_{n-1})$, называется матрица вида

$$B = \begin{pmatrix} b_0 & b_1 & \dots & b_{n-1} \\ b_{n-1} & b_0 & \dots & b_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \dots & b_0 \end{pmatrix}.$$

Каждая строка, начиная со второй, получается циклическим сдвигом предыдущей вправо на один элемент.

В [9] дается описание быстрого алгоритма построения поляризованного полинома по заданному вектору функции и поляризации d . Данный алгоритм можно использовать для получения периодов, участвующих в разложении периодической симметрической функции $f^{(n+1)}$ с периодом τ .

$$A^{(d)} \cdot F(\tau) = \begin{pmatrix} \tau_0 \\ \tau_1 \\ \vdots \\ \tau_{k-1} \end{pmatrix}, \text{ где}$$

$$f_{\tau}^{(n+1)} = f_{\tau_{k-1}}^{(n)} (x_{n+1} + d)^{k-1} + f_{\tau_{k-2}}^{(n)} (x_{n+1} + d)^{k-2} + \dots + f_{\tau_0}^{(n)} \quad (1)$$

$$A = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & k-1 & -(2^{k-2}) \bmod k & \dots & -((k-1)^{k-2}) \bmod k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k-1 & k-1 & k-1 & \dots & k-1 \end{pmatrix}$$

$A^{(d)}$ получается из матрицы A циклическим сдвигом столбцов A влево на d столбцов или построчным применением функции rol_d к матрице A .

$$F(\tau) = \begin{pmatrix} rol_0(\tau) \\ rol_1(\tau) \\ \vdots \\ rol_{k-1}(\tau) \end{pmatrix}$$

$F(\tau)$ – матрица, в которой в i -той строке стоит сдвиг влево периода τ на i элементов.

Например в трехзначной логике для функции с периодом $[1,1,2,2]$ получим

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 \\ 2 & 2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & 0 & 2 & 0 \\ 2 & 1 & 1 & 2 \end{pmatrix}$$

Пусть матрица A состоит из строк a_i , a'_i – строка, полученная из a_i , добавлением нулей справа так, что длина строки a'_i равна T . Тогда матрица A_i состоит из T циклических сдвигов вправо строки a'_i :

$$A_i = \begin{pmatrix} ror_0(a'_i) \\ ror_1(a'_i) \\ \vdots \\ ror_{T-1}(a'_i) \end{pmatrix},$$

матрица A_i – циклическая.

Аналогичным образом из матрицы $A^{(d)}$ строятся матрицы $A_i^{(d)}$. Матрица A_0 – единичная матрица размера $T \times T$. Для всех τ_i из формулы (1) верно $\tau_i = A_i \cdot \tau$. Период τ называется вырожденным, если $l(f_\tau^{(n)}) = \bar{o}(k^n)$ при $n \rightarrow$

∞ . Период τ называется сложным, если $\exists n_0, c > 0 : \forall n \geq n_0 \Rightarrow l(f_\tau^{(n)}) \geq c \cdot k^n$.

Обозначим через \tilde{n} – константный период, вида $[n, n, \dots, n]$, где $n \in E_k$.

Введем индуктивно множество периодов, порожденных периодом τ за i шагов – $\Pi_i(\tau)$. В данном случае матрицы рассматриваются, как множества строк.

$$\Pi_0(\tau) = \{\tau\}$$

$$\Pi_1(\tau) = \bigcup_{d \in E_k} A^{(d)} \cdot F(\tau)$$

$$\Pi_{i+1}(\tau) = \bigcup_{\theta \in \Pi_i(\tau)} \Pi_1(\theta)$$

Обозначим через $\Pi(\tau)$ такое $\Pi_i(\tau)$, что $\Pi_i(\tau) = \Pi_{i+1}(\tau)$.

Постановка задачи

1. Нахождение периодов, являющихся сложными, в k -значных логиках.
2. Отыскание критериев сложности периодов.
3. Получение нижних оценок длин некоторых симметрических функций k -значных логик в классе ППФ.

Результаты

Теорема 1. Если $\tilde{0} \notin \Pi(\tau)$, то τ сложный период.

