LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

# ZEMAS PAKĀPES POLINOMU BŪLA FUNKCIJU VAICĀJUMU SAREŽĢĪTĪBA

MAĢISTRA DARBS

Autors: **Igors Stepanovs**

Stud. apl. is06030

Darba vadītājs: profesors Andris Ambainis, Ph.D.

RĪGA 2012

# ANOTĀCIJA

Maģistra darbs "Zemas pakāpes polinomu Būla funkciju vaicājumu sarežģītība" mēģina uzlabot labāko zināmo attālumu starp Būla funkcijas pakāpi un determinēto vaicājumu sarežģītību. Šim nolūkam tiek meklēta 5. pakāpes 15 mainīgo Būla funkcija ar vaicājumu sarežģītību 15.

Tiek pētīti simetrizācijas polinomi, kas var atbilst meklējamai Būla funkcijai. Ir apkopotas zināmās un piedāvātas jaunas likumsakarības starp Būla funkcijas pakāpi, jutīgumu, determinēto vaicājumu sarežģītību un simetrizācijas polinomu. Šīs likumsakarības savā starpā salīdzinātas ar datora programmas palīdzību.

Tiek piedāvāts secīgi sadalīt simetrizācijas polinomus sīkākos, un ir izvirzītas īpašības, kurām ir jāizpildās šo sadalījumu gaitā. Ar datora programmas palīdzību parādīts, ka neeksistē 5. pakāpes 15 mainīgo Būla funkcija ar jutīgumu 15.

Atslēgvārdi: Būla funkcija, Būla funkcijas pakāpe, determinēta vaicājumu sarežģītība, funkcijas jutīgums, polinoma simetrizācijas metode.

# ABSTRACT

The master's thesis "Query Complexity of Boolean Functions with Low Polynomial Degree" attempts to improve the best known separation result between the degree and the deterministic query complexity of a Boolean function. For this purpose we search for a 15-variable Boolean function of degree 5 and query complexity 15.

We work with symmetrization polynomials which might correspond to a desired Boolean function. We study known, and propose new, relations between degree, sensitivity, deterministic query complexity of a Boolean function and its symmetrization polynomial.

We offer to repeatedly split symmetrization polynomials into polynomials of smaller size. We describe a set of conditions which must be met during this process. Using a computer program we prove no 15-variable Boolean function exists of degree 5 and sensitivity 15.

Keywords: Boolean function, degree of Boolean function, deterministic query complexity, function sensitivity, polynomial symmetrization method.

# AUTOREFERĀTS

Nozīmīgākais autora ieguldījums darba tapšanā ir sekojošs:

- 3.4.2. un 3.5.2. nodaļās ir aprakstītas jaunas metodes, kā no apakšas novērtēt Būla funkcijas jutīgumu un pakāpi, izmantojot šai Būla funkcijai atbilstošo simetrizācijas polinomu.

- 4.4. nodaļā ar datora programmu savā starpā salīdzināti pirmajās 3 nodaļās apskatītie kritēriji, kas apraksta likumsakarības starp Būla funkcijas pakāpi, jutīgumu, determinēto vaicājumu sarežģītību un simetrizācijas polinomu.

- 5. nodaļā aprakstīta simetrizācijas polinomu secīga sadalīšana sīkākos polinomos ar mērķi atrast sākotnējam simetrizācijas polinomam atbilstošo Būla funkciju, savukārt 6. nodaļā ir piedāvāti šīs metodes uzlabojumi.

- 7. nodaļā ir aprakstīta darba gaitā izveidotā C++ programma un ar tās palīdzību iegūtie rezultāti. Ar datora palīdzību tiek parādīts, ka neeksistē 5. pakāpes 15 mainīgo Būla funkcija ar jutīgumu 15.

# SATURA RĀDĪTĀJS

# APZĪMĒJUMU SARAKSTS

*Visur definēta funkcija* - funkcija, kas katru definīcijas apgabala elementu attēlo uz kādu vērtību apgabala elementu.

$\mathbb{N}_0$ - naturālu skaitļu kopa, ieskaitot 0.

*Kortežs* - sakārtota viena veida elementu virkne.

$[n]$ - skaitļu kopa $\{1, 2, ..., n\}$.

# IEVADS

Ir dota Būla funkcija $f : \{0,1\}^n \to \{0,1\}$, kuras argumentu vērtības nav zināmas. Determinēts vaicājošais algoritms katrā solī drīkst palūgt atklāt šīs Būla funkcijas patvaļīga argumenta vērtību. Par Būla funkcijas $f$ vaicājumu sarežģītību sauc mazāko soļu skaitu, ar kādu algoritms spēj izskaitļot funkcijas $f$ rezultātu neatkarīgi no saņemtām atbildēm.

Darba mērķis ir atrast tādu Būla funkciju, kurai attālums starp pakāpi un determinēto vaicājumu sarežģītību būtu lielāks par pašlaik labāko zināmo rezultātu. Jau iepriekš ir izpētītas Būla funkcijas, kuru pakāpe ir mazāka par 5. Šajā darbā mēģināsim atrast tādu 5. pakāpes 15 mainīgo Būla funkciju, kuras determinētā vaicājuma sarežģītība ir 15.

Izmantojot Būla funkcijas jutīguma jēdzienu un polinomu simetrizācijas metodi, aprakstīsim, kādām īpašībām ir jāpiemīt meklējamai Būla funkcijai. Pētīsim likumsakarības starp Būla funkcijas pakāpi, jutīgumu, determinēto vaicājumu sarežģītību un ar simetrizācijas metodi iegūstamiem simetrizācijas polinomiem, kuri varētu atbilst meklējamai Būla funkcijai. Tad mēģināsim atrastās likumsakarības pielietot, lai aprakstītu metodi 5. pakāpes 15 mainīgo Būla funkcijas meklēšanai.

Darba 1. nodaļā apskatīsim Būla funkcijas definīciju un ar Būla funkciju saistītos jēdzienus, kas būs nepieciešami darba gaitā. Darba 2. nodaļā izvirzīsim darba mērķus, kā arī aprakstīsim līdz šim labākos zināmos rezultātus.

Darba 3. nodaļā apskatīsim polinoma simetrizācijas metodi un vairākus kritērijus, kā no patvaļīga simetrizācijas polinoma iegūt informāciju par tam atbilstošās Būla funkcijas raksturlielumiem. Tam sekos 4. nodaļa, kurā mēs ar datora palīzību pārbaudīsim daudzas no iepriekšējās nodaļās aprakstītajām likumsakarībām, lai novērtētu to lietderīgumu darba mērķa sasniegšanai.

Darba 5. nodaļā aprakstīsim metodi nepieciešamo Būla funkciju meklēšanai, bet 6. nodaļā piedāvāsim šīs metodes uzlabojumu idejas.

Darbu beigsim ar 7. nodaļu, kurā aprakstīsim Būla funkciju meklēšanai darba gaitā izveidoto datora programmu un ar tās palīdzību iegūtos rezultātus.

# 1. BŪLA FUNKCIJAS VAICĀJUMU SAREŽĢĪTĪBA UN JUTĪGUMS

## 1.1. Būla funkcija

### *1.1.1. Būla funkcijas definīcija*

**Definīcija.** Būla funkcija ir visur definēta funkcija $f : \{0,1\}^n \to \{0,1\}$, kur $n \in \mathbb{N}_0$.

**Piemērs.** Apskatīsim Būla funkcijas piemēru, izmantojot Būla loģikas formulu

$$f(x_1, x_2, x_3, x_4) = x_2 \wedge (\neg x_3 \vee \neg x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3) \tag{1.1}$$

un tai atbilstošo vērtību tabulu (sk. 1.1. tab.). Darba gaitā šī funkcija tiks vairākkārt izmantota apskatāmo jēdzienu un metožu piemēros. □

*1.1. tabula*

**Būla funkcijas** (1.1) **vērtību tabula**

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $f(x_1, x_2, x_3, x_4)$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Būla funkcijai $f$ ar $n$ mainīgajiem eksistē $2^n$ dažādi ievaddatu korteži $x \in \{0,1\}^n$. Katram no šiem kortežiem atbilst viena no funkcijas vērtībām $f(x) \in \{0,1\}$. Tātad kopumā eksistē $2^{2^n}$ dažādas Būla funkcijas no $n$ mainīgajiem.

Korteža $x \in \{0,1\}^n$ Hemminga svaru jeb vienīnieku skaitu šajā kortežā $|\{i \in [n] \mid x_i = 1\}|$ apzīmēsim ar $|x|$. Ja ir dota kopa $B \subseteq \{1, ..., n\}$ un kortežs $x \in \{0,1\}^n$, tad

ar $x^B$ apzīmēsim tādu kortežu $y \in \{0,1\}^n$, kur

$$\forall i \in [n] : y_i = \begin{cases} x_i & \text{if } i \notin B \\ \neg x_i & \text{if } i \in B \end{cases}.$$

Savukārt ar $x^i$ apzīmēsim specgadījumu $x^{\{i\}}$.

**Definīcija.** Būla funkcijas $f(x)$ mainīgais $x_i$ ir fiktīvs, ja $\forall x \in \{0,1\}^n : f(x) = f(x^i)$. Ja mainīgais nav fiktīvs, tad to sauc par būtisku.

### *1.1.2. Būla funkciju reprezentējošais polinoms*

**Definīcija.** Ja $S$ ir mainīgo indeksu kopa, tad monoms $X_S$ ir mainīgo reizinājums $X_S = \prod_{i \in S} x_i$. Monoma pakāpe ir kopas $S$ izmērs. Par $n$ mainīgo multilineāru polinomu sauc funkciju $p : \mathbb{R}^n \to \mathbb{C}$, kuru var uzrakstīt kā $p(x) = \sum_{S \subseteq [n]} c_S X_S$ ar $c_S \in \mathbb{C}$. Polinoma $p$ pakāpe ir šī polinoma lielākā monoma pakāpe: $\deg(p) = \max\{|S| \mid c_S \neq 0\}$ [3].

**Definīcija.** Polinoms $p : \{0,1\}^n \to \{0,1\}$ reprezentē Būla funkciju $f : \{0,1\}^n \to \{0,1\}$, ja $\forall x \in \{0,1\}^n : p(x) = f(x)$.

Jebkuru $n$ argumentu Būla funkciju $f$ var reprezentēt ar polinoma palīdzību. Šāds polinoms vienmēr ir multilineārs, jo $x_i^n = x_i$ visiem $x_i \in \{0,1\}$ un $n \in \mathbb{N}$.

Ja $f : \{0,1\}^n \to \{0,1\}$ ir patvaļīga Būla funkcija un $x, y \in \{0,1\}^n$, tad funkciju $f$ reprezentē sekojošais polinoms:

$$p(y) = \sum_{x \in \{0,1\}^n : f(x)=1} \left( \prod_{i=0}^{n} g_{x_i}(y_i) \right), \text{ kur } g_{x_i}(y_i) = \begin{cases} 1 - y_i & \text{if } x_i = 0 \\ y_i & \text{if } x_i = 1 \end{cases}$$

**Lemma 1.** *Definēsim multilineārus polinomus $p, q : \mathbb{R}^n \to \mathbb{R}$, kuru pakāpes nepārsniedz $d$. Ja visiem $x \in \{0,1\}^n$ ar $|x| \leq d$ izpildās $p(x) = q(x)$ , tad $p = q$ [3].*

No šīs lemmas seko, ka katru Būla funkciju reprezentē viens unikāls multilineārs polinoms.

**Definīcija.** Par Būla funkcijas pakāpi $\deg(f)$ sauc tāda polinoma $p$ pakāpi $\deg(p)$, kas reprezentē šo Būla funkciju.

**Piemērs.** Apskatīsim Būla funkciju (1.1):

$$f(x_1, x_2, x_3, x_4) = x_2 \wedge (\neg x_3 \vee \neg x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3).$$

1.2. tabulā ir uzskaitīti visi ievaddatu korteži $x \in \{0,1\}^4$, kuriem $f(x) = 1$. Katram $x$ ir parādīts tam atbilstošais monoms $\prod_{i=0}^{n} g_{x_i}(y_i)$, kas ietilpst funkcijas $f$ reprezentējošā polinomā $p_f$. Saskaitot visus iegūtos monomus un atverot iekavas, iegūstam polinomu

$$p_f(x_1, x_2, x_3, x_4) = x_2 + x_3 - x_1x_3 - x_2x_3 + x_1x_2x_3 - x_2x_3x_4. \tag{1.2}$$

Tātad funkcijas $f$ pakāpe ir $\deg(f) = 3$. □

*1.2. tabula*

**Funkcijas** (1.1) **reprezentējošā polinoma monomu atrašana**

| $(x_1, x_2, x_3, x_4) : f(x_1, x_2, x_3, x_4) = 1$ | $\prod_{i=0}^{n} g_{x_i}(y_i)$ |
|---|---|
| $(0, 0, 1, 0)$ | $(1 - x_1) \cdot (1 - x_2) \cdot x_3 \cdot (1 - x_4)$ |
| $(0, 0, 1, 1)$ | $(1 - x_1) \cdot (1 - x_2) \cdot x_3 \cdot x_4$ |
| $(0, 1, 0, 0)$ | $(1 - x_1) \cdot x_2 \cdot (1 - x_3) \cdot (1 - x_4)$ |
| $(0, 1, 0, 1)$ | $(1 - x_1) \cdot x_2 \cdot (1 - x_3) \cdot x_4$ |
| $(0, 1, 1, 0)$ | $(1 - x_1) \cdot x_2 \cdot x_3 \cdot (1 - x_4)$ |
| $(1, 1, 0, 0)$ | $x_1 \cdot x_2 \cdot (1 - x_3) \cdot (1 - x_4)$ |
| $(1, 1, 0, 1)$ | $x_1 \cdot x_2 \cdot (1 - x_3) \cdot x_4$ |
| $(1, 1, 1, 0)$ | $x_1 \cdot x_2 \cdot x_3 \cdot (1 - x_4)$ |

## 1.2. Būla funkcijas vaicājumu sarežģītība

### *1.2.1. Būla funkcijas vaicājošais algoritms*

Ir dota Būla funkcija $f : \{0,1\}^n \to \{0,1\}$ un ir nepieciešams atrast tādu funkcijas rezultātu $f(x)$, kas atbilst sākotnēji nezināmiem ievaddatiem $x \in \{0,1\}^n$.

**Definīcija.** Par Būla funkcijas vaicājošo algoritmu sauc jautājumu uzdošanas stratēģiju, kas secīgi prasa funkcijas mainīgo vērtības $x_i$ un galā izdod funkcijas rezultātu $f(x)$.

**Definīcija.** Mēs sakām, ka vaicājošais algoritms izskaitļo $n$ mainīgo Būla funkciju $f$, ja visiem ievaddatiem $x \in \{0,1\}^n$ šis algoritms atgriež pareizu rezultātu $f(x)$.

Eksistē vairāku veidu vaicajošie algoritmi, piemēram – determinēts, nedeterminēts, varbūtisks vai kvantu vaicājošie algoritmi. Mēs apskatīsim determinētos vaicājošos algoritmus.

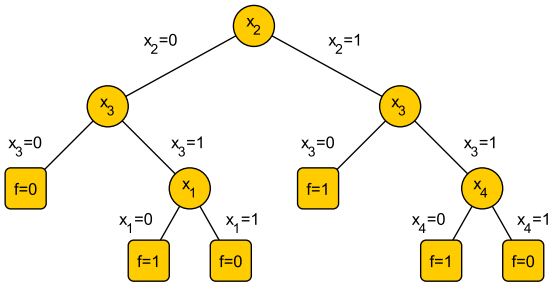
**Definīcija.** Par Būla funkcijas $f$ determinēto vaicajošo algoritmu sauc tādu vaicājošo algoritmu, kas izskaitļo $f$ un kura darbība jebkurā brīdī ir pilnīgi viennozīmīga.

Determinēto vaicājošo algoritmu var attēlot binārā lēmumu koka veidā. Katrai koka iekšējai virsotnei atbilst funkcijas mainīgais $x_i$. Šādai virsotnei vienmēr ir divi bērni, kas atbilst šī mainīgā vērtībām 0 un 1. Nonākot iekšējā virsotnē, prasām tai atbilstošā mainīgā vērtību. Iegūstam atbildi 0 vai 1 un nolaižamies uz pašreizējās virsotnes kreiso vai labo bērnu attiecīgi. Katrai koka lapai atbilst kāda funkcijas vērtība. Kad nonākam koka lapā, apstājamies un izdodam attiecīgo vērtību kā funkcijas rezultātu.

**Piemērs.** Apskatīsim Būla funkciju (1.1):

$$f(x_1, x_2, x_3, x_4) = x_2 \wedge (\neg x_3 \vee \neg x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3).$$

Viens no šīs funkcijas iespējamiem lēmumu kokiem ir redzams 1.1. attēlā. □




*1.1. att.* **Viens no iepējamiem funkcijas** (1.1) **lēmumu kokiem**

### *1.2.2. Būla funkcijas determinētā vaicājumu sarežģītība*

Jebkurš vaicājošais algoritms sliktākajā gadījumā uzdod tādu jautājumu skaitu, kas ir vienāds ar šī algoritma lēmumu koka augstumu.

**Definīcija.** Par Būla funkcijas $f$ vaicājumu sarežģītību $D(f)$ sauc mazāko lēmumu koka augstumu starp visiem vaicājošiem algoritmiem, kas aprēķina funkciju $f$.

**Piemērs.** Funkcijas (1.1) mainīgo skaits ir $n = 4$, un 1.1. attēlā redzamais lēmumu koks parāda vienu no iespējām sasniegt vaicājumu sarežģītību $D(f) = 3$. Tātad Būla funkcijas vaicājumu sarežģītība var būt mazāka par tās mainīgo skaitu pat tad, ja funkcija nesatur nevienu fiktīvu mainīgo. □

**Teorēma 2.** $\lfloor \log_2 n \rfloor + 1 \leq D(f) \leq n$.

**Pierādījums.** Acīmredzams, ka lielākā iespējamā vaicājumu sarežģītība ir vienāda ar funkcijas mainīgo skaitu. Piemēram, tāda ir funkcija $f(x_1, ..., x_n) = x_1 \vee ... \vee x_n$. Sliktākajā gadījumā $x_1 = ... = x_n = 0$, un pirms mēs varēsim pateikt, ka $f(x) = 0$, mums nāksies paprasīt katra mainīgā vērtību.

Apakšējam vaicājumu sarežģītības novērtējumam ir jēga tad, ja Būla funkcija nesatur fiktīvus mainīgos. Tad vaicājošā algoritma lēmumu kokā katram no mainīgajiem ir jāatbilst vismaz vienai virsotnei. Savukārt $n$ (iekšējo) virsotņu bināra koka augstums ir vismaz $\lfloor \log_2 n \rfloor + 1$. Šī ir mazākā iespējamā $D(f)$ vērtība. □

**Teorēma 3.** $\deg(f) \leq D(f)$ *[3].*

**Pierādījums.** Aplūkosim Būla funkcijas $f$ vaicājumu koku, kura augstums ir $D(f)$. Pieņemsim, ka $L$ ir šī koka lapa ar funkcijas rezultātu 1, kas atbilst mainīgajiem $x_1, ..., x_r$ un to vērtībām $b_1, ..., b_r$. Definēsim polinomu $p_L(x) = \prod_{i:b_i=1} x_i \prod_{i:b_i=0} (1 - x_i)$. Tad polinoma $p_L$ pakāpe ir $r \leq D(f)$. Šī polinoma vērtība ir $p_L(x) = 1$, ja lapa $L$ tiek sasniegta, apstrādājot ievaddatus $x$, un šī vērtība ir $p_L(x) = 0$ pretējā gadījumā. Definēsim $p = \sum_L p_L$, visu $p_L$ summu, kas atbilst vaicajumu koka lapām ar rezultātu 1. Tad $deg(p) \leq D(f)$ un $p(x) = 1$ tad un tikai tad, ja ar ievaddatiem $x$ tiek sasniegta kāda koka lapa ar rezultātu 1. Tātad polinoms $p$ reprezentē Būla funkciju $f$ [3]. □

## 1.3. Būla funkcijas jutīgums

**Definīcija.** Būla funkcijas $f : \{0,1\}^n \to \{0,1\}$ bloka jutīgums $bs_x(f)$ uz ievaddatu korteža $x \in \{0,1\}^n$ ir lielākā vērtība $b$, kurai eksistē tādas nešķeļošas kopas $B_1, ..., B_b$, ka visiem $i \in [b]$ izpildās $f(x) \neq f(x^{B_i})$. Funkcijas $f$ bloka jutīgums $bs(f) = max_x bs_x(f)$ ir lielākā no $b$ vērtībām pāri ievaddatu kopai $x \in \{0,1\}^n$ [3].

**Definīcija.** Funkcijas $f$ jutīgums $s(f)$ ir bloka jutīguma $bs(f)$ specgadījums, kad visu apskatīto kopu izmēri ir 1, jeb $\forall i \in [b] : |B_i| = 1$.

**Piemērs.** Sadalīsim $n = 4k^2$ mainīgos $2k$ blokos pa $2k$ mainīgajiem: pirmais bloks $B_1$ sastāv no $x_1, ..., x_{2k}$, otrais bloks $B_2$ sastāv no $x_{2k+1}, ..., x_{4k}$, utt.

Definēsim $f$ tā, lai $f(x) = 1$ tad un tikai tad, ja eksistē vismaz viens bloks $B_i$, kurā divi mainīgie blakus pozīcijās $x_{2i}, x_{2i+1}$ (kur $i \in \mathbb{N}$) ir vienādi ar 1, bet pārējie šī bloka $2k-2$ mainīgie ir vienādi ar 0.

Funkcijas $f$ jutīgums ir $s(f) = 2k$. Šis novērtējums sasniedzams gadījumā, kad katrā no blokiem tieši viens mainīgais ir vienāds ar 1. Tad katrā no $2k$ blokiem varam pamainīt tieši vienu mainīgo, lai funkcijas vērtība pamainītos no 0 uz 1.

Funkcijas $f$ bloka jutīgums ir $bs(f) = 2k^2$. Šis novērtējums ir sasniedzams, kad visi mainīgie ir vienādi ar 0. $4k^2$ elementus sadalām $2k^2$ kopās pa blakus pāriem. Pamainot jebkuras kopas abus mainīgos no 0 uz 1, funkcijas vērtība mainās no 0 uz 1 [8]. □

**Teorēma 4.** $s(f) \leq bs(f) \leq D(f)$ *[3].*

**Pierādījums.** Apskatām ievaddatu kortežu $x$ ar jutīguma kopām $B_1, ..., B_{bs(f)}$. Determinētam lēmuma kokam katrā no šīm kopām ir jāpaprasa vismaz viena mainīgā vērtību. Ja kādā no jutīguma kopām netiek paprasīta neviena mainīgā vērtība, tad mēs visus mainīgos šajā kopā varētu pamainīt uz pretējiem. Funkcijas rezultāts pamainītos, bet lēmumu koks šo faktu nebūtu pamanījis. Tātad lēmumu kokam ir nepieciešami vismaz $bs(f)$ jautājumi, jeb $bs(f) \leq D(f)$ [3].

Savukārt $s(f) \leq bs(f)$ izpildās, jo pēc definīcijas funkcijas jutīgums ir funkcijas bloka jutīguma specgadījums. □

**Piemērs.** Atradīsim jutīgumu Būla funkcijai (1.1):

$$f(x_1, x_2, x_3, x_4) = x_2 \wedge (\neg x_3 \vee \neg x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3).$$

Ievaddatu kortežs $x = (0, 0, 1, 1)$ ir viens no gadījumiem, kurā tiek sasniegts $s_x(f) = 3$, jo $f(0, 0, 1, 1) = f(0, 0, 1, 0) = 1$ un $f(1, 0, 1, 1) = f(0, 1, 1, 1) = f(0, 0, 0, 1) = 0$. Viegli pārbaudīt, ka neeksistē ievaddatu korteža $x \in \{0, 1\}^4$, kas dotu $s_x(f) = 4$. Tātad šīs funkcijas jutīgums ir $s(f) = 3$.

Teorēma 4 apgalvo, ka $bs(f) \geq s(f) = 3$. Bet $bs(f)$ nevar būt vienāds ar 4, jo tad visu izmantoto jutīguma kopu izmēram ir jābūt 1. No jutīguma $s(f)$ definīcijas sekotu, ka tad arī $s(f)$ ir jābūt vienādam ar 4. Tā kā $s(f) = 3$, tad šīs funkcijas bloka jutīgums ir $bs(f) = 3$. □

**Teorēma 5.** $\deg(f) \geq \sqrt{bs(f)/2}$ *[7].*

Šīs teorēmas pierādījumu tā lielā apjoma dēļ nesniegsim, tomēr vēlāk šo likumsakarību izmantosim, lai noformulētu darba mērķus.

# 2. BŪLA FUNKCIJAS AR ZEMU PAKĀPI UN AUGSTU VAICĀJUMU SAREŽĢĪTĪBU

## 2.1. Zināmās Būla funkcijas

Šajā nodaļā apskatīsim Būla funkcijas, kas sasniedz lielāko zināmo determinēto vaicājumu sarežģītību starp savas pakāpes funkcijām. Visas funkcijas definēsim, izmantojot polinomus, kuri reprezentē šīs funkcijas.

### *2.1.1. Nisan un Szegedy funkcija:* $\deg(f) = 2, D(f) = 3$

Funkcija

$$E(x_1, x_2, x_3) = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3 \tag{2.1}$$

ir vienāda ar 1 tad un tikai tad, ja precīzi divi no trijiem argumentiem ir savā starpā vienādi [7].

Šīs funkcijas pakāpe ir $\deg(E) = 2$, un tās determinētā vaicājumu sarežģītība ir $D(E) = 3$. Šādu vaicājumu sarežģītību iegūstam, jo sliktākajā gadījumā uz pirmajiem diviem jautājumiem mēs saņemsim vienādu atbildi. Ja trešā atbilde ir tāda pati, tad funkcijas vērtība ir 0, citādāk funkcijas vērtība ir 1.

Šīs funkcijas jutīgums ir $s(E) = 3$. Tas ir sasniedzams ar ievaddatu kortežu $x = (0, 0, 0)$, jo $E(0, 0, 0) = 0$, un $E(1, 0, 0) = E(0, 1, 0) = E(0, 0, 1) = 1$. Šī novērojuma arī būtu pieticis, lai apgalvotu, ka $D(E) = 3$, jo no teorēmām 2 un 4 iegūstam $s(E) \leq D(E) \leq n$, kur $s(E) = 3$ un $n = 3$.

Mūs interesē lielākais attālums starp $\deg(f)$ un $D(f)$. Mēs attāluma jēdzienu definējam kā $\log_{D(E)} \deg(E)$ [7]. Jo lielāks ir attālums, jo mazāka ir šī vērtība. Šajā gadījumā iegūstam $\log_{D(E)} \deg(E) = \log_3 2 \approx 0.63$.

### *2.1.2. Kushilevitz funkcija:* $deg(f) = 3, D(f) = 6$

Kushilevitz funkcija ir definēta kā $E'(z_1, ..., z_6) = \sum_i z_i - \sum_{ij} z_iz_j + z_1z_3z_4 + z_1z_2z_5 + z_1z_4z_5 + z_2z_3z_4 + z_2z_3z_5 + z_1z_2z_6 + z_1z_3z_6 + z_2z_4z_6 + z_3z_5z_6 + z_4z_5z_6$ [6].

Šīs funkcijas pakāpe ir $\deg(E') = 3$. Tās lielākais iespējamais jutīgums $s(E') = 6$ ir sasniedzams ar ievaddatu kortežu $x = (0, 0, 0, 0, 0, 0)$, kuram atbilst funkcijas vērtība

$E'(x) = 0$, jo $E'(y) = 1$ visiem ievaddatu kortežiem $y \in \{0,1\}^6$ ar Hemminga svaru $|y| = 1$. Līdz ar to šīs funkcijas determinētā vaicājumu sarežģītība arī ir $D(E') = 6$.

Attālums starp Kushilevitz funkcijas pakāpi un vaicājumu sarežģītību ir $\log_{D(E')} \deg(E') = \log_6 3 \approx 0.61$. Šis rezultāts ir labāks par iepriekšējo attālumu $\log_3 2$, kas tika iegūts Nisan un Szegedy funkcijas gadījumā.

### *2.1.3. Nisan un Szegedy funkcijas vispārinājums:* $\deg(f) = 4, D(f) = 9$

Definējam funkciju $E^k = E(E^{k-1}, E^{k-1}, E^{k-1})$, kur katra no funkcijām $E^{k-1}$ sastāv no $3^{k-1}$ dažādiem mainīgajiem un $E^0(x) = x$. Šī funkcija ir vispārināta 2.1.1. nodaļas funkcija, kur $E = E^1$ [7].

Atkarībā no koeficienta $k$ iegūstam $\deg(E^k) = 2^k$ un $n = D(E^k) = s(E^k) = 3^k$. Izvēloties $k = 2$, iegūstam funkciju $E^2(x_1, ..., x_9)$, kurai $\deg(E^2) = 4$ un $D(E^2) = 9$.

Attālums starp pakāpi un vaicājumu sarežģītību nemainās atkarībā no $k$ vērtības, jo $\log_{D(E^k)} \deg(E^k) = \log_{3^k} 2^k = \log_3 2$.

Jāpiebilst, ka līdzīgā veidā var vispārināt arī Kushilevitz funkciju, iegūstot funkcijas ar pakāpi $3^k$ un determinēto vaicājumu sarežģītību $6^k$ [6].

## 2.2. Darba mērķis un motivācija

### *2.2.1. Attālums starp* $\deg(f)$ *un* $D(f)$

Iepriekš apskatītās Būla funkcijas sasniedz lielāko zināmo determinēto vaicājumu sarežģītību starp funkcijām ar pakāpēm $2 \leq \deg(f) \leq 4$. Pakāpe $\deg(f) = 1$ netika minēta, jo šis gadījums ir triviāls ar $f(x_1) = x_1$ un $n = \deg(f) = D(f) = 1$, līdz ar ko $\deg(f) = n$[1].

Nav zināms, kāda ir lielākā iespējamā vaicājumu sarežģītība funkcijām ar $\deg(f) = 5$ vai lielākām pakāpēm. Šī darba galvenais mērķis ir atrast 5. pakāpes Būla funkciju, kas uzlabotu mazāko zināmo attālumu starp $\deg(f)$ un $D(f)$. Līdz ar to mūs interesē, vai eksistē $\deg(f) = 5$ pakāpes Būla funkcija, kurai $\deg(f) < D(f)^{\log_6 3}$ jeb $D(f) > \deg(f)^{1/\log_6 3} \approx 13.80$. Tātad mēs vēlamies atrast 5. pakāpes Būla funkciju ar $D(f) \geq 14$.

Ja meklējamā funkcija tiks atrasta, tas varētu noderēt arī citu problēmu risināšanā. Piemēram, iepriekš pieminējām, ka eksistē arī kvantu vaicājošie algoritmi. Kvantu vaicājumu sarežģītība $Q_E(f)$ raksturo kvantu vaicājošos algoritmus, kas aprēķina funkciju $f$. Vērtībai $Q_E(f)$ ir zināmi sekojošie apakšējie novērtējumi:

**Lemma 6.** $Q_E(f) \geq \dfrac{\deg(f)}{2}$ *[2].*

**Lemma 7.** $Q_E(f) \geq \sqrt{\dfrac{bs(f)}{8}}$ *[2].*

Atrodot Būla funkciju ar pēc iespējas lielāku attālumu starp $\deg(f)$ un $D(f)$, var cerēt, ka šai funkcijai izpildīsies arī $Q_E(f) < D(f)/2$. Šādu funkciju atrašana pašlaik ir aktuāla problēma.

### *2.2.2. Attālums starp* $\deg(f)$ *un* $s(f)$

Atsevišķas uzmanības vērts ir jautājums, kāds ir mazākais attālums starp funkcijas pakāpi un funkcijas jutīgumu vai bloka jutīgumu. Līdzīgi kā ar determinēto vaicājumu sarežģītību, arī šajā gadījumā Kushilevitz funkcija (sk. 2.1.2. nod.) sniedz pašlaik mazāko zināmo attālumu starp šiem lielumiem: $\deg(f) = n^{\log_6 3}$, kur $n = s(f) = bs(f)$.

5. teorēma parāda, ka $\deg(f)$ var būt ne vairāk kā kvadrātiski mazāka par $bs(f)$, jeb $\deg(f) \geq \sqrt{\dfrac{bs(f)}{2}} \approx 0.7 bs(f)^{0.5}$. Tātad labākais zināmais attālums ir jau ļoti tuvs minimāli iespējamam attālumam. Šis novērtojums attiecās arī uz $s(f)$, jo $s(f) \leq bs(f)$.

Var pamanīt, ka visām iepriekš apskatītām funkcijām izpildās $s(f) = bs(f) = D(f) = n$. Pēc analoģijas būtu vērts arī 5. pakāpes funkciju sākt meklēt, iesākot ar gadījumiem, kad izpildās $s(f) = n$.

Attālums starp $\deg(f)$ un $s(f)$ mūs interesē ne tikai determinētās vaicājumu sarežģītības kontekstā. Cita starpā no $s(f)$ vērtības ir atkarīga arī funkcijas komunikāciju sarežģītība.

**Definīcija.** Apskatīsim Būla funkciju $f : X \times Y \to \{0, 1\}$. Šī funkcija kā ievaddatus pieņem divus kortežus $x$ un $y$. Alise zina $x$ un Bobs zina $y$. Lai izrēķinātu funkciju $f$, viņiem savā starpā ir jāapmainās ar informāciju par $x$ un $y$. Par funkcijas komunikācijas sarežģītību $c(f)$ sauc mazāko pārsūtāmo bitu skaitu, ar kuru vienmēr pietiks, lai izrēķinātu funkcijas $f$ vērtību [6].

**Lemma 8.** *Ja $f$ ir $n$ argumentu Būla funkcija un $s(f) = n$, tad $c(f) = \Omega(n)$ [6].*

Interesanti, ka pati komunikāciju sarežģītība kalpo par apakšējo novērtējumu determinētai vaicājumu sarežģītībai.

**Lemma 9.** *Ja $f$ ir $2n$ argumentu Būla funkcija, tad $c(f) \leq D(f)$ [4].*

# 3. POLINOMU SIMETRIZĀCIJAS METODE

## 3.1. Polinomu simetrizācijas metodes jēdziens

**Definīcija.** Apskatīsim polinomu $p : \mathbb{R}^n \to \mathbb{R}$. Ja $\pi$ ir permutācija un $x = x_1...x_n$, tad definēsim $\pi(x) = (x_{\pi(1)}, ..., x_{\pi(n)})$. Ar $S_n$ apzīmēsim kopu, kas sastāv no visām iespējamām $n$ elementu $n!$ permutācijām. Tad simetrizācija $p^{sym}$ tiek iegūta kā polinoma $p$ vidējā vērtība, ievaddatu virknes veidā padodot visas iespējamās permutācijas no virknes $x$ elementiem [7]:

$$p^{sym}(x) = \frac{\sum_{\pi \in S_n} p(\pi(x))}{n!}.$$

Simetrizācijas polinoma pakāpe nepārsniedz sākotnējā polinoma pakāpi, bet var būt arī mazāka: $\deg(p^{sym}) \leq \deg(p)$.

**Piemērs.** Vairāki polinomu simetrizācijas piemēri ir aplūkojami 3.1. tabulā. Tai skaitā tiek simetrizēts Nisan un Szegedy polinoms $E$ (2.1), kā arī polinoms $p_f$ (1.2), kas reprezentē Būla funkciju (1.1). Simetrizējot polinomu $p_f$, novērojam, ka $\deg(p_f) = 3$, bet $\deg(p_f^{sym}) = 2$. □

*3.1. tabula*

**Daži polinomu simetrizācijas piemēri**

| Sākotnējais polinoms | Simetrizētais polinoms |
|---|---|
| $p_0 = x_1 + x_2$ | $p_0^{sym} = \frac{(x_1 + x_2) + (x_2 + x_1)}{2} = x_1 + x_2$ |
| $p_1 = x_1 - x_2 - x_1x_2$ | $p_1^{sym} = \frac{(x_1 - x_2 - x_1x_2) + (x_2 - x_1 - x_2x_1)}{2} = -x_1x_2$ |
| $p_2 = x_1 - x_2 + x_3 + x_1x_2x_3$ | $p_2^{sym} = \frac{2(x_1 + x_2 + x_3) + 6x_1x_2x_3}{6} = \frac{1}{3}\sum_{i \in [3]} x_i + x_1x_2x_3$ |
| $E = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3$ | $E^{sym} = \frac{6}{6}\sum_{i \in [3]} x_i - \frac{6}{6}\sum_{\substack{i,j \in [3] \\ i \neq j}} x_ix_j$ |
| $p_f = x_2 + x_3 - x_2x_3 - x_1x_3 + x_1x_2x_3 - x_2x_3x_4$ | $p_f^{sym} = \frac{12}{24}\sum_{i \in [4]} x_i - \frac{8}{24}\sum_{\substack{i,j \in [4] \\ i \neq j}} x_ix_j$ |

**Definīcija.** Polinomu sauc par simetrisku tad, ja tā vērtība ir pilnībā atkarīga no ievaddatu Hemminga svara $|x|$.

Simetrizācijas metode sākotnējam polinomam ievaddatu veidā padod visas iespējamās ievaddatu korteža permutācijas. Līdz ar to katra ievaddatu korteža pozīcija simetrizācijas polinomā ir vienlīdz nozīmīga. Tas savukārt parāda, ka visi ar simetrizācijas metodi iegūtie polinomi ir simetriski.

## 3.2. Simetrizācijas polinoms kā viena mainīgā polinoms

**Lemma 10.** *Ja $p : \mathbb{R}^n \to \mathbb{R}$ ir multilineārs polinoms, tad eksistē tāds viena mainīgā polinoms $q : \mathbb{R} \to \mathbb{R}$ ar pakāpi $\leq \deg(p)$, ka $p^{sym}(x) = q(|x|)$ visiem $x \in \{0,1\}^n$ [3].*

**Pierādījums.** Ar $d \leq \deg(p)$ apzīmēsim $\deg(p^{sym})$. Ar $V_j$ apzīmēsim visu iespējamo $\binom{n}{j}$ monomu summu, katrs no kuriem sastāv no $j$ dažādiem mainīgiem: $V_1 = x_1 + ... + x_n$, $V_2 = x_1x_2 + x_1x_3 + ... + x_{n-1}x_n$, utt.