Доказательство. Теорема 1 является следствием теоремы 2. □

Теорема 2. Если $\tilde{0} \notin \Pi(\tau)$, то $\forall \sigma \in \Pi(\tau) \forall n \geq n_0 \Rightarrow l(f_\sigma^{(n)}) \geq c \cdot k^n$, где

T – длина периода τ

$k^{n_0} > T > k^{n_0-1}$, нетрудно заметить, что $n_0 = \lceil \log_k T \rceil$

$$c = \frac{1}{k^{n_0}}$$

Доказательство. Доказательство проведем индукцией по n .

Базис индукции:

$$l(f_\sigma^{(n_0)}) \geq 1 = \frac{1}{k^{n_0}} \cdot k^{n_0}$$

Индуктивный переход:

Предположим, что теорема верна для $n \geq n_0$, докажем ее для $n + 1$.

Для любой поляризации $d \in E_k$ функция $f_\sigma^{(n+1)}$ выражается следующим полиномом:

$$f_\sigma^{(n+1)} = f_{\sigma_{k-1}}^{(n)} \cdot (x_{n+1} + d)^{k-1} + \dots + f_{\sigma_1}^{(n)} \cdot (x_{n+1} + d) + f_{\sigma_0}^{(n)}, \text{ где}$$

функции $f_{\sigma_i}^{(n)}$ могут быть различными при разных поляризациях d . По предположению индукции $\forall i \in E_k \Rightarrow l(f_{\sigma_i}^{(n)}) \geq \frac{1}{k^{n_0}} \cdot k^n$, следовательно $l(f_\sigma^{(n+1)}) \geq k \cdot \frac{1}{k^{n_0}} \cdot k^n = \frac{1}{k^{n_0}} \cdot k^{n+1}$ □

Теорема 3. Все периоды длины 4 в трехзначной логике, отличные от $[0, 0, 0, 0]$, $[1, 0, 1, 0]$, $[2, 0, 2, 0]$, $[0, 1, 0, 1]$, $[1, 1, 1, 1]$, $[2, 1, 2, 1]$, $[0, 2, 0, 2]$, $[1, 2, 1, 2]$, $[2, 2, 2, 2]$ являются сложными.

Доказательство. Все матрицы $A_0^{(d)}$ являются невырожденными, поэтому из них нельзя получить нулевой вектор умножением на ненулевой.

Рассмотрим матрицы $A_i^{(d)}$, где $d \in [0..2]$, $i \in [1,2]$.

$$\begin{pmatrix} 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 2 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Базисным вектором ядра этой матрицы является вектор $[1, 1, 1, 1]$. Следовательно при умножении этой матрицы на вектор x будет нулевой вектор $\Leftrightarrow x \in \{[0, 0, 0, 0], [1, 1, 1, 1], [2, 2, 2, 2]\}$.

$$\begin{pmatrix} 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 \end{pmatrix}$$

Базисным вектором ядра этой матрицы является вектор $[1, 1, 1, 1]$. Следовательно при умножении этой матрицы на вектор x будет нулевой вектор $\Leftrightarrow x \in \{[0, 0, 0, 0], [1, 1, 1, 1], [2, 2, 2, 2]\}$.

$$\begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

Базисным вектором ядра этой матрицы является вектор $[1, 1, 1, 1]$. Следовательно при умножении этой матрицы на вектор x будет нулевой вектор $\Leftrightarrow x \in \{[0, 0, 0, 0], [1, 1, 1, 1], [2, 2, 2, 2]\}$.

$$\begin{pmatrix} 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 \end{pmatrix}$$

Базисным вектором ядра этой матрицы является вектор $[1, 1, 1, 1]$. Следовательно при умножении этой матрицы на вектор x будет нулевой вектор $\Leftrightarrow x \in \{[0, 0, 0, 0], [1, 1, 1, 1], [2, 2, 2, 2]\}$.

$$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{pmatrix}$$

Базисными векторами ядра этой матрицы являются векторы $[0, 1, 0, 1], [1, 0, 1, 0]$. Следовательно при умножении этой матрицы на вектор x будет нулевой вектор $\Leftrightarrow x \in \{[0, 0, 0, 0], [1, 0, 1, 0], [2, 0, 2, 0], [0, 1, 0, 1], [1, 1, 1, 1], [2, 1, 2, 1], [0, 2, 0, 2], [1, 2, 1, 2], [2, 2, 2, 2]\}$.

$$\begin{pmatrix} 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 \end{pmatrix}$$

Базисным вектором ядра этой матрицы является вектор $[1, 1, 1, 1]$. Следовательно при умножении этой матрицы на вектор x будет нулевой вектор $\Leftrightarrow x \in \{[0, 0, 0, 0], [1, 1, 1, 1], [2, 2, 2, 2]\}$. \square

Следующее утверждение было получено с использованием программы на Sage, листинг приведен на странице [17](#).