Polinoms $p^{sym}$ ir simetrisks, tāpēc pēc indukcijas ir viegli parādīt, ka $p^{sym}(x) = c_0 + c_1V_1 + c_1V_2 + ... + c_dV_d$ ar $c_i \in \mathbb{R}$. Tā kā $x_i \in \{0,1\}$, tad

$$V_j = \binom{|x|}{j} = \frac{|x|(|x|-1)(|x|-2)...(|x|-j+1)}{j!}$$

ir $j$. pakāpes polinoms. Tātad mēs varam nodefinēt viena mainīgā polinomu kā $q(|x|) = c_0 + c_1\binom{|x|}{1} + c_2\binom{|x|}{2} + ... + c_d\binom{|x|}{d}$ [3]. □

**Piemērs.** Katram no 3.1. tabulā iegūtajiem simetrizācijas polinomiem 3.2. tabulā atrodam atbilstošo viena mainīgā simetrizācijas polinomu.

Sīkāk izteiksim viena mainīgā polinomu $q_f$, kas iegūts no $p_f^{sym}$:

$$q_f(|x|) = \frac{1}{2}\binom{|x|}{1} - \frac{1}{3}\binom{|x|}{2} = \frac{3|x| - |x|(|x|-1)}{6} = \frac{4|x| - |x|^2}{6}. \quad (3.1)$$

Šī polinoma vērtības ir: $q_f(0) = 0$, $q_f(1) = \frac{1}{2}$, $q_f(2) = \frac{2}{3}$, $q_f(3) = \frac{1}{2}$ un $q_f(4) = 0$. □

## 3.3. Simetrizācijā ietvertā informācija par sākotnējo polinomu

Dažādiem sākotnējiem polinomiem $p$ var atbilst viens un tas pats simetrizācijas polinoms $p^{sym}$. Piemēram, polinomu $p_a(x_1, x_2) = 0$ un $p_b(x_1, x_2) = x_1 - x_2$ simetrizācijas polinomi

sakrīt: $p_a^{sym}(x_1, x_2) = p_b^{sym}(x_1, x_2) = 0$. Tātad nav iespējams no simetrizācijas polinoma viennozīmīgi atjaunot sākotnējo polinomu. Turpmāk mēs pētīsim, kādu informāciju simetrizācijas polinoms satur par tam atbilstošiem sākotnējiem polinomiem.

*3.2. tabula*

**Viena mainīgā polinomu iegūšana no simetrizācijas polinomiem**

| Simetrizācijas polinoms | Viena mainīgā polinoms |
|---|---|
| $p_0^{sym} = x_1 + x_2 = V_1$ | $q_0 = \binom{\lvert x\rvert}{1}$ |
| $p_1^{sym} = -x_1x_2 = -V_2$ | $q_1 = -\binom{\lvert x\rvert}{2}$ |
| $p_2^{sym} = \frac{1}{3}\sum_{i\in[3]} x_i + x_1x_2x_3 = \frac{1}{3}V_1 + V_3$ | $q_2 = \frac{1}{3}\binom{\lvert x\rvert}{1} + \binom{\lvert x\rvert}{3}$ |
| $E^{sym} = \frac{6}{6}\sum_{i\in[3]} x_i - \frac{6}{6}\sum_{\substack{i,j\in[3]\\ i\neq j}} x_ix_j = V_1 + V_2$ | $q_E = \binom{\lvert x\rvert}{1} + \binom{\lvert x\rvert}{2}$ |
| $p_f^{sym} = \frac{12}{24}\sum_{i\in[4]} x_i - \frac{8}{24}\sum_{\substack{i,j\in[4]\\ i\neq j}} x_ix_j = \frac{1}{2}V_1 - \frac{1}{3}V_2$ | $q_f = \frac{1}{2}\binom{\lvert x\rvert}{1} - \frac{1}{3}\binom{\lvert x\rvert}{2}$ |

**Teorēma 11.** *Ja $f : \{0,1\}^n \to \{0,1\}$ ir Būla funkcija un $q$ ir tās viena mainīgā simetrizācijas polinoms, tad*

$$\forall k \in \{0, ..., n\} : q(k)\binom{n}{k} = |\{x \in \{0,1\}^n \mid |x| = k, f(x) = 1\}|.$$

**Pierādījums.** No simetrizācijas metodes definīcijas un 10. lemmas zināms, ka $q(k) = p^{sym}(x) = \frac{\sum_{\pi\in S_n} p(\pi(x))}{n!}$ jebkuram ievaddatu kortežam $x \in \{0,1\}^n$ ar $|x| = k$. Pareizinām abas puses ar $n!$ un iegūstam $\sum_{\pi\in S_n} p(\pi(x)) = q(k) \cdot n!$.

Šajā summā katrs ievaddatu kortežs $x \in \{0,1\}^n$, kas sastāv no $|x| = k$ vieniniekiem un kuram izpildās $f(x) = 1$, ietilpst $k!(n-k)!$ reizes. Jo tik veidos var savā starpā ievaddatos pārkārtot $k$ vieniniekus un $n-k$ nulles. Tātad kopējais šādu unikālo kortežu skaits ir vienāds ar $\frac{q(k)\cdot n!}{k!(n-k)!} = q(k)\binom{n}{k}$. □

Reprezentējot simetrizācijas polinomu kā $q(k)\binom{n}{k}$, kļūst acīmredzams, ka šī polinoma vērtības ir stipri ierobežotas – punktā $k$ pastāv tikai $\binom{n}{k} + 1$ dažādas iespējamās vērtības. Šis novērojums seko no 11. teorēmas.

**Sekas 12.** *Ja* $f : \{0,1\}^n \to \{0,1\}$ *ir Būla funkcija un* $q$ *ir tās viena mainīgā simetrizācijas polinoms, tad visiem* $k \in \{0,...,n\}$*:*

$$q(k)\binom{n}{k} \in \mathbb{N}_0 \text{ un } 0 \leq q(k)\binom{n}{k} \leq \binom{n}{k}.$$

**Piemērs.** Izteiksim Būla funkcijai $f$ (1.1) atbilstošo viena mainīgā simetrizācijas polinomu $q_f$ (3.1) kā $q_f(k)\binom{n}{k} = \frac{4 \cdot k - k^2}{6}\binom{n}{k}$:

- $q(0)\binom{4}{0} = \frac{4 \cdot 0 - 0^2}{6}\binom{4}{0} = 0$. Ja $|x| = 0$, tad 0 gadījumos izpildās $f(x) = 1$.
- $q(1)\binom{4}{1} = \frac{4 \cdot 1 - 1^2}{6}\binom{4}{1} = 2$. Ja $|x| = 1$, tad 2 gadījumos izpildās $f(x) = 1$, utt.

Visas iegūtās vērtības var aplūkot 3.3. tabulā. □

*3.3. tabula*

**Polinoma** $q_f$ (3.1) **reprezentācija** $q_f(k)\binom{n}{k}$

| $k$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $q_f(k)\binom{n}{k}$ | 0 | 2 | 4 | 2 | 0 |

Tupmāk darbā par viena mainīgā simetrizācijas polinomu $q$ visbiežāk runāsim, reprezentējot to ar vērtībām $q(k)\binom{n}{k}$. Atkarībā no šīm vērtībām, spēsim dot apakšējos novērtējumus vairākiem sākotnējās Būla funkcijas raksturlielumiem.

**Piezīme.** No šī brīža pieturēsimies pie sekojošas semantikas:

- Par simetrizācijas polinomu $p^{sym}$ turpināsim saukt polinomu, kas ir iegūts no $n$ mainīgo Būla funkcijas $f$ ar simetrizācijas metodes palīdzību.
- Par viena mainīgā polinomu $q$ turpināsim saukt polinomu, kas tiek iegūts no simetrizācijas polinoma $p^{sym}$.
- Par *simetrizācijas polinomu* $r$ sauksim viena mainīgā simetrizācijas polinoma $q$ vērtības, kas ir izteiktas kā $q(k)\binom{n}{k}$. Tātad $r(k) \equiv q(k)\binom{n}{k}$ $\forall k \in \{0,...,n\}$.

Šāda notācija var likties maldinoša, tomēr tā palīdz vienkāršot turpmākā teksta uztveri. Ievērosim arī to, ka jebkuru no šīm trīs reprezentācijām var pārveidot par jebkuru citu reprezentāciju. Tāpēc, kamēr par atkarību starp Būla funkciju un simetrizācijas polinomu runājam vispārīgā līmenī, mēs varam nenorādīt, kāda ir šī polinoma reprezentācija.

## 3.4. Būla funkcijas jutīguma atkarība no simetrizācijas polinoma

### *3.4.1. Simetrizācijas polinomi, kas atbilst Būla funkcijām ar* $s(f) \geq m$

Mūs interesē kritēriji, kas patvaļīgai $m$ vērtībai palīdzētu atpazīt tādus simetrizācijas polinomus, kas atbilst Būla funkcijām $f$ ar jutīgumu $s(f) \geq m$.

**Teorēma 13.** *Ja $f$ ir $n$ argumentu Būla funkcija, $r$ ir šīs funkcijas simetrizācijas polinoms un $r(0) = 0$, tad $s(f) \geq r(1)$.*

**Pierādījums.** Aplūkosim ievaddatu kortežu $x = (0, 0, ..., 0)$. Nosacījums $r(0) = 0$ nozīmē, ka $f(x) = 0$. Savukārt katrs no $r(1)$ dažādiem kortežiem $x' \in \{0, 1\}^n$, kuriem izpildās $f(x') = 1$, ir iegūstams no $x$ ar vienas pozīcijas nomaiņu. Pēc funkcijas jutīguma definīcijas iegūstam $s(f) \geq s_x(f) = r(1)$. □

Tātad, ja patvaļīgai $m$ vērtībai aplūkojam visus simetrizācijas polinomus $r$, kuriem izpildās $r(0) = 0$ un $r(1) = m$, tad šiem polinomiem atbilst tikai tādas Būla funkcijas $f$, kurām $s(f) \geq m$. Nākamā teorēma parāda, ka jebkuru Būla funkciju $f$ ar $s(f) = m$ var pārveidot tā, lai tās simetrizācijas polinoms atbilstu šiem kritērijiem.

**Teorēma 14.** *Ja $n$ argumentu Būla funkcijas $f$ jutīgums ir $s(f) = m$, tad eksistē Būla funkcija, kurai izpildās $r(0) = 0$, un $r(1) = m$.*

**Pierādījums.** Pieņemsim, ka novērtējums $s(f) = m$ tiek sasniegts ar ievaddatu virkni $x \in \{0, 1\}^n$, jeb $s(f) = s_x(f) = m$.

Tad funkcijas $f$ loģikas formulā pieliekam negācijas operatorus visiem mainīgajiem $x_i$, kuru vērtība ievaddatu kortežā ir $x_i = 1$. Rezultātā no funkcijas $f$ tiek izveidota tāda funkcija $f'$, kuras maksimālais jutīgums $m$ tiek sasniegts punktā $x' = (0, 0, ..., 0)$.

Atliek pārbaudīt, ka $f'(x') = 0$. Ja šis nosacījums neizpildās, tad esam ieguvuši $r_{f'}(0) = 1$ un $r_{f'}(1) = n - m$. Šajā gadījumā pieliekam negāciju funkcijas $f'$ formulas priekšā, lai iegūtu funkciju $f''$, kurai izpildās $r_{f''}(0) = 0$ un $r_{f''}(1) = m$. □

**Piemērs.** Apskatīsim funkciju (1.1):

$$f(x_1, x_2, x_3, x_4) = x_2 \wedge (\neg x_3 \vee \neg x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3).$$

1.3. nodaļas piemērā redzējām, ka šīs funkcijas jutīgums $s(f) = 3$ ir sasniedzams ar ievaddatiem $x = (0, 0, 1, 1)$. Savukārt 3.3. tabulā var aplūkot šai funkcijai atbilstošā simterizācijas polinoma $r_f(k) = q_f(k)\binom{n}{k}$ vērtības.

Tā kā $x_3 = 1$ un $x_4 = 1$, tad formulā pirms visām mainīgo $x_3$ un $x_4$ kopijām ievietojam negāciju un iegūstam formulu

$$f'(x_1, x_2, x_3, x_4) = x_2 \wedge (x_3 \vee x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge \neg x_3).$$

Pārbaudam, ka $f'(0, 0, 0, 0) = f'(0, 0, 0, 1) = 1$ un $f'(1, 0, 0, 0) = f'(0, 1, 0, 0) = f'(0, 0, 1, 0) = 0$. Tātad iegūtā formula priekš $|x| = 0$ ir vienāda ar 1, un priekš $|x| = 1$ tieši vienā no četriem gadījumiem ir vienāda ar 1, jeb $r_{f'}(0) = 1$ un $r_{f'}(1) = 1$.

Pieliekam formulai $f'$ priekšā negāciju un iegūstam formulu

$$f''(x_1, x_2, x_3, x_4) = \neg(x_2 \wedge (x_3 \vee x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)).$$

Visi formulas $f''$ rezultāti ir pretēji tiem, ko ieguvām ar formulu $f'$, tātad $r_{f''}(0) = \binom{4}{0} - r_{f'}(0) = 0$ un $r_{f''}(1) = \binom{4}{1} - r_{f'}(1) = 3$. □

### *3.4.2. Patvaļīga simetrizācijas polinoma Būla funkcijas jutīgums*

Iepriekšējās nodaļas rezultāti ir noderīgi tad, ja mēs pārlasām simetrizācijas polinomus un mūs no tiem interesē identificēt tādus, kas atbilst Būla funkcijai $f$ ar jutīgumu $s(f) \geq m$ patvaļīgai $m$ vērtībai. Tomēr mūs interesētu novērtēt jutīgumu arī tad, kad apskatām kādu konkrētu simetrizācijas polinomu, kas netika iegūts speciāli šim nolūkam. Tāpēc vispārināsim 13. teorēmu.

Šīs nodaļas ietvaros uzskatīsim, ka simetrizācijas polinoms $r$ ir definēts arī punktos $-1$ un $n + 1$, kuros tā vērtības ir $r(-1) = r(n + 1) = 0$.

**Teorēma 15.** *Apskatīsim $n$ argumentu Būla funkciju $f$ un tās simetrizācijas polinomu $r$ patvaļīgā punktā $0 \leq k \leq n$. Ja $r(k) \neq 0$, tad*

$$s(f) \geq n - \left\lfloor \frac{r(k-1) \cdot \min(n-k+1, r(k)) + r(k+1) \cdot \min(k+1, r(k))}{r(k)} \right\rfloor.$$

**Pierādījums.** Eksistē $r(k)$ ievaddatu korteži $x \in \{0, 1\}^n$, kuri satur precīzi $|x| = k$ vieniniekus un kuriem izpildās $f(x) = 1$. Atradīsim lielāko jutīgumu $s_x(f)$, kuru kāds no šiem kortežiem sasniegs sliktākajā gadījumā.

Ja no $k$-vieninieku kortеža $x$ ar $f(x) = 1$ var precīzi $m$ dažādos veidos ar vienas pozīcijas izmaiņu iegūt kortežus $x'$ ar $|x'| \in \{k-1, k+1\}$ un $f(x') = 1$, tad $s_x(f) = n-m$ pēc funkcijas jutīguma definīcijas. Tātad sliktākais novērtējums tiks iegūts tad, ja no apskatītiem $r(k)$ kortežiem ar viena bita nomaiņu tiks iegūts pēc iespējas lielāks skaits tādu kortežu, kas satur $k-1$ vai $k+1$ vieniniekus un kuriem funkcijas vērtība arī ir 1.

Katram no $r(k-1)$ kortežiem pastāv $(n-k+1)$ iespējas nomainīt kādu 0-bitu uz 1-bitu, lai potenciāli iegūtu vienu no $r(k)$ kortežiem. Un katram no $r(k+1)$ kortežiem pastāv $(k+1)$ iespēja nomainīt vienu 1-bitu uz 0-bitu, lai potenciāli iegūtu vienu no $r(k)$ kortežiem. Abos gadījumos koeficientu $(n-k+1)$ un $(k+1)$ vietā varam izmantot $\min(n-k+1, r(k))$ un $\min(k+1, r(k))$, jo nevaram iegūt tādu $k$-vienineku kortežu skaitu, kas ir lielāks par $r(k)$ .

Sliktākajā gadījumā apskatītie $r(k)$ korteži summāri tiks iegūti

$$r(k-1) \cdot \min\left(n-k+1, r(k)\right) + r(k+1) \cdot \min\left(k+1, r(k)\right)$$

reizes, un vismaz viens no tiem tiks iegūts ne vairāk kā

$$\left\lfloor \frac{r(k-1) \cdot \min(n-k+1, r(k)) + r(k+1) \cdot \min(k+1, r(k))}{r(k)} \right\rfloor$$

reizes. No šī novērojuma seko teorēmas formulējums. □

Pēc analoģijas noformulēsim simetrisko gadījumu, kad funkcijas jutīguma novērtēšanai tiek izmanti ievaddati, kuriem funkcijas vērtība ir 0 nevis 1. Ievērosim, ka eksistē $\binom{n}{k} - r(k)$ tādu kortežu $x \in \{0,1\}^n$, kuriem $|x| = k$ un $f(x) = 0$.

**Sekas 16.** *Apskatīsim $n$ argumentu Būla funkciju $f$ un tās simetrizācijas polinomu $r$ patvaļīgā punktā $0 \le k \le n$. Ja $\binom{n}{k} - r(k) \neq 0$, tad $s(f) \ge n -$*

$$\left\lfloor \frac{\left(\binom{n}{k-1} - r(k-1)\right) \min\left(n-k+1, \binom{n}{k} - r(k)\right) + \left(\binom{n}{k+1} - r(k+1)\right) \min\left(k+1, \binom{n}{k} - r(k)\right)}{\binom{n}{k} - r(k)} \right\rfloor.$$

**Piemērs.** Atgriezīsimies pie Būla funkcijas $f$ (1.1), kuras mainīgo skaits ir $n = 4$ un kuras jutīgums $s(f) = 3$ ir sasniedzams ar ievaddatiem $x = (0,0,1,1)$, kam izpildās $f(x) = 1$ (sk. 1.3. nod.) Šīs funkcijas simetrizācijas polinoms ir $r(k) = (0,2,4,2,0)$ (sk. 3.3. tab.).

Tā kā $|x| = 2$, tad varētu sagaidīt, ka šīs funkcijas labāko jutīguma novērtējumu dos 15. teorēma punktā $k = 2$. Tomēr iegūstam $s(f) \ge 4 - \left\lfloor \frac{2 \cdot \min(3,4) + 2 \cdot \min(3,4)}{4} \right\rfloor = 1$. Tātad šī likumsakarība var dot diezgan neprecīzu rezultātu.

Labāku novērtējumu iegūstam punktā $k = 0$, izmantojot 16. sekas: $s(f) \ge 4 -$

$\left\lfloor \frac{0 + \left(\binom{4}{1} - 2\right) \cdot \min\left(1, \binom{4}{0} - 0\right)}{\binom{4}{0} - 0} \right\rfloor = 2$. Šis rezultāts nav optimāls, tomēr viegli pārbaudīt, ka nevienā punktā šajā nodaļā aprakstītā metode nedos labāku jutīguma novērtējumu. □

## 3.5. Funkcijas pakāpes atkarība no tās simetrizācijas polinoma

Iepriekšējās nodaļās redzējām, ka $\deg(q) = \deg(p^{sym}) \leq \deg(p)$. Tomēr pastāv arī citas iespējas novērtēt sākotnējā polinoma $p$ pakāpi no apakšas, vai pat noteikt to precīzi.

### *3.5.1. Gadījuma* $\deg(p) = n$ *viennozīmīga noteikšana*

Definēsim $X_1^{even} = \{x \mid |x| \text{ ir pāra un } f(x) = 1\}$ un līdzīgi definēsim $X_1^{odd}$ nepāra vienīnieku skaitam, $X_1^{odd} = \{x \mid |x| \text{ ir nepāra un } f(x) = 1\}$. Apzīmēsim $X_1 = X_1^{even} \cup X_1^{odd}$.

**Teorēma 17.** $\deg(f) = n \iff |X_1^{even}| \neq |X_1^{odd}|$ *[3].*

**Pierādījums.** Apskatīsim unikālo polinomu $p = \sum_S c_S X_S$, kas reprezentē Būla funkciju $f$, kur $c_S$ ir monoma $X_S = \prod_{i \in S} x_i$ koeficients. Tad no Mebiusa inversijas formulas seko $c_S = \sum_{T \subseteq S} (-1)^{|S|-|T|} f(T)$, kur $f(T)$ ir funkcijas $f$ rezultāts uz ievaddatiem, kam ar 1 ir vienādi tie un tikai tie mainīgie, kuru kārtas numuri pieder kopai $T$.

Izvēloties $S = \{1, ..., n\}$, iegūstam $c_S = \sum_{T \subseteq S} (-1)^{|S|-|T|} f(T) = (-1)^n \sum_{x \in X_1} (-1)^{|x|} = (-1)^n(|X_1^{even}| - |X_1^{odd}|)$. Tā kā gadījums $\deg(f) = n$ izpildās tad un tikai tad, ja monoma $x_1...x_n$ koeficients nav nulle, tad iegūstam šīs lemmas formulējumu. [3] □

**Piezīme.** Alternatīvi monoma $x_1...x_n$ koeficienta formulu varējām pārveidot par $c_S = \sum_{k=0}^{n} \sum_{|T|=k} (-1)^{n-k} f(T) = \sum_{k=0}^{n} (-1)^{n-k} r(k)$. Tad 17. teorēmas formulējums sekotu uzreiz pēc šī pārveidojuma.

**Piemērs.** Izmantosim 3.3. tabulā aprēķinātās vērtības $r_f(k) = q_f(k)\binom{n}{k}$. Iegūstam $X_1^{even} = r_f(0) + r_f(2) + r_f(4) = 4$ un $X_1^{odd} = r_f(1) + r_f(3) = 4$. Tā kā $|X_1^{even}| = |X_1^{odd}|$, tad varam apgalvot, ka $\deg(f) < n = 4$. Tik tiešām funkcijas $f$ (1.1) pakāpe ir mazāka par 4 ($\deg(f) = 3$). □

### *3.5.2. Pakāpes apakšējais novērtējums, izmantojot mainīgo ietekmi*

**Definīcija.** Par mainīgā $x_i$ ietekmi uz Būla funkciju $f$ saucam

$$Inf_i(f) = Pr[f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \neq f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n)],$$

kur vērtības $x_1, ..., x_{i-1}, x_{i+1}, ..., x_n$ ir nejauši izvēlētas no kopas $\{0, 1\}$.

**Lemma 18.** *Ja $f$ ir $n$ mainīgo Būla funkcija $f$, tad* $\sum_{i=1}^{n} Inf_i(f) \leq \deg(f)$ *[7].*

Pārveidosim ietekmju summas izteiksmi mums ērtākā veidā:
$\sum_{i=1}^{n} Inf_i(f) = \sum_{i=1}^{n} \frac{|\{x \mid x_i = 1, f(x^i) \neq f(x)\}|}{2^{n-1}} = \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{|\{x \mid |x| = k, x_i = 1, f(x^i) \neq f(x)\}|}{2^{n-1}} =$
$\sum_{k=1}^{n} \frac{g(k)}{2^{n-1}}$, kur $g(k) = \sum_{i=1}^{n} |\{x \mid |x| = k, x_i = 1, f(x^i) \neq f(x)\}|$.

Tātad $g(k)$ ir vienāds ar gadījumu skaitu, kad patvaļīgam $(k-1)$-vieninieku ievaddatu kortežam, pamainot kādu elementu no 0 uz 1, attiecīgais funkcijas rezultāts arī mainās uz pretējo.

**Teorēma 19.** $g(k) \geq |r(k) \cdot k - r(k-1) \cdot (n-k+1)|$

**Pierādījums.** Eksistē $r(k-1)$ dažādi korteži $x \in \{0,1\}^n$, kuriem izpildās $|x| = k-1$ un $f(x) = 1$. Katram no šādiem kortežiem varam $(n-k+1)$ veidos nomainīt kādu nulles vērtību uz vieninieku. Kopumā $A = r(k-1)(n-k+1)$ šādas situācijas.

Eksistē $r(k)$ dažādi korteži $x' \in \{0,1\}^n$, kuriem $|x'| = k$ un $f(x') = 1$. Katru no šādiem kortežiem varam iegūt precīzi $k$ veidos, kādam $(k-1)$-vieninieku kortežam nomainot 0-bitu uz 1-bitu. Kopumā $B = r(k)k$ šādas atbilstības.

Vismazākā iespējamā $g(k)$ vērtība (sliktākais gadījums) ir sasniedzama tad, ja pēc iespējas daudzos gadījumos nulles nomaiņa uz vieninieku kortežā $x$ nemaina funkcijas rezultāta vērtību.

Ja $A \geq B$, tad vismaz $A - B$ gadījumos, nomainot kādu $x$ vērtību no 0 uz 1, no $x$ iegūstam $y \in \{0,1\}^n$, kur $f(x) = 1$ un $f(y) = 0$. Savukārt ja $A < B$, tad vismaz $B - A$ gadījumos, nomainot kādu $x'$ vērtību no 1 uz 0, no $x'$ iegūstam $z \in \{0,1\}^n$, kur $f(z) = 0$ un $f(x') = 1$. Tātad $g(k) \geq |A - B|$. □

Izmantojot 18. lemmu un 19. teorēmu, varam noformulēt Būla funkcijas pakāpes atkarību no simetrizācijas polinoma.

**Sekas 20.** *Ja $f$ ir $n$ mainīgo Būla funkcija $f$, tad*

$$\deg(f) \geq \frac{\sum_{k=1}^{n} |r(k) \cdot k - r(k-1) \cdot (n-k+1)|}{2^{n-1}}.$$

**Piemērs.** Iepriekš noskaidrojām, ka funkcijas $f$ (1.1) (ar mainīgo skaitu $n = 4$) pakāpe ir $deg(f) = 3$, bet viena mainīgā simetrizācijas polinoma vērtības ir $r(k) = (0, 2, 4, 2, 0)$.

Tātad $\deg(f) \geq \frac{|2 \cdot 1 - 0| + |4 \cdot 2 - 2 \cdot 3| + |2 \cdot 3 - 4 \cdot 2| + |0 - 2 \cdot 1|}{2^{4-1}} = \frac{2+2+2+2}{8} = 1$. Redzam, ka arī šoreiz iegūtā likumsakarība var dot diezgan neoptimālus novērtējumus. □

### *3.5.3. Būla funkcijas pakāpes atkarība no tās nulles punktu skaita*

**Definīcija.** Par $n$ argumentu Būla funkcijas $f$ nulles punktiem sauc tādus ievaddatu kortežus $x \in \{0, 1\}^n$, kuriem $f(x) = 0$.

Šajā nodaļā apskatīsim teorēmu, kas ierobežo Būla funkcijas pakāpi atkarībā no šīs funkcijas nulles punktu skaita.

**Teorēma 21.** *Ja $p$ ir $d$. pakāpes $n$ mainīgo polinoms galīgā laukā ar $z$ elementiem, tad risinājumu skaits šī polinoma vienādībai $p(x_1, ..., x_n) = 0$ dalās ar $z^{\lceil \frac{n}{d} \rceil - 1}$ [1].*

Pārformulēsim šo teorēmu tā, lai no pretrunas iegūtu polinoma pakāpes apakšējo novērtējumu.

**Sekas 22.** *Ja $p$ ir $d$. pakāpes $n$ mainīgo polinoms, kas reprezentē Būla funkciju un kuram risinājumu $p(x_1, ..., x_n) = 0$ skaits nedalās ar $2^m$, tad $\left\lceil \frac{n}{d} \right\rceil - 1 < m$ jeb šī polinoma pakāpe ir $d > \frac{n}{m+1}$.*

**Piemērs.** Būla funkcijai $f$ (1.1) mainīgo skaits ir $n = 4$ un pakāpe ir $deg(f) = 3$. Šīs funkcijas rezultāts ir vienāds ar 0 precīzi 8 gadījumos.

Mazākā divnieka pakāpe, ar kuru nedalās šīs funkcijas nulles punktu skaits, ir $2^4$. Tātad $\deg(f) > \frac{4}{4+1}$ jeb $\deg(f) \geq 1$, kas diemžēl ir diezgan neoptimāls funkcijas pakāpes novērtējums. □

# 4. SIMETRIZĀCIJAS POLINOMU PĀRLASE

## 4.1. Simetrizācijas polinomu pārlases motivācija

Mēs vēlamies atrast 5. pakāpes Būla funkciju $f$ ar determinēto vaicājumu sarežģītību $D(f) \geq 14$. No 2. teorēmas zinām, ka $n \geq D(f) \geq 14$. Taču priekš mainīgo skaita $n = 14$ eksistē jau $2^{2^{14}} \approx 10^{4932}$ dažādas Būla funkcijas. Tāpēc varbūtība šādu funkciju atrast ar pilnu pārlasi ir maza.

Meklējamai funkcijai $f$ atbilst kāds simetrizācijas polinoms $p^{sym}$. Iespējamo polinomu $p^{sym}$ skaits ir mazāks par Būla funkciju $f$ skaitu, jo dažādām Būla funkcijām var atbilst viens un tas pats simetrizācijas polinoms. Līdz ar to mēģināsim atrast tādus simetrizācijas polinomus, kuri varētu atbilst $n \geq 14$ mainīgo Būla funkcijai $f$ ar pakāpi $\deg(p^{sym}) \leq \deg(f) = 5$.

Atcerēsimies 4. teorēmu, kas pierāda $s(f) < D(f)$. Lai ierobežotu $D(f) \geq 14$, mēs apskatīsim tikai Būla funkcijas ar $s(f) \geq 14$. Šādi tiks apskatīta tikai daļa no meklējamai funkcijai atbilstošiem simetrizācijas polinomiem, jo funkcijas jutīgums varētu būt arī mazāks par 14. Tomēr katra no 2.1. nodaļā apskatītām 2., 3. un 4. pakāpju funkcijām savu determinēto sarežģītību sasniedz pie $s(f) = D(f) = n$. Var cerēt, ka arī priekš meklējamās 5. pakāpes eksistē Būla funkcija ar tādu jutīgumu, kas ir vienāds ar funkcijas mainīgo skaitu. Izmantojot $s(f)$ kā kritēriju, ir izredzes uzlabot novērtējumu arī lielākajam iespējamajam attālumam starp $\deg(f)$ un $s(f)$.

## 4.2. Interpolācijas izmantošana polinomu pārlasei

Ģenerēsim visus viena mainīgā simetrizācijas polinomus $q$, kas varētu atbilst 5. pakāpes Būla funkcijai $f$ ar jutīgumu $s(f) \geq 14$. Tad katram no šiem polinomiem mēģināsim atrast īpašības, kas vai nu ļautu pierādīt, ka Būla funkciju ar meklējamiem parametriem neeksistē, vai nu atvieglotu šādu funkciju turpmāku meklešanu. Viena mainīgā simetrizācijas polinomus šim nolūkam pārlasīt pirmais piedāvāja A.Lučko [5].

Teorēma 13 apgalvo, ka $s(f) \geq 14$ izpildās visām Būla funkcijām, kurām atbilst viena mainīgā simetrizācijas polinoms $q$ ar $q(0)\binom{n}{0} = 0$ un $q(1)\binom{n}{1} \geq 14$. Savukārt 14. teorēma parāda, ka jebkuru Būla funkciju $f$ ar $s(f) \geq 14$ var pārveidot tā, lai tai atbilstošais polinoms $q$ apmierinātu šos kritērijus.

Mēs zinām, ka $d$. pakāpes polinoms $q$ ir viennozīmīgi definējams ar tā vērtībām $d+1$ dažādos punktos $\{\{x_0, y_0\}, ..., \{x_d, y_d\}\}$, kur visiem $0 \leq i \leq d$ izpildās $q(x_i) = y_i$. Polinomu $q$ iegūsim ar Lagranža interpolāciju:

$$q(x) = \sum_{j=0}^{d} y_j \prod_{\substack{0 \leq i \leq d \\ i \neq j}} \frac{x - x_i}{x_j - x_i}.$$

Tātad 5. pakāpes polinomu ģenerēšanai pietiek pārlasīt šo polinomu iespējamās vērtības 6 punktos, no kuriem divos punktos mēs vērtības varam nofiksēt kā $q(0) = 0/\binom{n}{0} = 0$ un $q(1) = m/\binom{n}{1} = m/n$ kādai konstantei $m \geq s(f)$, kur $n \geq s(f)$ ir meklētās Būla funkcijas mainīgo skaits.

Viena mainīgā simetrizācijas polinoms, kas atbilst $n$ mainīgo Būla funkcijai $f$, ir definēts $n+1$ punktā. Šī polinoma rezultātam $q(k)$ punktā $k \in \{0, ..., n\}$ eksistē $\binom{n}{k} + 1$ dažādas vērtības, un ir jāizpildās īpašībai $q(k)\binom{n}{k} \in \mathbb{N}_0$ (sk. 12. sekas). Vismazākais vērtību skaits ir punktos $k = 0$ un $k = n$, otrais mazākais punktos $k = 1$ un $k = n-1$, utt. Ģenerējot 5. pakāpes polinomu ar fiksētām vērtībām punktos $k = 0$ un $k = 1$, visizdevīgāk ir pārlasīt pārējās četras vērtības punktos $k \in \{n, n-1, n-2, 2\}$.

## 4.3. Ar interpolāciju iegūstamo simetrizācijas polinomu skaits

Iesākumā dažādiem 5. pakāpes Būla funkcijas jutīgumiem $s(f) \geq 14$ pārbaudīsim, kāds ir viena mainīgā polinomu skaits, kas tiek atrasts mazām $n \geq s(f)$ vērtībām, lai visiem $0 \leq i \leq n$ izpildītos $q(i)\binom{n}{i} \in \mathbb{N}_0$. Ar datora programmu atrasto polinomu skaits ir apskatāms 4.1. tabulā.

*4.1. tabula*

**Atrasto 5. pakāpes $n$ mainīgo polinomu $q$ skaits ar jutīgumu $s(f)$**

| $s(f)\backslash n$ | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 247 | 382 | 629 | 1166 | 2198 | 3880 | 4609 | 5058 | 5656 | 6360 | 7297 | 8074 |
| 15 | 0 | 8 | 24 | 65 | 228 | 625 | 1527 | 2664 | 2984 | 3313 | 3864 | 4483 |
| 16 | 0 | 0 | 0 | 0 | 0 | 2 | 43 | 291 | 1033 | 1290 | 1504 | 1776 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 106 | 354 | 416 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 |

No 4.1.. tabulas vērtībām šajā darbā pievērsīsim uzmanību 8 polinomiem, kuriem izpildās $s(f) = n = 15$. Par labu šādai izvēlei kalpo novērojums, ka visos līdz šim labāka-

jos zināmos gadījumos $s(f)$ un $n$ vērtības sakrīt. Savukārt priekš 5. pakāpes polinomiem 15 ir lielākā vērtība, kurai vēl eksistē šāda veida polinomi, jo priekš $s(f) = n = 16$ vairs nav atrasts neviens derīgs viena mainīgā simetrizācijas polinoms.

*4.2. tabula*

**Atrasto 5. pakāpes $s(f) = n = 15$ simetrizācijas polinomu vērtības $r(k)$**

| $N\backslash k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. | 0 | 15 | 104 | 270 | 270 | 6 | 280 | 1953 | 4020 | 4395 | 2832 | 1040 | 174 | 0 | 0 | 1 |
| 2. | 0 | 15 | 104 | 271 | 280 | 51 | 400 | 2163 | 4272 | 4605 | 2952 | 1085 | 184 | 1 | 0 | 1 |
| 3. | 0 | 15 | 105 | 280 | 315 | 126 | 490 | 2205 | 4230 | 4515 | 2877 | 1050 | 175 | 0 | 0 | 1 |
| 4. | 0 | 15 | 105 | 281 | 325 | 171 | 610 | 2415 | 4482 | 4725 | 2997 | 1095 | 185 | 1 | 0 | 1 |
| 5. | 0 | 15 | 105 | 277 | 286 | 1 | 175 | 1695 | 3684 | 4137 | 2727 | 1035 | 190 | 7 | 1 | 1 |
| 6. | 0 | 15 | 105 | 278 | 296 | 46 | 295 | 1905 | 3936 | 4347 | 2847 | 1080 | 200 | 8 | 1 | 1 |
| 7. | 0 | 15 | 105 | 279 | 306 | 91 | 415 | 2115 | 4188 | 4557 | 2967 | 1125 | 210 | 9 | 1 | 1 |
| 8. | 0 | 15 | 105 | 277 | 287 | 11 | 220 | 1815 | 3894 | 4389 | 2937 | 1155 | 235 | 17 | 2 | 1 |

Svarīgi ir arī tas, lai izvēle krīt uz salīdzinoši mazu $n$ vērtību, jo līdz ar to ir vairāk izredžu veiksmīgi izpētīt konkrēto gadījumu ar datora apstrādes palīdzību, pēc iespējas izvairoties no atmiņas un ātrdarbības problēmām. Visi atrastie 5. pakāpes simetrizācijas polinomi $r$ ar $s(f) = n = 15$ ir apskatāmi 4.2. tabulā.

## 4.4. Būla funkcijas likumsakarību salīdzinājums

Iepriekšējās nodaļās apskatījām dažādas likumsakarības starp Būla funkcijas pakāpi, jutīgumu, determinēto vaicājumu sarežģītību, mainīgo skaitu un viena mainīgā simetrizācijas polinomu (sk. 4.3. tab.). Šajā nodaļā visas apskatītās likumsakarības savā starpā salīdzināsim.

Ģenerēsim visus iespējamos simetrizācijas polinomus, kuru pakāpe nepārsniedz 5 un kuru mainīgo skaits nepārsniedz 15 (bez ierobežojuma uz $s(f)$ vērtību). Katram no simetrizācijas polinomiem pielietosim visas iepriekš apskatītās likumsakarības, salīdzinot to rezultātus savā starpā. Mūsu mērķis ir atsijāt vājākās likumsakarības, kas uz citu likumsakarību fona nedod nekādus uzlabojumus.

Starp visām likumsakarībām, sākumā aprēķināsim Būla funkcijas jutīguma novērtējumu, tad pakāpes novērtējumu (tai skaitā izmantojot $\deg(f) \geq \sqrt{s(f)/2}$), un visbeidzot funkcijas determinētās sarežģītības novērtējumu, kas var būt atkarīga gan no pakāpes ($\deg(f) \leq D(f)$), gan no jutīguma ($s(f) \leq D(f)$).