Утверждение. Все периоды длины γ в пятизначной логике, кроме константных, являются сложными.

Теорема 4. Произведение циклических матриц коммутативно.

Доказательство. Рассмотрим циклические матрицы размера $n \times n$ A и B , порожденные векторами $(a_0, a_1, \dots, a_{n-1})$ и $(b_0, b_1, \dots, b_{n-1})$ соответственно

$$A = \begin{pmatrix} a_0 & a_1 & \dots & a_{n-1} \\ a_{n-1} & a_0 & \dots & a_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \dots & a_0 \end{pmatrix}, B = \begin{pmatrix} b_0 & b_1 & \dots & b_{n-1} \\ b_{n-1} & b_0 & \dots & b_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \dots & b_0 \end{pmatrix}$$

и матрицы $C = A \cdot B$ и $C' = B \cdot A$. Докажем, что $c_{ij} = c'_{ij}$. Все операции сложения и вычитания в индексах будут производиться по модулю n .

Пусть $j = l - i$, где l – некоторое число из $[0..n - 1]$. Рассмотрим i строку матрицы A $(a_{-i}, a_{1-i}, \dots, a_{n-1-i})$ и j столбец матрицы B $(b_{l-i}, b_{l-1-i}, \dots, b_{l+1-i})$, а также i строку матрицы B $(b_{-i}, b_{1-i}, \dots, b_{n-1-i})$ и j столбец матрицы A $(a_{l-i}, a_{l-1-i}, \dots, a_{l+1-i})$. При вычислении c_{ij} a_{m-i} будет умножаться на b_{l-m-i} , $\forall m \in [0..n - 1]$, а при вычислении c'_{ij} $b_{(l-m)-i}$ будет умножаться на $a_{l-(l-m)-i} = a_{m-i}$. □

Теорема 5. Если $\tilde{0} \in \Pi_1(\tau)$, то τ вырожденный период.

Доказательство.

$$f_{\tau}^{(n+1)} = f_{\tau_{k-1}^{(d)}}^{(n)} \cdot (x_{n+1} + d)^{k-1} + \dots + f_{\tau_1^{(d)}}^{(n)} \cdot (x_{n+1} + d) + f_{\tau_0^{(d)}}^{(n)} \quad (2)$$

$\tau_i^{(d)} = A_i^{(d)} \cdot \tau$. Если существует $\tau_j^{(d')} = \tilde{0}$, то для любых периодов $\tau_i^{(d)} \neq 0$ тоже будет существовать нулевой период в разложении.

Обозначим $\tau_i^{(d)}$ за σ , получим для σ разложение (2). Тогда

$$\sigma_j^{(d')} = A_j^{(d')} \cdot \tau_i^{(d)} = A_j^{(d')} \cdot A_i^{(d)} \cdot \tau \underset{\text{по теореме 4}}{=} A_i^{(d)} \cdot A_j^{(d')} \cdot \tau = A_i^{(d)} \cdot \tilde{0} = \tilde{0}.$$

Это значит, что при любой поляризации будет существовать нулевой период в разложении. Следовательно $l(f_\tau^{(n)}) \leq (k-1)^n = \bar{o}(k^n)$.

□

После запуска программы на различных числовых интервалах была выдвинута следующая гипотеза. *Все неконстантные периоды лдины l в k -значной логике будут сложными \Leftrightarrow все простые делители $l > k$.*

Заключение

В работе получены следующие результаты:

1. Доказана теорема о нижней оценки длины функций, задаваемых сложными периодами.
2. Получено достаточное условие сложности периодов.
3. Написаны программы для работы с периодами в которых реализованы функции:
 - Проверка периода на сложность
 - Построение периодов, порожденных данным
 - Построение специальных матриц, описанных в основных определениях, и функций для работы с ними.
 - Вывод полиномов функций по векторам периодов,
 - Построение ядер, описанных матриц,
 - Проверка выдвинутой гипотезы.