*4.3. tabula*

**Likumsakarības starp Būla funkcijas raksturlielumiem**

| $N$ | Formula | Teorēma |
|---|---|---|
| 1. | $\deg(f) \leq D(f)$ | 3 |
| 2. | $s(f) \leq D(f)$ | 4 |
| 3. | $\deg(f) \geq \sqrt{s(f)/2}$ | 5 |
| 4. | $\deg(f) \geq \deg(p^{sym})$ | – |
| 5. | $s(f) \geq n - \left\lfloor \frac{r(k-1)\min(n-k+1, r(k)) + r(k+1)\min(k+1, r(k))}{r(k)} \right\rfloor$ | 15 |
| 6. | $s(f) \geq n - ...$ (iepriekšējās formulas duālais gadījums) | 16 |
| 7. | $\deg(f) = n \iff \lvert X_1^{even}\rvert \neq \lvert X_1^{odd}\rvert$ | 17 |
| 8. | $\deg(f) \geq \frac{\sum_{k=1}^{n} \lvert r(k)\cdot k - r(k-1)\cdot(n-k+1)\rvert}{2^{n-1}}$ | 20 |
| 9. | $2^m \nmid \lvert x : p(x_1, ..., x_n) = 0\rvert \Rightarrow \deg(f) > \frac{n}{m+1}$ | 22 |

Kopumā ir iegūstami 1958008 simetrizācijas polinomi ar pakāpi $1 \leq \deg(q) \leq 5$ un mainīgo skaitu, kas nepārsniedz 15 un nav mazāks par konkrētajā brīdī apskatīto pakāpi. Katram no šiem simetrizācijas polinomiem pielietojam visas 4.3. tabulā uzskatītās likumsakarības, lai novērtētu polinomam atbilstošās Būla funkcijas $f$ raksturlielumu vērtības $\deg(f)$, $s(f)$, $D(f)$. Tad katrai no likumsakarībām saskaitām, cik gadījumos (no iespējamiem 1958008) tā dod optimālo rezultātu, salīdzinot ar citām apskatītām likumsakarībām. Iegūtie rezultāti ir apkopoti 4.4. tabulas otrajā kolonnā.

*4.4. tabula*

**Simetrizācijas polinomu skaits, kuriem konkrētā likumsakarība ir optimāla**

| | | |
|---|---|---|
| Būla funkcijas jutīgums $s(f)$ | | |
| $s(f) \geq n - \left\lfloor \frac{r(k-1)\min(n-k+1, r(k)) + r(k+1)\min(k+1, r(k))}{r(k)} \right\rfloor$ | 1323854 | <u>1323854</u> |
| $s(f) \geq n - ...$ (iepriekšējās formulas duālais gadījums) | 1323854 | 1107701 |
| Būla funkcijas pakāpe $\deg(f)$ | | |
| $\deg(f) \geq \deg(p^{sym})$ | 1958008 | <u>1958008</u> |
| $\deg(f) = n \iff \lvert X_1^{even}\rvert \neq \lvert X_1^{odd}\rvert$ | 16802 | 0 |
| $2^m \nmid \lvert x : p(x_1, ..., x_n) = 0\rvert \Rightarrow \deg(f) > \frac{n}{m+1}$ | 528 | 0 |
| $\deg(f) \geq \sqrt{s(f)/2}$ | 400 | 0 |
| $\deg(f) \geq \frac{\sum_{k=1}^{n} \lvert r(k)\cdot k - r(k-1)\cdot(n-k+1)\rvert}{2^{n-1}}$ | 268 | 0 |
| Būla funkcijas determinētā vaicājumu sarežģītība $D(f)$ | | |
| $s(f) \leq D(f)$ | 1665952 | <u>1665442</u> |
| $\deg(f) \leq D(f)$ | 552840 | 292056 |

Daudziem polinomiem kāda raksturlieluma lielākā vērtība var tikt iegūta vairākos veidos. Atradīsim mazāko nepieciešamo likumsakarību skaitu, kas garantē to, ka katram polinomam tiek sasniegti visi labākie novērtējumi. Katram no trim apskatītiem raksturlielumiem (jutīgums, pakāpe, vaicājumu sarežģītība) piefiksēsim to likumsakarību, kas sasniedz lielāko optimālo novērtējumu skaitu. Visām pārējām likumsakarībām skaitīsim tikai tādus gadījumus, kas netiek sasniegti ar attiecīgā veida nofiksēto likumsakarību (sk. 4.4. tab. 3. kolonnu, kur fiksētām likumsakarībām atbilstošo optimālo gadījumu skaits ir pasvītrots).

Kā redzams, priekš Būla funkcijas pakāpes novērtējuma mums pietiek ņemt vērtā tikai nosacījumu $\deg(f) \geq \deg(p^{sym})$, bet visas pārējās likumsakarības nekad nesniedz papildus informāciju. Savukārt gan Būla funkcijas jutīguma, gan determinētās vaicājumu sarežģītības noteikšanai ir jāapskata abas attiecīgā veida likumsakarības, jo eksistē brīži, kad jebkura no tām ir labāka par otru.

# 5. SECĪGA SIMETRIZĀCIJAS POLINOMU SADALĪŠANA

## 5.1. Simetrizācijas polinoma sadalīšana divos mazākos

Apskatīsim $n$ mainīgo polinomu $p$ ar pakāpi $\deg(p)$, kas reprezentē meklējamo Būla funkciju $f$. Jebkuru šādu polinomu var izteikt kā

$$p(x_1, ..., x_{n-1}, x_n) = A + x_n B,$$

kur $A$ un $B$ ir polinomi, kas nesatur mainīgo $x_n$.

**Definīcija.** Ja $p$ ir $n$ mainīgo polinoms, tad ar $p_{a_1 a_2 ... a_k}$ apzīmēsim šo polinomu ar $k$ fiksētām mainīgo vērtībām: $p(x_1, ..., x_{n-k}, x_{n-k+1} = a_k, x_{n-k+2} = a_{k-1}, ..., x_n = a_1)$, kur $a_i \in \{0, 1\}$. Savukārt ar $r_{a_1 a_2 ... a_k}$ apzīmēsim polinomam $p_{a_1 a_2 ... a_k}$ atbilstošo simetrizācijas polinomu.

Augstāk minētai izteiksmei iegūstam $p_0 = A$ un $p_1 = A + B$. Sākotnējo polinomu $p$ var izteikt ar $p_0$ un $p_1$: $p = (1 - x_n)p_0 + x_n p_1$. Starp simetrizācijas polinomiem $r$, $r_0$ un $r_1$, kas atbilst polinomiem $p$, $p_0$ un $p_1$, pastāv sekojošas likumsakarības:

- $r(0) = r_0(0)$
- $\forall k \in \{1, ..., n-1\} : r(k) = r_0(k) + r_1(k-1)$
- $r(n) = r_1(n-1)$

Katrā no gadījumiem $r(k)$ apzīmē tādu ievaddatu kortežu $x \in \{0, 1\}^n$ skaitu, kuriem izpildās $|x| = k$ un $p(x) = 1$. Līdz ar to mēs šķirojam divas iespējas: vai nu $x_n = 0$ un mūs aizvien interesē koreteži ar $k$ vieniniekiem (to skaits ir $r_0(k)$), vai arī $x_n = 1$ un mums turpmāk jāskaita kortežus ar $k - 1$ vieniniekiem (to skaits ir $r_1(k-1)$).

**Piemērs.** Būla funkciju (1.1) reprezentē polinoms (1.2):

$$p(x_1, x_2, x_3, x_4) = x_2 + x_3 - x_1 x_3 - x_2 x_3 + x_1 x_2 x_3 - x_2 x_3 x_4.$$

Šo polinomu varam izteikt kā $p = p_0 + x_4 \cdot p_1$, kur

$$p_0(x_1, x_2, x_3) = x_2 + x_3 - x_1 x_3 - x_2 x_3 + x_1 x_2 x_3$$

un

$$p_1(x_1, x_2, x_3) = x_2 + x_3 - x_1 x_3 - 2x_2 x_3 + x_1 x_2 x_3.$$

Atbilstošie simetrizācijas polinomi $r$, $r_0$ un $r_1$ ir apskatāmi 5.1. tabulā. Iekrāsotās tabulas šūnas norāda uz likumsakarību $r(k) = r_0(k) + r_1(k-1)$. □

*5.1. tabula*

**Polinomam** (1.2) **atbilstošā simetrizācijas polinoma** $r$ **sadalījums** $r_0$ **un** $r_1$

| $k$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $r(k)$ | 0 | 2 | 4 | 2 | 0 |
| $r_0(k)$ | 0 | 2 | 2 | 1 | |
| $r_1(k)$ | 0 | 2 | 1 | 0 | |

## 5.2. Polinoma raksturlielumu izmaiņas sadalīšanas procesā

Pieņemsim, ka $n$ mainīgo multilineārs polinoms $p : \{0,1\}^n \to \{0,1\}$ ar pakāpi $\deg(p)$ un jutīgumu $s(f)$ tiek sadalīts sīkākos polinomos $p_0 = p(x_1, ..., x_{n-1}, x_n = 0)$ un $p_1 = p(x_1, ..., x_{n-1}, x_n = 1)$. Apskatīsim, kāds var būt iegūto polinomu $p_0$ un $p_1$ mainīgo skaits, pakāpe, jutīgums un determinētā vaicājumu sarežģītība. Šim nolūkam mums pietiks apskatīt divus piemērus.

**Piemērs.** Izvēlēsimies sakotnējo polinomu vienādu ar $p = x_1...x_n$. Atkarībā no mainīgā $x_n$ vērtības iegūstam divus polinomus: $p_0 = 0$ un $p_1 = x_1...x_{n-1}$.

Polinomiem $p$, $p_0$ un $p_1$ mainīgo skaits, jutīgums, pakāpe un determinētā vaicājumu sarežģītība savā starpā sakrīt. Visu šo raksturlielumu sākotnējā vērtība ir $n$, savukārt pēc sadalījuma tā var palikt par 0 vai $n-1$.

Alternatīvi mēs varējām aplūkot gadījumu ar sākotnējo polinomu $p = (1 - x_n) \cdot x_1...x_{n-1}$, kas pēc sadalījuma paliek par $p_0 = x_1...x_{n-1}$ un $p_1 = 0$. □

Šis piemērs parāda, ka polinoma sadalīšanas gaitā visi tā raksturlielumi var svārstīties robežās no 0 līdz $n-1$, kur $n$ ir sākotnējā polinoma mainīgo skaits. Savukārt nākamais piemērs parāda: ja pakāpes un jutīguma vērtības ir mazākas par mainīgo skaitu, tad šīs vērtības sadalīšanas procesā var nemainīties.

**Piemērs.** Apskatām 3 mainīgo polinomu $p = x_1x_2 + x_2x_3 - x_1x_2x_3$. Šī polinoma sadalījums sastāv no $p_0 = x_1x_2$ un $p_1 = x_2$.

*5.2. tabula*

**Polinomam $p = x_1x_2 + x_2x_3 - x_1x_2x_3$ atbilstošā simetrizācijas polinoma $r$ sadalījums**

| $k$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $r(k)$ | 0 | 0 | 2 | 1 |
| $r_0(k)$ | 0 | 0 | 1 | |
| $r_1(k)$ | 0 | 1 | 1 | |

Polinomu $p$ un $p_0$ pakāpes ir vienādas ar 2. Polinoma $p$ jutīgums ir vienāds ar 2 un tiek sasniegts punktā $(0, 1, 0)$, savukārt polinoma $p_0$ jutīgums arī ir vienāds ar 2 un tiek sasniegts punktā $(1, 1)$.

Aplūkosim polinomiem $p$ un $p_1$ atbilstošos simetrizācijas polinomus $r$ un $r_1$ (sk. 5.2. tab.). Lai gan polinoms $p_1$ sastāv no viena mainīgā, mēs šim polinomam uzbūvējām trīs vērtību simetrizācijas polinomu $r_1$. Tā nav pretruna, bet gan šajā gadījumā mēs polinomu $p_1$ uztveram kā divu mainīgo polinomu, kurā viens mainīgais ir fiktīvs. □

Sadalot patvaļīgu polinomu $p$ sīkākos polinomos $p_0$ un $p_1$, iegūto polinomu pakāpes nekad nepārsniedz sākotnējā polinoma pakāpi. Nākamais piemērs parāda, ka atbilstošiem simetrizācijas polinomiem $r$, $r_0$ un $r_1$ analoģiskā īpašība ne vienmēr izpildās [9].

**Piemērs.** Apskatīsim polinomu $p = 1 - x_3 - x_1x_2 + x_1x_3$. Šī polinoma simetrizācijas polinoms ir $p^{sym} = \dfrac{6 - 2x_1 - 2x_2 - 2x_3}{6}$ un viena mainīgā simetrizācijas polinoms ir $q = 1 - \dfrac{2}{6}x$.

Ja $x_3 = 0$, tad no $p$ iegūstam polinomu $p_0 = 1 - x_1x_2$. Polinoma $p_0$ simetrizācijas polinoms ir $p_0^{sym} = \dfrac{2 - 2x_1x_2}{2}$ un viena mainīgā simetrizācijas polinoms ir $q_0 = 1 - \binom{x}{2}$.

Savukārt, ja $x_3 = 1$, tad no $p$ iegūstam polinomu $p_1 = x_1 - x_1x_2$. Šī polinoma simetrizācijas polinoms ir $p_1^{sym} = \dfrac{x_1 + x_2 - 2x_1x_2}{2}$ un viena mainīgā simetrizācijas polinoms ir $q_1 = \dfrac{1}{2}x - \binom{x}{2}$.

Tātad $\deg(q) = 1$, bet $\deg(q_0) = \deg(q_1) = 2$. □

## 5.3. Secīga simetrizācijas polinomu sadalīšana

Mēs vēlamies atrast polinomu $p$, kurš atbilst simetrizācijas polinomam $r$. Šim nolūkam sadalīsim polinomu $r$ divos mazākos simetrizācijas polinomos $r_0$ un $r_1$, atradīsim tiem atbilstošos polinomus $p_0$ un $p_1$, un no tiem atjaunosim sākotnējo polinomu $p$.

Simetrizācijas polinomam $r$ varētu atbilst vairāki sadalījuma pāri $r_0$ un $r_1$. Tāpēc pārlasīsim visus iespējamos simetrizācijas polinomus $r_0$, un katram no tiem viennozīmīgi uzbūvēsim atbilstošo simetrizācijas polinomu $r_1$, izmantojot 5.1. nodaļā aprakstītās formulas: $\forall k \in \{0, ..., n-2\} : r_1(k) = r(k+1) - r_0(k+1)$ un $r_1(n-1) = r(n)$.

Vērtību $r_0(0) = r(0)$ izvēlamies viennozīmīgi, bet jebkuru citu simetrizācijas polinoma $r_0$ vērtību punktā $1 \leq k \leq n-1$ izvēlamies tā, lai izpildītos sekojoši kritēriji:

- $0 \leq r_0(k) \leq \binom{n-1}{k}$
- $0 \leq r_1(k) = r(k+1) - r_0(k+1) \leq \binom{n-1}{k}$, kuru pārveidojam par $r(k) - \binom{n-1}{k-1} \leq r_0(k) \leq r(k)$

Tātad $\max\left(0, r(k) - \binom{n-1}{k-1}\right) \leq r_0(k) \leq \min\left(\binom{n-1}{k}, r(k)\right)$.

Ja meklējam $n$ mainīgo Būla funkciju $f$ ar pakāpi $\deg(f)$, tad sākumā atrodam visus simetrizācijas polinomus $r$, kuri varētu atbilst funkcijai $f$. No simetrizācijas polinoma $r$ atrodam visus iespējamos sadalījumus polinomos $r_0$ un $r_1$. Tad katru no atrastajiem sadalījumiem $r_0$ un $r_1$ sadalām vēl sīkāk, iegūstot simetrizācijas polinomos $r_{00}, r_{01}, r_{10}, r_{11}$.

Šo procesu atkārtosim secīgi, katru reizi iegūstot divas reizes vairāk simetrizācijas polinomu, katrs no kuriem satur par vienu mainīgo mazāk nekā sadalāmais polinoms. Mēs apstājamies tad, ja polinomu $r$ ir izdevies sadalīt $n$ secīgas reizes, kā rezultātā esam ieguvuši $2^n$ simetrizācijas polinomus, katrs no kuriem atbilst 0 mainīgo polinomam jeb konstantei $c \in \{0, 1\}$.

**Definīcija.** Par Būla funkcijai $f$ atbilstošā simetrizācijas polinoma $k$-to sadalījuma līmeni sauksim $2^k$ simetrizācijas polinomu komplektu $(r_{00..00}, r_{00..10}, ..., r_{11..10}, r_{11..11})$, kas atbilst funkcijai $f$, $2^k$ dažādos veidos nofiksējot tajā mainīgos $\{x_{n-k+1}, x_{n-k+2}, ..., x_{n-1}, x_n\}$.

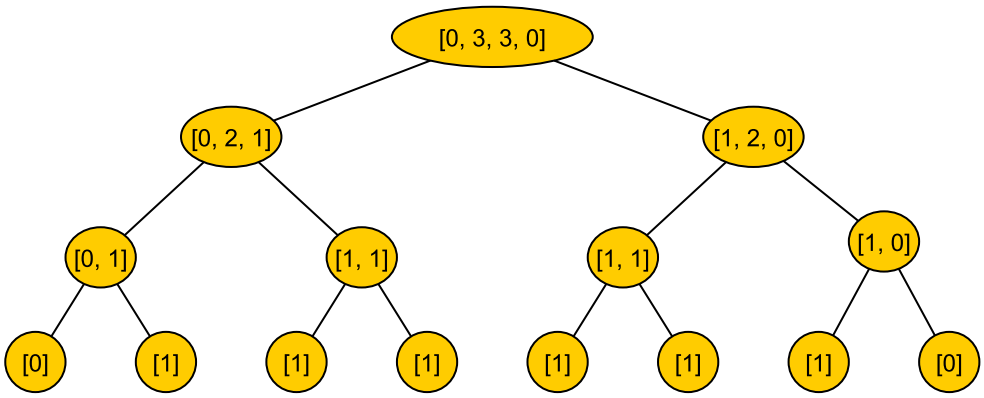
Jāatzīmē, ka mums nav jāpārlasa polinoma $r_0$ vērtības visos $n-1$ punktos. Ja simetrizācijas polinoms $r$ atbilst Būla funkcijai ar pakāpi $\leq d$ (mūsu gadījumā parasti $d = 5$) un mainīgo skaitu $n$, tad simetrizācijas polinoms $r_0$ atbilst Būla funkcijai ar pakāpi $\leq d_0 = \min(d, n-1)$. Līdz ar to pietiek pārlasīt vērtības $d_0 + 1$ punktos un interpolēt pārējās vērtības.

**Definīcija.** Ja $n$ mainīgo Būla funkcijai atbilst simetrizācijas polinoms $r$, tad par šī polinoma triviālu sadalījumu sauksim $n$-to sadalījuma līmeni.

**Piemērs.** Apskatīsim Nisan un Szegedy funkciju (sk. 2.1.1. nod.):

$$E(x_1, x_2, x_3) = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3.$$

Šai funkcijai atbilstošā simetrizācijas polinoma $r$ vērtības ir $[0, 3, 3, 0]$. Polinoma $r$ triviāls sadalījums ir apskatāms 5.1. attēlā, kur katras simetrizācijas polinoma $r_x$ virsotnes kreisais bērns ir $r_{x0}$ un labais bērns ir $r_{x1}$. □




*5.1. att.* **Nisan un Szegedy funkcijas** (2.1) **triviāls sadalījums.**

## 5.4. Būla funkcijas iegūšana no pilnīga sadalījuma

Secīgi sadalot simetrizācijas polinomus, mūsu mērķis ir atrast triviālu sadalījumu. Triviālā sadalījuma zemākais $n$-tais līmenis satur tikai tādus simetrizācijas polinomus $r_i$, kas ir definēti vienā punktā un atbilst konstantai Būla funkcijai $f_i$. Šo funkciju reprezentē polinoms $p_i = r_i(0)$.

Tātad simetrizācijas polinoma $r$ triviālais sadalījums ir vienāds ar meklējamās Būla funkcijas reprezentējošā polinoma $p$ sīkāko (jeb triviālo) sadalījumu. Atliek iemācīties, kā no diviem polinomiem $p_{x0}$ un $p_{x1}$ (kas atbilst simetrizācijas polinomiem $r_{x0}$ un $r_{x1}$) uzbūvēt polinomu $p_x$ (kas atbilst simetrizācijas polinomam $r_x$). Vairākkārt pielietojot šādu algoritmu, spēsim no triviālā sadalījuma atjaunot sākotnējo polinomu $p$.

Ja sadalījuma $k$-tā līmeņa polinomu $p_x$ sadalījām divos sīkākos, tad polinomam $p_{x0}$ atbilst mainīgā vērtība $x_{n-k} = 0$ un polinomam $p_{x1}$ atbilst mainīgā vērtība $x_{n-k} = 1$. Kā jau redzējām 5.1. nodaļā, polinomu $p_x$ varam izrēķināt kā $p_x = (1 - x_{n-k})p_{x0} + x_{n-k}p_{x1}$. Ja polinoms $p_0$ atbilst simetrizācijas polinoma $r_{x0}$ īpašībām, un polinoms $p_{x1}$ atbilst si-

metrizācijas polinoma $r_{x1}$ īpašībām, tad acīmredzami polinoms $p_x$ atbilstīs simetrizācijas polinoma $r_x$ īpašībām.

**Piemērs.** Izmantosim 5.1. attēlā redzamo Nisan un Szegedy funkcijas simetrizācijas polinoma triviālo sadalījumu, lai atjaunotu Nisan un Szegedy funkcijai atbilstošo polinomu. Katram no simetrizācijas polinomiem $r_x$ (sākot ar zemāko sadalījuma līmeni) 5.3. tabulā minēti simetrizācijas polinomi $r_{x0}$, $r_{x1}$ un parādīta attiecīgā polinoma $p_x$ iegūšana no polinomiem $p_{x0}$ un $p_{x1}$. □

*5.3. tabula*

**Nisan un Szegedy polinoma atjaunošana no triviāla sadalījuma ($n = 3$)**

| $k$ | $r_x$ | $r_{x0}$ | $r_{x1}$ | $p_x = (1 - x_{n-k})p_{x0} + x_{n-k}p_{x1}$ |
|---|---|---|---|---|
| 3 | $[0]$ | – | – | $0$ |
| 3 | $[1]$ | – | – | $1$ |
| 2 | $[0, 1]$ | $[0]$ | $[1]$ | $(1 - x_1) \cdot 0 + x_1 \cdot 1 = x_1$ |
| 2 | $[1, 1]$ | $[1]$ | $[1]$ | $(1 - x_1) \cdot 1 + x_1 \cdot 1 = 1$ |
| 2 | $[1, 0]$ | $[1]$ | $[0]$ | $(1 - x_1) \cdot 1 + x_1 \cdot 0 = 1 - x_1$ |
| 1 | $[0, 2, 1]$ | $[0, 1]$ | $[1, 1]$ | $(1 - x_2) \cdot x_1 + x_2 \cdot 1 = x_1 + x_2 - x_1x_2$ |
| 1 | $[1, 2, 0]$ | $[1, 1]$ | $[1, 0]$ | $(1 - x_2) \cdot 1 + x_2 \cdot (1 - x_1) = 1 - x_1x_2$ |
| 0 | $[0, 3, 3, 0]$ | $[0, 2, 1]$ | $[1, 2, 0]$ | $(1 - x_3) \cdot (x_1 + x_2 - x_1x_2) + x_3(1 - x_1x_2) = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3$ |

Iepriekšējais piemērs parādīja, kā atjaunot sākotnējo polinomu no triviāla sadalījuma. Tomēr ne vienmēr tas ir iespējams. Lai gan uzbūvētais polinoms $p$ vienmēr atbilst nepieciešamajām simetrizācijas polinoma $q$ īpašībām, tomēr polinoma $p$ atjaunošanas procesā iegūtā pakāpe un mainīgo skaits bieži vien atšķirsies no vēlamā.

**Piemērs.** Polinomiem $p_a = x_1 - x_1x_2$ un $p_b = x_2 - x_1x_2$ atbilst viens un tas pats simetrizācijas polinoms $r_{ab}$ ar vērtībām $[0, 1, 0]$ Apskatīsim, kāds polinoms var atbilst simetrizācijas polinomam $r$ ar vērtībām $[0, 1, 1, 0]$, kas var tikt iegūts no $r_0 = r_1 = r_{ab}$.

- $p_u = (1 - x_2)p_a + x_2p_a = x_1 - x_1x_2$
- $p_v = (1 - x_2)p_a + x_2p_b = x_1 - x_1x_2 - x_1x_3 + x_2x_3$

Savā starpā pārsaucot polinomu $p_a$ un $p_b$ mainīgos, rezultātā simetrizācijas polinomam $[0, 1, 1, 0]$ uzbūvējam vai nu 2 mainīgo polinomu $p_u$, vai nu 3 mainīgo polinomu $p_v$. □

Lai uzbūvētu sākotnējo polinomu $p$ no triviāla sadalījuma, katru reizi apvienojot divus polinomus vienā, mums nāksies fiksēt mainīgo nosaukumus polinomā $p_0$ un visos iespējamos veidos savā starpā mainīt mainīgo nosaukumus polinomā $p_1$. Līdz ar to polinomu $p$ būs iespējams iegūt priekš dažādām pakāpju un mainīgo skaita vērtībām.

Tomēr atbilstoši 5.2. nodaļai mēs nevaram precīzi zināt, kāda ir Būla funkcijas pakāpe vai mainīgo skaits, kas atbilst konkrētajam simetrizācijas polinomam, kas iegūts sadalījuma gaitā. Līdz ar to bieži vien nāksies soli pēc soļa secīgi no sīkākiem polinomiom būvēt sākotnējo polinomu $p$ un tikai pēdējā solī varēs saprast, ka iegūtais polinoms neatbilst vēlamiem Būla funkcijas raksturlielumiem. Pašlaik autoram nav zināms, kā efektīvi ajaunot sākotnējo polinomu no triviāla sadalījuma.

# 6. SECĪGAS POLINOMU SADALĪŠANAS UZLABOJUMI

## 6.1. Pakāpju likumsakarības viena sadalījuma līmeņa ietvaros

Atgriezīsimies pie idejas $n$ mainīgo polinomu $p$ ar pakāpi $\deg(p) \leq d$ izteikt kā

$$p = A + x_n B,$$

kur $A$ un $B$ ir polinomi, kas nesatur mainīgo $x_n$. Tad $\deg(A) \leq d$ un $\deg(B) \leq d-1$.

Polinomi $p_0 = A$ un $p_1 = A + B$ satur saskaitāmo $A$, tāpēc $\deg(p_0)$, $\deg(p_1) \leq d$. Tomēr $\deg(p_1 - p_0) = \deg(B) \leq d-1$. Tātad arī $\deg(p_1^{sym} - p_0^{sym}) \leq \deg(p_1 - p_0) \leq d-1$. Šādi esam ieguvuši jaunu nosacījumu, kuram ir jāizpildās simetrizācijas polinoma sadalīšanas procesā.

**Piezīme.** Šajā nodaļā mēs izmantosim īpašību, ka jebkuriem diviem polinomiem $p_0$ un $p_1$ izpildās $(p_0+p_1)^{sym} = p_0^{sym}+p_1^{sym}$ un $(p_0-p_1)^{sym} = p_0^{sym}-p_1^{sym}$. Šāda īpašība acīmredzami seko no polinomu simetrizācijas metodes definīcijas, jo patvaļīgai ievaddatu permutācijai $x$ izpildās $p_0(x) \pm p_1(x) = (p_0 \pm p_1)(x)$. Tātad arī $\deg(p_0^{sym} \pm p_1^{sym}) = \deg((p_0 \pm p_1)^{sym}) \leq \deg(p_0 \pm p_1)$.

Jebkuru $n$ mainīgo polinomu $p$ ar pakāpi $\deg(p) \leq d$ varam izteikt kā

$$p = A + Bx_n + Cx_{n-1} + Dx_{n-2} + Ex_nx_{n-1} + Fx_nx_{n-2} + Gx_{n-1}x_{n-2} + Hx_nx_{n-1}x_{n-2},$$

kur $A, B, C, D, E, F, G, H$ ir polinomi, kas nesatur mainīgos $x_n, x_{n-1}, x_{n-2}$. Tad 3. līmeņa sadalījumā iegūstam:

- $p_{000} = A$
- $p_{001} = A + D$
- $p_{010} = A + C$
- $p_{011} = A + C + D + G$
- $p_{100} = A + B$
- $p_{101} = A + B + D + F$
- $p_{110} = A + B + C + E$
- $p_{111} = A + B + C + D + E + F + G + H$

Katru no polinomiem $A, B, C, D, E, F, G, H$ varam iegūt sekojošā veidā:

- $A = p_{000}$
- $B = p_{100} - p_{000}$
- $C = p_{010} - p_{000}$
- $D = p_{001} - p_{000}$
- $E = p_{110} - p_{100} - p_{010} + p_{000}$
- $F = p_{101} - p_{100} - p_{001} + p_{000}$
- $G = p_{011} - p_{010} - p_{001} + p_{000}$
- $H = p_{111} - p_{110} - p_{101} - p_{011} + p_{100} + p_{010} + p_{001} - p_{000}$

Līdz ar to varam katrā gadījumā pārbaudīt, ka izpildās:

- $\deg(A) \leq d$
- $\deg(B), \deg(C), \deg(D) \leq d - 1$
- $\deg(E), \deg(F), \deg(G) \leq d - 2$
- $\deg(H) \leq d - 3$

Protams, ja $d-3$ vērtība ir mazāka par 0, tad mums ir īstenībā jāparbauda, ka polinoms $H$ ir konstante 0.

Mēs strādāsim nevis ar sākotnējā polinoma sadalījumiem, bet gan ar atbilstošiem simetrizācijas polinomiem, tomēr īpašība $\deg(p^{sym}) \leq \deg(p)$ ļauj mums pielietot iegūtos kritērijus arī simetrizācijas polinomu pakāpēm.

Vispārināsim iegūtos nosacījumus priekš $k$-tā sadalījuma līmeņa. Katram sadalījuma polinomam $p_{a_1 a_2 \ldots a_k}$ ar $a_i \in \{0, 1\}$ izrēķinam polinomu izteiksmi

$$Z = \sum_{\substack{b_1,\ldots,b_k \in \{0,1\} \\ b_1 \leq a_1, \ldots, b_k \leq a_k}} (-1)^{(\sum a_i - \sum b_i)} \cdot p_{b_1 \ldots b_k}$$

un pārbaudam, ka izpildās nosacījums $\deg(Z) \leq d - \sum a_i$.

Lai pārbaudītu šos kritērijus, mums nav jāģenerē pilns $k$-tā sadalījuma līmenis. Mēs tos varam pārbaudīt pēc katra $k$-tā sadalījuma līmeņa polinoma iegūšanas no $(k-1)$-ā

sadalījuma līmeņa. Šim nolūkam apskatīsim $(k-1)$-ā sadalījuma līmeņa $2^{k-1}$ polinomus $p_{a_1a_2\ldots a_{k-1}}$ pieaugošā secībā pēc to $a = \sum_{i=1}^{k-1} a_i$ vērtības. Piemēram 3. sadalījuma līmeņa polinomus apskatīsim secībā $(p_{000}, p_{001}, p_{010}, p_{100}, p_{011}, p_{101}, p_{110}, p_{111})$ un katru no tiem sadalīsim divos 4. sadalījuma līmeņa polinomos.

Kad sadalām $(k-1)$-ā sadalījuma polinomu $p_{a_1a_2\ldots a_{k-1}}$ ar $a = \sum a_i$ divos $k$-tā sadalījuma līmeņa polinomos $p_{a_1a_2\ldots a_{k-1}0}$ un $p_{a_1a_2\ldots a_{k-1}1}$, tad uzreiz varam abiem polinomiem pārbaudīt šajā nodaļā aprakstīto nosacījumu, jo visi $k$-tā sadalījuma līmeņa polinomi ar $\sum a_i < a$ jau ir iegūti iepriekšējos soļos.

## 6.2. Vienādojumu sistēmas risināšana secīgu sadalījumu gaitā

Ir dots $n$ mainīgo polinoms $p$ un mēs to vēlamies sadalīt divos gadījumos $p_0$ un $p_1$. Iepriekš esam pieņemuši, ka sadalīšana notiek, pielīdzinot mainīgā $x_n$ vērtību kādai konstantei $c \in \{0, 1\}$. Tomēr šim nolūkam mēs varam piešķirt vērtību $c$ jebkuram no šī polinoma mainīgajiem. Kopā eksistē $n$ iespējas to izradīt, un katrā no tām iegūstam polinomu $p_0$ un $p_1$ komplektus, kas savā starpā var būt vienādi vai atšķirīgi.

**Definīcija.** Ja $n$ mainīgo polinomā $p$ pielīdzinam $x_i = c \in \{0, 1\}$, tad iegūto polinomu apzīmēsim ar $p_c^i$, bet tam atbilstošo simetrizācijas polinomu apzīmēsim ar $r_c^i$.

**Teorēma 23.** $\sum_{i=1}^{n} r_1^i(k) = \binom{k+1}{1} r(k+1)$

**Pierādījums.** Vienādojuma kreisā puses saskaita, cik gadījumos mēs varam pielīdzināt polinoma $p$ patvaļīgu mainīgo $x_i = 1$ un tad vieniniekiem pielīdzināt vēl precīzi $k$ citus mainīgos, lai beigās iegūtu rezultātu 1.

Tātad mēs saskaitam, cik kopumā ir iespējas precīzi $k+1$ mainīgos pielīdzināt vieniniekam tā, lai beigās iegūtu polinoma rezultātu 1. Tomēr katru no šādiem gadījumiem mēs esam ieskaitījuši $\binom{k+1}{1}$ reizes, izvēloties pirmo vienienika pozīciju atsevišķi no pārējām $k$ pozīcijām. □

**Teorēma 24.** $\sum_{i=1}^{n} r_0^i(k) = \binom{n-k}{1} r(k)$

**Pierādījums.** Vienādojuma kreisā puses saskaita, cik gadījumos mēs varam pielīdzināt polinoma $p$ patvaļīgu mainīgo $x_i = 0$ un tad precīzi $k$ no palikušiem mainīgajiem pielīdzināt vieniniekiem tādā veidā, lai iegūtā polinoma vērtība būtu vienāda ar 1.

Tomēr mēs šīs darbības varējām veikt otrādā secībā. Vispirms atrast visus gadījumus, kā precīzi $k$ mainīgos pielīdzināt vieniniekam, lai polinoma $p$ vērtībā būtu vienāda ar 1. Katrā no šiem gadījumiem pāri palikušie $n-k$ mainīgie ir vienādi ar 0, un mēs $\binom{n-k}{1}$ veidos varam izvēlēties jebkuru no tiem. □

Iegūtās teorēmas varam pielietot, mēģinot sadalīt patvaļīgu simetrizācijas polinomu $r$, kas ir definēts $n+1$ punktā, divos mazākos polinomos. Ar interpolāciju atradīsim $m$ iespējamos sadalījumu pārus $r_0$ un $r_1$. Tad izveidosim vienādojumu sistēmu, kas sastāv no sekojošiem $2n+1$ vienādojumiem:

- $c_1 + ... + c_m = n$. Kopā eksistē $n$ iespējas izvēlēties mainīgo, kura vērtība tiks pielīdzināta konstantei $c \in \{0, 1\}$. Vērtība $c_i$ ir vienāda ar reižu skaitu, kad, fiksējot kādu no $n$ mainīgajiem, tiek iegūts $i$-tais sadalījums polinomos $r_0$ un $r_1$.
- $\forall k \in \{0, .., n-1\} : c_1 r_1^1(k) + ... + c_m r_1^m(k) = \binom{k+1}{1} r(k+1)$. Šeit ar $r_1^i$ ir apzīmēts simetrizācijas polinoms $r_1$ no $i$-tā atrastā sadalījumu pāra. Šie $n$ vienādojumi pārbauda 23. teorēmas nosacījumu katrai no iespējamām $k$ vērtībām.
- $\forall k \in \{1, .., n\} : c_1 r_0^1(k) + ... + c_m r_0^m(k) = \binom{n-k}{1} r(k)$. Šie $n$ vienādojumi pārbauda 24. teorēmas nosacījumus.

Mūs interesē, vai šai vienādojumu sistēmai eksistē tāds risinājums, ka $\forall c_i \in \mathbb{N}_0$. Ja šai vienādojumu sistēmai neeksistē neviena risinājuma vai eksistē tikai viens risinājums, kas nav izsakāms naturālos skaitļos, tad simetrizācijas polinomu $r$ nav iespējams sadalīt sīkākos polinomos. Pārējos gadījumos uzskatīsim, ka risinājums naturālos skaitļos eksistē.