Приложение

В приложении приводится код программ, использованных для получения некоторых результатов. Также программы сами по себе представляют интерес, так как могут быть использованы для более широкого ряда задач, чем требовалось при выполнении курсовой работы.

Программа на Sage

Используя систему компьютерной алгебры Sage [13] была написана программа для нахождения ядер матриц $A_i^{(d)}$ и для проверке гипотезы о зависимости сложности векторов от их длины.

```
def list_transpose(l):
    return [list(i) for i in zip(*l)]

def rol(l,n):
    n = n % len(l)
    return l[n:] + l[:n]

def ror(l,n):
    n = n % len(l)
    return l[-n:] + l[:-n]

def a(k, d = 0):
    def a(i,j):
        if (i == 0 and j == 0):
            return 1
        elif(i == k-1):
            return (-1) % k
        elif(i == 0 or j == 0):
            return 0
        else:
            return (-(j)^((k-1-i) % k)) % k
```

```

m = [[0 for i in range(k)] for j in range(k)]

for i in range(k):
    for j in range(k):
        m[i][j] = a(i,j)

m = list_transpose(rol(list_transpose(m), d))

return matrix(GF(k), m)
#return m

def a_i(k, i, l, d = 0):
    v = list(a(k,d)[i])
    v += [0] * (l - len(v))

    m = []
    for n in range(l):
        m.append(ror(v,n))

    return matrix(GF(k), m)
#return m

def to_matrix(k, l):
    return map(lambda m: matrix(GF(k), m), l)

def matrices(k, t):
    return to_matrix(k, [a_i(k,i,t,d) for d in range(k) for i in range(1,k)])

def all_matrices(k, t):
    return to_matrix(k, [a_i(k,i,t,d) for d in range(k) for i in range(0,k)])

def kers(l, k = 0):
    if (type(l[0]) == sage.matrix.matrix_modn_dense_float.Matrix_modn_dense_float):
        k = l[0].base_ring().cardinality()
        ks = map(lambda x: x.right_kernel(), l)
    else:
        if (k < 2):
            raise(ValueError("k is not get or has value < 2"))

```

```

        n = len(l[0])
        ks = map(lambda m: matrix(GF(k), m).right_kernel(), l)
    return ks

def test_hypot(k, n):
    return unit_kers(matrices(k,n)) != is_prime(n)

def unit_kers(l, k = 0):
    if (type(l[0]) == sage.matrix.matrix_modn_dense_float.Matrix_modn_dense_float):
        n = l[0].nrows()
        k = l[0].base_ring().cardinality()
        ks = map(lambda x: x.right_kernel(), l)
    else:
        if (k < 2):
            raise(ValueError("k is not get or has value < 2"))
        n = len(l[0])
        ks = kers(l, k)
    m = []
    for i in range(k):
        m.append([i] * n)
    return len(filter(lambda y: m == y, map(lambda x: map(list, list(x)), ks))) == len(l)

```

Программа на Haskell

На функциональном языке программирования Haskell была написана программа, позволяющая находить $\Pi(\tau)$, а также проверяющая периоды на сложность. Еще с помощью этой программы можно представление полиномов по векторам периодов.

```

k :: Int
k = 5

rol :: [a] -> Int -> [a]
rol xs i = let (l,r) = splitAt (i `mod` length xs) xs in r ++ l

ror :: [a] -> Int -> [a]

```

```

ror xs i = let (l,r) = splitAt (length xs - (i `mod` length xs)) xs in r ++ l

step n v = iterate iter [v] !! n where
  iter vs = nub $
    concatMap (\v->concatMap (\d->toList $ modkM $ aPolar d * fromPeriod v) [0..k-1]) vs

invert' k 1 = 1
invert' k p = (n * k + 1) `div` p
  where n = p - invert' p (k `mod` p)

invert = invert' k

allVectors n = sequence $ take (fromIntegral n) $ repeat [0..k-1]

allNormalVectors n = map (1:) $ allVectors (n-1)

normalize :: [Int] -> [Int]
normalize [] = error "Empty list"
normalize xs = normalize' xs xs where
  normalize' [] ys = error "Zero list"
  normalize' (x:xs) ys | x == 0 = normalize' xs ys
                       | otherwise = map (\n -> n * invert x `mod` k) ys

reverseMatrix = fromLists . reverse . toLists

modkM = fmap (`mod` k)

a_ij = uncurry a_ij' where
  a_ij' i j | i == 1 && j == 1 = 1
            | i == k = (-1) `mod` k
            | i == 1 || j == 1 = 0
            | otherwise = (-(j-1)^((k-1)-(i-1)) `mod` k) `mod` k

a = matrix k k a_ij

aPolar d = fromLists $ Data.List.transpose $
  rol (Data.List.transpose $ toLists a) d