*6.1. tabula*

**Visi iespējamie simetrizācijas polinoma $r$ sadalījumi polinomos $r_0$ un $r_1$**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| $r$ | 0 | 8 | 27 | 30 | 10 | 0 | 3 | 2 | 0 |
| $r_0^1$ | 0 | 7 | 21 | 21 | 7 | 0 | 0 | 0 | – |
| $r_1^1$ | 1 | 6 | 9 | 3 | 0 | 3 | 2 | 0 | – |
| $r_0^2$ | 0 | 7 | 21 | 20 | 5 | 0 | 2 | 1 | – |
| $r_1^2$ | 1 | 6 | 10 | 5 | 0 | 1 | 1 | 0 | – |

**Piemērs.** Pārbaudīsim 6.1. tabulas simetrizācijas polinoma $r$ sadalījumus, risinot vienādojumu sistēmu, kuru var aplūkot 6.2. tabulā. No vienādojumu sistēmas pēdējām rindiņām iegūstam pretrunu, $c_2 = 3$ un $c_2 = 2$ vienlaicīgi. Tātad simetrizācijas polinomu

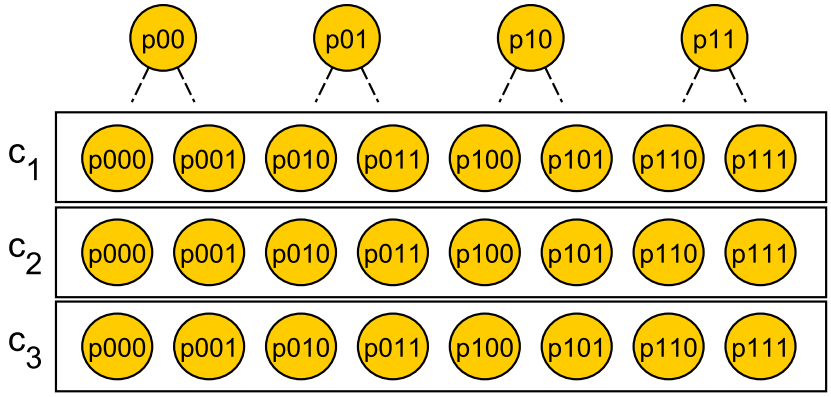
$r$ nav iespējams sadalīt sīkākos polinomos un līdz ar to šādam simetrizācijas polinomam neatbilst neviena Būla funkcija. □



**6.1. tabulas sadalījumam atbilstošā vienādojumu sistēma**

| Pārbaudāmā likumsakarība | Atbilstošais vienādojums |
|---|---|
| $c_1 + c_2 = n$ | $c_1 + c_2 = 8$ |
| $c_1 r_1^1(0) + c_2 r_1^2(0) = \binom{1}{1} r(1)$ | $c_1 + c_2 = 8$ |
| $c_1 r_1^1(1) + c_2 r_1^2(1) = \binom{2}{1} r(2)$ | $6c_1 + 6c_2 = 54$ |
| $c_1 r_1^1(2) + c_2 r_1^2(2) = \binom{3}{1} r(3)$ | $9c_1 + 10c_2 = 90$ |
| $c_1 r_1^1(3) + c_2 r_1^2(3) = \binom{4}{1} r(4)$ | $3c_1 + 5c_2 = 40$ |
| $c_1 r_1^1(4) + c_2 r_1^2(4) = \binom{5}{1} r(5)$ | $0 + 0 = 0$ |
| $c_1 r_1^1(5) + c_2 r_1^2(5) = \binom{6}{1} r(6)$ | $3c_1 + c_2 = 18$ |
| $c_1 r_1^1(6) + c_2 r_1^2(6) = \binom{7}{1} r(7)$ | $2c_1 + 1c_2 = 14$ |
| $c_1 r_1^1(7) + c_2 r_1^2(7) = \binom{8}{1} r(8)$ | $0 + 0 = 0$ |
| $c_1 r_0^1(0) + c_2 r_0^2(0) = \binom{8}{1} r(0)$ | $0 + 0 = 0$ |
| $c_1 r_0^1(1) + c_2 r_0^2(1) = \binom{7}{1} r(1)$ | $7c_1 + 7c_2 = 56$ |
| $c_1 r_0^1(2) + c_2 r_0^2(2) = \binom{6}{1} r(2)$ | $21c_1 + 21c_2 = 162$ |
| $c_1 r_0^1(3) + c_2 r_0^2(3) = \binom{5}{1} r(3)$ | $21c_1 + 20c_2 = 150$ |
| $c_1 r_0^1(4) + c_2 r_0^2(4) = \binom{4}{1} r(4)$ | $7c_1 + 5c_2 = 40$ |
| $c_1 r_0^1(5) + c_2 r_0^2(5) = \binom{3}{1} r(5)$ | $0 + 0 = 0$ |
| $c_1 r_0^1(6) + c_2 r_0^2(6) = \binom{2}{1} r(6)$ | $0 + 2c_2 = 6$ |
| $c_1 r_0^1(7) + c_2 r_0^2(7) = \binom{1}{1} r(7)$ | $0 + 1c_2 = 2$ |

Vienādojumu sistēmu varam risināt arī tad, kad simetrizācijas polinoma sadalīšanas procesā no $p$-tā sadalījuma līmeņa ieguvām vairākus iespējamos $(p + 1)$-mā sadalījuma līmeņus.




*6.1. att.* **Shematisks piemērs, kas parāda $m = 3$ iespējamo sadalījumu iegūšanu no 2. sadalījuma līmeņā**

Pieņemsim, ka tika atrasti $m$ veidi no $p$-tā sadalījuma līmeņa iegūt $(p+1)$-mo sadalījuma līmeni (sk. 6.1. att.). Tad izveidosim vienādojumu sistēmu, kas sastāv no sekojošiem vienādojumiem:

- $c_1 + ... + c_m = n - p$. Ja sadalām simetrizācijas polinomu, kas ir definēts $n+1$ punktos, tad $p$-jā sadalījuma līmenī būsim ieguvuši simetrizācijas polinomus, kas ir definēti $n-p+1$ punktos jeb atbilstīs $n-p$ mainīgo Būla funkcijai (daži no kuriem var būt fiktīvi). Savukārt vērtība $c_i$ ir vienāda ar reižu skaitu, kad, fiksējot vienu no $n-p$ mainīgajiem, tiek iegūts $i$-tais no $m$ iespējamiem $(k+1)$-mā sadalījuma līmeņiem.

- Katram no $2^p$ polinomiem $r_x$, kas ietilpst $p$-tā sadalījuma līmenī risināsim $\forall k \in \{0, ..., n-p-1\} : c_1 r^1_{x1}(k) + ... + c_m r^m_{x1}(k) = \binom{k+1}{1} r_x(k+1)$. Šie $2^p(n-k)$ vienādojumi pārbauda 23. teorēmas nosacījumus.

- Katram no $2^p$ polinomiem $r_x$, kas ietilpst $p$-tā sadalījuma līmenī risināsim $\forall k \in \{1, ..., n-p\} : c_1 r^1_{x0}(k) + ... + c_m r^m_{x0}(k) = \binom{(n-p)-k}{1} r_x(k)$. Šie $2^p(n-k)$ vienādojumi pārbauda 24. teorēmas nosacījumus.

Simetrizācijas polinomu secīgas sadalīšanas metodi un tās uzlabojumus iepriekš ir aprakstījis I.Stepanovs [9].

## 6.3. Polinomu sadalīšanas tālākās optimizācijas

Pieņemsim, ka vēlamies atrast Būla funkciju $f$, kurai attālums starp pakāpi $\deg(f)$ un jutīgumu $s(f)$ ir lielāks par šobrīd labāko rezultātu, jeb $\log_{s(f)} \deg(f) < \log_6 3$. (Analoģiska ideja būtu spēkā arī priekš vaicājumu sarežģītības $D(f)$.)

Izvēlēsimies vērtības $d$, $n$, $s$, kurām izpildās $\log_s d < \log_6 3$, un meklēsim $n$ mainīgo Būla funkciju ar pakāpi $d$ un jutīgumu $s$. Atradīsim visus atbilstošo simetrizācijas polinomus $r$ un mēģināsim sadalīt tos sīkākos.

Katram no sadalījuma simetrizācijas polinomiem ar iepriekš apskatītām likumsakarībām varam novērtēt atbilstošās Būla funkcijas pakāpes $\deg(r') \geq d'$ un jutīguma $s(f') \geq s'$ apakšējos novērtējumus. Pieņēmsim, ka kādam no sadalījuma simetrizācijas polinomiem izpildās $\log_{s'} d' < \log_6 3$. Ja polinoms ar pakāpi $d'$ un jutīgumu $s'$ eksistē, tad mums pietiktu atrast šo polinomu, lai uzlabotu pašlaik labāko zināmo rezultātu. Šī polinoma mainīgo skaits ir noteikti mazāks par $n$, tāpēc pārbaudīt šī polinoma eksistenci varētu būt vieglāk nekā pārbaudīt mūsu $d$. pakāpes $n$ mainīgo polinomu ar jutīgumu $s$. Tātad šī polinoma eksistenci pārbaudīt vajadzētu jau pirms tam, kad sākam $n$ mainīgo polinoma meklešanu. Savukārt, meklējot $n$ mainīgo polinomu, varam pieņemt, ka neeksistē polinomu ar mazāku mainīgo skaitu, kas dod attālumu labāku par $\log_6 3$.

Līdz ar to katru reizi, kad pamanām simetrizācijas polinomu ar pakāpi $\deg(f') \geq d'$ un jutīgumu $s(f') \geq s'$, kuram izpildās $\log_{s'} d' < \log_6 3$, mēs varētu pieņemt, ka attiecīgās Būla funkcijas neeksistē. Un palielināt pakāpes apakšējo novērtējumu $\deg(f')$ līdz tādai vērtībai, ka šis polinoms vairs neuzlabotu rezultātu $\log_6 3$, tātad $\deg(f') \geq {s'}^{\log_6 3}$. Savukārt šīs izmaiņas rezultātā citi izmantotie kritēriji varētu nobrāķēt šādu simetrizācijas polinomu, un kopumā mums būs jāapskata mazāku pozīciju skaitu.

Minētā optimizācija šajā darbā izstrādātajā datora programmā netika izmantota.

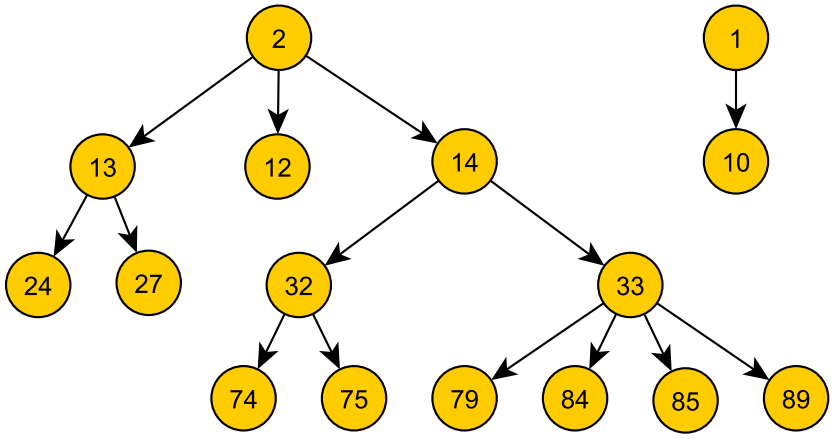
## 7.1. Pētījuma mērķiem izstrādātā datora programma

Darba ietvaros tika izstrādāta C++ programma, kas implementē visas darbā aprakstītās metodes. Programmas izstrādei tika izmantota Microsoft Visual Studio vide un TortoiseSVN versiju pārvaldības sistēma. Izveidotais Microsoft Visual Studio projekts sastāv no 17 failiem, kas kopā satur $\sim 2000$ pirmkoda rindiņas, tai skaitā vienību testus.

Izstrādātais C++ projekts satur 5 klases, kuru deklarācijas var apskatīt 1. pielikumā. Savukārt 2. pielikumā ir aplūkojami daži no šī darba interesantākiem vai sarežģītākiem pirmkoda fragmentiem.

## 7.2. Ar datora programmu iegūtie rezultāti

Pārlasot 2. pakāpes 3 mainīgo simetrizācijas polinomus, izstrādātā programma veiksmīgi atrod Nisan un Szegedy funkcijas triviālu sadalījumu (sk. 5.1. att.). Savukārt, pārlasot 3. pakāpes 6 mainīgo simetrizācijas polinomus, tiek atrasts Kushilevitz funkcijas triviāls sadalījums. Šie rezultāti var kalpot kā apliecinājums, ka izstrādātā programma strādā korekti.




*7.1. att.* **Atrastie sadalījumi, meklējot** 5. **pakāpes** 15 **mainīgo Būla funkciju**

Meklējot 5. pakāpes 15 mainīgo Būla funkciju ar jutīgumu 15 (un tātad arī determinēto vaicājumu sarežģītību 15), iegūtās korektās pozīcijas var aplūkot 7.1. attēlā. Virsotnes 1 un 2 atbilst sākuma simetrizācijas polinomiem, savukārt visas pārējās virsot-

nes atbilst sadalījuma līmeņiem, kuru iegūšanas veids ir izsekojams ar bultiņām no sākuma simetrizācijas polinomiem. Attēla virsotņu numerācija norāda uz to, ka simetrizācijas polinomu sadalījumi ar pārējiem kārtas numuriem tika nobrāķēti, jo tie neapmierina kādu no darbā aprakstītiem nosacījumiem. Kopumā apstrādes laikā tika apstrādātas 165. dažādu sadalījuma līmeņu instances. Apstrādes laikā tika atrasti arī 4. līmeņa sadalījumi, bet neviens no tiem nav bijis korekts. Tātad neeksistē 5. pakāpes 15 mainīgo Būla funkcija ar jutīgumu 15. Katrai attēla virsotnei atbilstošu simetrizācijas polinoma sadalījuma līmeni var apskatīt 3. pielikumā.

Meklējot 2. pakāpes 3 mainīgo polinomus, 3. pakāpes 6 mainīgo polinomus un 5. pakāpes 15 mainīgo polinomus programma strādā gandrīz momentāni, jo katrā gadījumā apstrādājamo pozīciju skaits ir ļoti mazs. Tas skar 5. pakāpes 15 mainīgo gadījumu tāpēc, ka priekš šiem parametriem eksistē tikai 8 sākuma polinomi (sk. 4.2. tab.), no kuriem 6 nevar dot pat korektu 1. līmeņa sadalījumu.

Salīdzinājumam, 5. pakāpes 14 mainīgo gadījumam eksistē 247 sākotnējie simetrizācijas polinomi. Šo polinomu apstrāde aizņēm vairāk par 3 minūtēm, kā rezultātā tiek atrasts vairāk par 14000 derīgu pirmā līmeņa sadalījumu, bet atrasto pozīciju uzglabāšanai izmantots vairāk par 500MB operatīvās atmiņas. Četru stundu laikā programma tiek līdz 4. līmeņa sadalījumu iegūšanas no 3. līmeņa sadalījumiem. Autoram nav izdevies sagaidīt, līdz programma pabeidz savu darbību. Kopumā ir problēmas gan ar ātrdarbību, gan ar atmiņas resursiem.

# SECINĀJUMI

Darba rezultātā ir izdevies parādīt, ka neeksistē 5. pakāpes 15 mainīgo Būla funkcijas ar jutīgumu 15. Tomēr šis rezultāts ir iegūts ar programmas palīdzību, kas nesniedz pietiekamu pārliecību par rezultāta patiesumu. Būtu nepieciešams nākotnē atrast veidu šo rezultātu pierādīt ar formālām metodēm.

Iegūtais rezultāts nenozīmē, ka neeksistē 5. pakāpes 15 mainīgo Būla funkcijas ar determinēto vaicājumu sarežģītību 15, jo šādām funkcijām var atbilst arī mazāka jutīguma vērtība. Tomēr arī iegūtais rezultāts mums ir interesants.

Darbā ir piedāvāta diezgan spēcīga metode mūs interesējošo Būla funkciju meklešanai, tomēr vispārīgā gadījumā tā vēl aizvien nav pietiekami efektīva, lai iegūtu rezultātus patvaļīgai pakāpei un mainīgo skaitam. Līdz ar to nav izdevies atrast tādu Būla funkciju, kura uzlabotu pašlaik labāko zināmo attālumu starp pakāpi un determinēto vaicājumu sarežģītību vai jutīgumu.

Novērojām, ka izstrādātā datora programma apstrādāja 5. pakāpes 15 mainīgo gadījumu gandrīz momentāni, tomēr nespēja tikt galā ar 5. pakāpes 14 mainīgo gadījumu. Starpība ir tajā, ka eksistē ļoti maz 5. pakāpes 15 mainīgo simetrizācijas polinomu ar jutīgumu 15. Pēc analoģijas vajadzētu mēģināt atrast un pārbaudīt citus pakāpes un mainīgā skaita pārus, kam ir ļoti maz sākuma pozīciju.

Darba pirmajā pusē tika analizētas daudzas likumsakarības starp Būla funkcijas pakāpi, jutīgumu un šai funkcijai atbilstošo simetrizācijas polinomu. Diezgan pārsteidzoši izrādījās, ka gandrīz visas apskatītās pakāpes likumsakarības izrādījās sliktākas par visvienkāršāko kritēriju $\deg(p) \geq \deg(p^{sym})$. Šobrīd neliekas lietderīgi turpināt meklēt papildus likumsakarības starp šiem Būla funkcijas raksturlielumiem.

Lai gan 6.3. nodaļā piedāvātā optimizācija neskan pārliecinoši, tomēr būtu nepieciešams pārbaudīt, vai tā neuzlabos šobrīd izmantotās metodes efektivitāti.

Lai gan sākotnējos darba mērķus nav izdevies sasniegt, tomēr ir aprakstītas jaunas idejas šī mērķa sasniegšanai un ir parādīts, ka neeksistē 5. pakāpes 15 mainīgo Būla funkcijas ar jutīgumu 15.

# IZMANTOTĀ LITERATŪRA


[1] James Ax. Zeroes of polynomials over finite fields. *American Journal of Mathematics*, 86(2):255–261, April 1964.

[2] Beals, Buhrman, Cleve, Mosca, and de Wolf. Quantum lower bounds by polynomials. *JACM: Journal of the ACM*, 48, 2001.

[3] Buhrman and de Wolf. Complexity measures and decision tree complexity: A survey. *TCS: Theoretical Computer Science*, 288, 2002.

[4] Eyal Kushilevitz. Communication complexity. `http://www.cs.bris.ac.uk/probtcs08/materials/kushilevitz.pdf`, July 2008. Tiešsaite: 20/05/2012.

[5] Aleksejs Lučko. Būla funkcijas ar zemu polinoma pakāpi pielietojumi kvantu skaitļošanā. Bakalaura darbs, Latvijas Universitāte, Fizikas un matemātikas fakultāte, Rīga, 2002.

[6] Nisan and Wigderson. On rank vs. communication complexity. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 1994.

[7] Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994.

[8] David Rubinstein. Sensitivity vs. block sensitivity of boolean functions. *Combinatorica*, 15(2):297–299, 1995.

[9] Igors Stepanovs. Zemu polinoma pakāpju Būla funkciju vaicājumu sarežģītība. Maģistra kursa darbs, Latvijas Universitāte, Datorikas fakultāte, Rīga, 2012.

# PIELIKUMI

## 1. pielikums **Izstrādātās programmas klašu deklarācijas**

Sāksim ar *Polynomial* klasi, kuru izmanto visas klases, kas ir saistītas ar polinomu apstrādi – ģenerēšanu, sadalīšanu, utt.

Šī klase satur visu zināmo informāciju par konkrēto simetrizācijas polinomu. Mainīgais *degreeCap* satur lielāko iespējamo pakāpi, kura varētu atbilst simetrizācijas polinomam, savukārt mainīgais *likelyDegree* satur apakšējo pakāpes novērtējumu, kas tiek iegūts ar kādu no darbā apskatītām likumsakarībām.

Mainīgais *variables* satur šī simetrizācijas polinoma mainīgo skaitu, bet masīva šūna *term[i]* satur simetrizācijas polinoma vērtību $r[i]$. Metode *interpolateAndComputePolynomial* kā argumentus pieņem šī simetrizācijas polinoma vērtības *degreeCap + 1* punktos, interpolē pārējās vērtības un saglabā tās masīvā *term*.