```

```

fromPeriod xs = fromLists $ map (rol xs) [0..k-1]

polarCoeffs d xs = reverse $ toLists $ modkM $ aPolar d * fromPeriod xs

allCoeffs xs = concatMap (\d -> polarCoeffs d xs) [0..k-1]

setSort = compress . sort where
  compress [] = []
  compress (x:xs) = x : (compress $ dropWhile (== x) xs)

makeAllFamily ps = makeAllFamily' [normalize ps] $ setSort $
  map normalize $ allCoeffs ps where
    makeAllFamily' xss yss | xss == yss = yss
                          | otherwise = makeAllFamily' (insert zs xss) zss where
                                zs = normalize $ firstNotIn yss xss
                                zss = setSort $ yss ++ (map normalize $ allCoeffs zs)
  firstNotIn (y:ys) (x:xs) | y < x = y
                          | y > x = firstNotIn (y:ys) xs
                          | y == x = firstNotIn ys xs
  firstNotIn (y:ys) [] = y

singular [] = putStr ""
singular (x:xs) = do
  result <- try (evaluate (makeAllFamily x)) :: IO (Either SomeException [[Int]])
  case result of
    Left ex   -> putStrLn $ "Singular " ++ show x
    Right val -> putStrLn $ "Nonsingular " ++ show x ++ " " ++ (show $ length val)
  singular xs

```

Список литературы

1. Угрюмов Е. П. Цифровая схемотехника. СПб.: БХВ-Петербург, 2004.
2. Sasao T., Besslich P. On the complexity of mod-2 sum PLA's // IEEE Trans.on Comput. 39. N 2. 1990. P. 262–266.
3. Супрун В. П. Сложность булевых функций в классе канонических поляризованных полиномов // Дискретная математика. 5. №2. 1993. С. 111–115.
4. Перязев Н. А. Сложность булевых функций в классе полиномиальных поляризованных форм // Алгебра и логика. 34. №3. 1995. С. 323–326.
5. Селезнева С. Н. О сложности представления функций многозначных логик поляризованными полиномами. Дискретная математика. 14. №2. 2002. С. 48–53.
6. Кириченко К. Д. Верхняя оценка сложности полиномиальных нормальных форм булевых функций // Дискретная математика. 17. №3. 2005. С. 80–88.
7. Селезнева С. Н. Дайняк А. Б. О сложности обобщенных полиномов k -значных функций // Вестник Московского университета. Серия 15. Вычислительная математика и кибернетика. №3. 2008. С. 34–39.
8. Маркелов Н. К. Нижняя оценка сложности функций трехзначной логики в классе поляризованных полиномов // Вестник Московского университета. Серия 15. Вычислительная математика и кибернетика. №3. 2012. С. 40–45.
9. Селезнева С. Н. Маркелов Н. К. Быстрый алгоритм построения векторов коэффициентов поляризованных полиномов k -значных функций // Ученые записки Казанского университета. Серия Физико-математические науки. 2009. 151. №2 С. 147–151.
10. Башов М. А., Селезнев С. Н. О длине функций k -значной логики в классе полиномиальных нормальных форм по модулю k // Дискретная математика. – 2014. – Т. 26, вып. 3. – С. 3–9.

11. Селезнева С. Н. Сложность систем функций алгебры логики и систем функций трехзначной логики в классах поляризованных полиномиальных форм // Дискретная математика. – 2015. – Т. 27, вып. 1. – С. 111 – 122.
12. Балюк А. С., Янушковский Г. В. Верхние оценки функций над конечными полями в некоторых классах кронекеровых форм // Известия Иркутского государственного университета. Серия: Математика. – 2015. – Т. 14. – С. 3-17.
13. *SageMath, the Sage Mathematics Software System (Version 7.2)*, The Sage Developers, 2016, <http://www.sagemath.org>.