Savukārt mainīgais *valid* satur informāciju, vai konkrētais simetrizācijas polinoms ir korekts. Piemēram, ja kāda no interpolācijas rezultātā iegūtajām $r[k]$ vērtībām nav vesels skaitlis starp 0 un $\binom{n}{k}$, tad konkrētais simetrizācijas polinoms nav korekts.

```
1  class Polynomial
2  {
3  public:
4      // Arithmetic operations
5      friend Polynomial operator-(const Polynomial&a, const Polynomial& b);
6      friend Polynomial operator+(const Polynomial&a, const Polynomial& b);
7
8      // Interface to outside
9      bool isValid() const          { return valid; }
10
11     int getLikelyDegree() const      { return likelyDegree; }
12     int getDegreeCap() const         { return degreeCap; }
13     int getVariables() const         { return variables; }
14
15     int getTerm(int id) const { return term[id]; }
16
17     bool isConstant() const          { return likelyDegree == 0 && variables
           == 0; }
18     bool isZeroPolynomial() const    { return likelyDegree == 0 &&
           lagrangeNumerator[0] == 0; }
```

```
19
20      void printOnScreen() const;
21
22      friend class PolynomialFramework;
23      friend class PolynomialDecomposition;
24  protected:
25      Polynomial(int variables, int degreeCap);
26
27      // Internal state
28      bool valid; // can be interpolated as a polynomial of specific degree
29
30      int degreeCap;
31      int likelyDegree;
32      int minRange, maxRange;
33
34      int variables;
35      int term[VARIABLES + 1];
36
37      void interpolateAndComputePolynomial(int maxDegree, int* termIndex, int
            * termValue);
38
39      long long lagrangeNumerator[VARIABLES + 1];
40      long long lagrangeDenominator;
41      void interpolateLagrangePolynomial(int maxDegree, int* x, int* y);
42      void computePolynomialValues(int maxDegree, int* termIndex, int*
            termValue);
43  };
```

Klase *PolynomialFramework* mantojas no klases *Polynomial* un ir paredzēta secīgai pakāpes *degreeCap* un mainīgo skaita *variables* simetrizācijas polinomu ģenerācijai. Šīs klases metode *findNextValidPolynomial()* atrod kārtējo (leksikogrāfiskā secībā) simetrizācijas polinomu, kas atbilst nepieciešamajiem kritērijiem. Šim nolūkam tiek pārlasītas deg(*degreeCap*) pakāpes simetrizācijas polinoma vērtības deg(*degreeCap* + 1) punktos, tad tiek interpolēti pārējie punkti. Masīva šūnā *termIndex[i]* glabājas $i$-tā punkta indekss, kurā ir fiksēta viena no polinoma vērtībām $minTermValue[i] \leq termValue[i] \leq maxTermValue[i]$.

```
1  enum PolynomialType { BASE, LEFT, RIGHT };
```

```
2
3  class PolynomialFramework: public Polynomial
4  {
5  public:
6      PolynomialFramework(int degreeCap, int variables, int
           minSensitivityBound);
7      PolynomialFramework(const Polynomial& parent, PolynomialType type);
8
9      bool findNextValidPolynomial();
10 private:
11     PolynomialType type;
12
13     bool initialized;
14     void initializeFramework(int minSensitivityBound);
15     bool increasePolynomialTerms();
16
17     int termIndex[DEGREE + 1];
18     int termValue[DEGREE + 1];
19     int minTermValue[DEGREE + 1];
20     int maxTermValue[DEGREE + 1];
21 };
```

*PolynomialDecomposition* klase izmanto iepriekšējās klases, lai ģenerētu visus simetrizācijas polinoma sadalījumus divos mazākos polinomos. Šī klase inicializācijas laikā atrod visus šādus sadalījumus, tai skaitā tā sastāda vienādojumu sistēmu un pārbauda, ka iegūtie sadalījumi apmierina darbā aprakstītos nosacījumus. Turpmāk šī klase tiek izmantota, lai uzglabātu un piekļūtu šiem sadalījumiem.

```
1  struct PolynomialDecompositionStruct
2  {
3      PolynomialDecompositionStruct(const Polynomial& poly)
4      :    base(poly)
5      {
6          decompositionCount = 0;
7      }
8
9      int decompositionCount;
10     Polynomial base;
11
```

```
12      vector<PolynomialFramework> left;
13      vector<Polynomial> right;
14  };
15
16  class PolynomialDecomposition
17  {
18  public:
19      PolynomialDecomposition(const Polynomial& base_polynomial);
20      PolynomialDecomposition(const PolynomialDecomposition& parent,
            PolynomialType type, int state);
21      ~PolynomialDecomposition() {}
22
23      const Polynomial& getBase() const { return pd->base; }
24      const Polynomial& getLeft(int state) const { return pd->left[state]; }
25      const Polynomial& getRight(int state) const { return pd->right[state];
            }
26
27      int getDecompositionCount() const { return pd->decompositionCount; }
28  protected:
29      void initializeDecomposition(const Polynomial& base_polynomial);
30      string PolynomialDecomposition::computePolynomialID(const Polynomial&
            poly) const;
31
32      PolynomialDecompositionStruct* pd;
33  };
```

Klase *PolynomialHierarchy* tiek izmantota, lai glabātu un apstrādātu simetrizācijas polinomu sadalījuma līmeņus. Klase glabā divus simetrizācijas polinoma blakus sadalījuma līmeņus, kas ļauj no augšējā līmeņa secīgi ģenerēt apakšējo līmeni. Funkcija *deriveNextLevel* atrod nākamo korekto sadalījuma līmeni.

```
1   class PolynomialHierarchy
2   {
3   public:
4       PolynomialHierarchy(const Polynomial& base_polynomial);
5       PolynomialHierarchy(const PolynomialHierarchy& base, const
            PolynomialHierarchy& base2);
6
7       void showDescription() const;
```

```
8       int hierarchyLength() const { return sequence.size(); };
9       const PolynomialDecomposition& getSequenceMember(int id) const { return
            sequence[id]; };
10      bool deriveNextLevel(int initialDegree, int previousInQueue, vector<
            pair<PolynomialHierarchy, int> >& queue);
11  protected:
12      void initOrder();
13      bool satisfiesDegreeRestrictions(int initialDegree, int pos) const;
14
15      bool hasFaultyDecomposition;
16      vector<int> processingOrder;
17      vector<int> currentState;
18
19      vector<const PolynomialDecomposition> sequence;
20  };
```

Klase *GaussianElimination* tiek izmantota vienādojumu sistēmu risināšanai, lai pārbaudītu, ka simetrizācijas polinomu sadalījums ir korekts.

```
1   class GaussianElimination
2   {
3   public:
4       GaussianElimination(int variables, int equations);
5       ~GaussianElimination();
6
7       void addEquation(int* equation, int size);
8       void addEquation(vector<int> equation);
9
10      void solve();
11      bool hasNoSolutions();
12      void printOnScreen();
13  private:
14      int variables;
15      int equations;
16      int triangleSize;
17      int initializedEquations;
18      int** system;
19
20      bool noSolutions;
21
```

```
22      void applyGcdToAllEquationsBelow(int belowEquation);
23      void normalizeSignsInColumns(int belowEquation, int column);
24      int findEquationWithSmallestValueInColumn(int belowEquation, int column
            );
25      void swapEquations(int a, int b);
26      void swapColumns(int a, int b);
27      void substractEquationsFromSystem(int equationId);
28      bool isSatisfiedByIntegerValues();
29  };
```

2. pielikums **Daži izstrādātās programmas pirmkoda fragmenti**

Sekojošās divas *Polynomial* klases metodes interpolē simetrizācijas polinomu no *degreeCap + 1* punktiem, aizpilda masīvu *term* ar iegūtām polinoma vērtībām visos punktos un pārbauda, ka šis polinoms ir korekts.

```
1  // maxDegree == max allowed degree of the particular symmetrization
       polynomial
2  void Polynomial::interpolateLagrangePolynomial(int maxDegree, int* x, int*
       y)
3  {
4      memset(lagrangeNumerator, 0, sizeof(long long) * (variables + 1));
5      lagrangeDenominator = 1;
6
7      long long* tmp = new long long[maxDegree + 1];
8      for (int j = 0; j < maxDegree + 1; ++j)
9      {
10         long long currentDenominator = 1;
11         memset(tmp, 0, sizeof(long long) * (maxDegree + 1));
12         tmp[0] = 1;
13         int currentDegree = 0;
14         for (int m = 0; m < maxDegree + 1; ++m)
15         {
16             if (m == j) continue;
17             currentDenominator *= (x[j] - x[m]);
18             for (int i = ++currentDegree; i > 0; --i)
19             {
20                 tmp[i] = tmp[i] * (-x[m]) + tmp[i - 1];
21             }
22             tmp[0] *= (-x[m]);
23         }
24         currentDenominator *= cnk[variables][x[j]];
25         long long tmpDenominator = lagrangeDenominator * currentDenominator
               / gcd(lagrangeDenominator, currentDenominator);
26         for (int i = 0; i < maxDegree + 1; ++i)
27         {
28             lagrangeNumerator[i] = lagrangeNumerator[i] * (tmpDenominator /
                   lagrangeDenominator) + tmp[i] * y[j] * (tmpDenominator /
                  currentDenominator);
29         }
```

```
30          lagrangeDenominator = tmpDenominator;
31      }
32
33      while (maxDegree > 0 && lagrangeNumerator[maxDegree] == 0)
34      {
35          --maxDegree;
36      }
37      this->likelyDegree = maxDegree;
38      delete[] tmp;
39  }
40
41  // maxDegree == the amount of data the acquired polynomial is based on
42  void Polynomial::computePolynomialValues(int maxDegree, int* termIndex, int
        * termValue)
43  {
44      for (int i = 0; i < variables + 1; ++i)
45      {
46          long long tmp = 1, res = 0;
47          for (int j = 0; j < likelyDegree + 1; ++j)
48          {
49              res += tmp * lagrangeNumerator[j];
50
51              if (j != likelyDegree)
52              {
53                  tmp = tmp * i;
54              }
55          }
56          res *= cnk[variables][i];
57          valid = valid && (res % lagrangeDenominator == 0);
58          res /= lagrangeDenominator;
59          valid = valid && res >= minRange * cnk[variables][i] && res <=
                maxRange * cnk[variables][i];
60          term[i] = res;
61          if (!valid) return;
62      }
63  }
```

Nākamā funkcija pārbauda pakāpju nosacījumu izpildīšanos viena simetrizācijas polinoma sadalījuma līmeņa ietvaros.

```
1  bool PolynomialHierarchy::satisfiesDegreeRestrictions(int degreeCap, int
       pos) const
2  {
3      int pos_map = processingOrder[pos];
4      int posBits = getAmountOfBitsInInteger(pos_map);
5      Polynomial res = (pos_map % 2 == 0 ?
6          sequence[pos_map / 2].getLeft(currentState[pos_map / 2]) :
7          sequence[pos_map / 2].getRight(currentState[pos_map / 2]));
8
9      for (int tmp_pos = 0; getAmountOfBitsInInteger(processingOrder[tmp_pos
           ]) < posBits; ++tmp_pos)
10     {
11         if (!firstSetIsSubsetOfSecondSet(processingOrder[tmp_pos],
               processingOrder[pos]))
12         {
13             continue;
14         }
15         pos_map = processingOrder[tmp_pos];
16         if (getAmountOfBitsInInteger(pos_map) % 2 == posBits % 2)
17         {
18             res = res + (pos_map % 2 == 0 ?
19                 sequence[pos_map / 2].getLeft(currentState[pos_map / 2]) :
20                 sequence[pos_map / 2].getRight(currentState[pos_map / 2]));
21         }
22         else
23         {
24             res = res - (pos_map % 2 == 0 ?
25                 sequence[pos_map / 2].getLeft(currentState[pos_map / 2]) :
26                 sequence[pos_map / 2].getRight(currentState[pos_map / 2]));
27         }
28         if (!res.isValid()) return false;
29     }
30
31     if (degreeCap >= posBits)
32         return res.getLikelyDegree() <= degreeCap - posBits;
33     else
34         return res.isZeroPolynomial();
35 }
```

Funkcija substractPolynomials izmanto pieejamās klases, lai ģenerētu visus sākotnējos simetrizācijas polinomus, katru no tiem mēģinot sadalīt sīkākos polinomos un pārbaudot, ka starp blakus sadalījumu līmeņiem izpildās nosacījumi, kas ir aprakstāmi ar vienādojumu sistēmu.

```
1  void substractPolynomials(int degreeCap, int variables, int
       minSensitivityBound)
2  {
3      vector<pair<PolynomialHierarchy, int> > queue;
4      vector<int> hierarchyVariables;
5      vector<bool> correctHierarchy;
6      int hierarchyProgress;
7      int currentVariables = variables;
8
9    for (int degree = degreeCap; degree > -1; --degree)
10   {
11     PolynomialFramework p(degree, variables, minSensitivityBound);
12     while (p.findNextValidPolynomial())
13     {
14             //p.printOnScreen();
15       PolynomialHierarchy ph(p);
16       queue.push_back(make_pair(ph, -1));
17       correctHierarchy.push_back(true);
18       hierarchyVariables.push_back(currentVariables);
19     }
20   }
21     hierarchyProgress = queue.size();
22     printf("Done generating\n");
23
24     for (int i = 0; i < queue.size(); ++i)
25     {
26         printf("\r%d/%d\r", i, queue.size());
27         if (i == hierarchyProgress)
28         {
29             --currentVariables;
30             for (int id = i; id < queue.size(); ++id)
31             {
32                 correctHierarchy.push_back(true);
33                 hierarchyVariables.push_back(currentVariables);
```

```
34              }
35              for (int rootid = i - 1; rootid > - 1; --rootid)
36              {
37                  if (!correctHierarchy[rootid]) continue;
38                  vector<int> descendants;
39                  for (int candidate = rootid + 1; candidate < queue.size();
                        ++candidate)
40                      if (queue[candidate].second == rootid &&
                            correctHierarchy[candidate])
41                      {
42                          descendants.push_back(candidate);
43                      }
44                  // gauss
45                  const PolynomialHierarchy& root = queue[rootid].first;
46                  const int var = hierarchyVariables[rootid];
47                  GaussianElimination gauss(descendants.size(), 2 * var *
                        root.hierarchyLength() + 1);
48                  // sum
49                  vector<int> sum;
50                  for (int i = 0; i < descendants.size(); ++i)
51                      sum.push_back(1);
52                  sum.push_back(var);
53                  gauss.addEquation(sum);
54                  // for each of the root polynomials
55                  for (int rootEquationId = 0; rootEquationId < root.
                        hierarchyLength(); ++rootEquationId)
56                  {
57                      // for 0
58                      for (int k = 0; k < var; ++k)
59                      {
60                          vector<int> equation;
61                          for (int descendant = 0; descendant < descendants.
                                size(); ++descendant)
62                              equation.push_back(queue[descendants[descendant
                                    ]].first.getSequenceMember(rootEquationId *
                                    2).getBase().getTerm(k));
63                          equation.push_back((var - k) * root.
                                getSequenceMember(rootEquationId).getBase().
                                getTerm(k));
```

```
64                          gauss.addEquation(equation);
65
66                          equation.clear();
67                          for (int descendant = 0; descendant < descendants.
                                size(); ++descendant)
68                              equation.push_back(queue[descendants[descendant
                                    ]].first.getSequenceMember(rootEquationId *
                                    2 + 1).getBase().getTerm(k));
69                          equation.push_back((k + 1) * root.getSequenceMember
                                (rootEquationId).getBase().getTerm(k + 1));
70                          gauss.addEquation(equation);
71                      }
72                  }
73                  gauss.solve();
74                  if (gauss.hasNoSolutions())
75                  {
76                      correctHierarchy[rootid] = false;
77                      for (int descendant = 0; descendant < descendants.size
                            (); ++descendant)
78                          correctHierarchy[descendants[descendant]] = false;
79                  }
80              }
81              hierarchyProgress = queue.size();
82          }
83
84          if (correctHierarchy[i])
85          {
86              bool hierarchyIsCorrect = true;
87              for (int tmp = i; tmp != -1;)
88              {
89                  if (!correctHierarchy[tmp])
90                  {
91                      hierarchyIsCorrect = false;
92                      break;
93                  }
94                  tmp = queue[tmp].second;
95              }
96              correctHierarchy[i] = hierarchyIsCorrect;
97          }
```

```
98
99          if (!correctHierarchy[i]) continue;
100         PolynomialHierarchy ph = queue[i].first;
101
102         int len = queue.size();
103         while (ph.deriveNextLevel(degreeCap, i, queue)) {}
104         if (queue.size() - len > 0 || hierarchyVariables[i] == 0)
105         {
106             int tmp = i;
107             for (; tmp != -1;)
108             {
109                 if (tmp != i) printf(" -> ");
110                 printf("%d", tmp);
111                 tmp = queue[tmp].second;
112             }
113             printf(" (%d)\n", queue.size() - len);
114             //ph.showDescription();
115         } else
116         {
117             correctHierarchy[i] = false;
118         }
119     }
120     printf("\nTotal positions processed: %d\n", queue.size());
121 }
```

3. pielikums **Simetrizācijas polinomu sadalījumi, kas tiek iegūti, meklējot 5. pakāpes 15 mainīgo Būla funkciju**

```
1: 
r: [0, 15, 104, 271, 280, 51, 400, 2163, 4272, 4605, 2952, 1085, 184, 1, 0, 1]
2: 
r: [0, 15, 105, 280, 315, 126, 490, 2205, 4230, 4515, 2877, 1050, 175, 0, 0, 1]
10: 
r0: [0, 14, 90, 216, 204, 36, 252, 1176, 2016, 1854, 986, 288, 36, 0, 0]
r1: [1, 14, 55, 76, 15, 148, 987, 2256, 2751, 1966, 797, 148, 1, 0, 1]
12: 
r0: [0, 14, 91, 222, 213, 12, 126, 924, 1722, 1638, 887, 262, 33, 0, 0]
r1: [1, 14, 58, 102, 114, 364, 1281, 2508, 2877, 1990, 788, 142, 0, 0, 1]
13: 
r0: [0, 14, 91, 223, 222, 48, 210, 1050, 1848, 1722, 923, 271, 34, 0, 0]
r1: [1, 14, 57, 93, 78, 280, 1155, 2382, 2793, 1954, 779, 141, 0, 0, 1]
14: 
r0: [0, 14, 91, 224, 231, 84, 294, 1176, 1974, 1806, 959, 280, 35, 0, 0]
r1: [1, 14, 56, 84, 42, 196, 1029, 2256, 2709, 1918, 770, 140, 0, 0, 1]
24: 
r00: [0, 13, 78, 175, 156, 18, 84, 462, 720, 561, 238, 51, 4, 0]
r01: [1, 13, 48, 66, 30, 126, 588, 1128, 1161, 685, 220, 30, 0, 0]
r10: [1, 13, 48, 66, 30, 126, 588, 1128, 1161, 685, 220, 30, 0, 0]
r11: [1, 9, 27, 48, 154, 567, 1254, 1632, 1269, 559, 111, 0, 0, 1]
27: 
r00: [0, 13, 78, 175, 157, 26, 112, 518, 790, 617, 266, 59, 5, 0]
r01: [1, 13, 48, 65, 22, 98, 532, 1058, 1105, 657, 212, 29, 0, 0]
r10: [1, 13, 49, 74, 58, 182, 658, 1184, 1189, 693, 221, 30, 0, 0]
r11: [1, 8, 19, 20, 98, 497, 1198, 1604, 1261, 558, 111, 0, 0, 1]
32: 
r00: [0, 13, 78, 175, 157, 26, 112, 518, 790, 617, 266, 59, 5, 0]
r01: [1, 13, 49, 74, 58, 182, 658, 1184, 1189, 693, 221, 30, 0, 0]
r10: [1, 13, 48, 65, 22, 98, 532, 1058, 1105, 657, 212, 29, 0, 0]
r11: [1, 8, 19, 20, 98, 497, 1198, 1604, 1261, 558, 111, 0, 0, 1]
33: 
r00: [0, 13, 78, 176, 165, 54, 168, 588, 846, 645, 274, 60, 5, 0]
r01: [1, 13, 48, 66, 30, 126, 588, 1128, 1161, 685, 220, 30, 0, 0]
```

```
r10: [1, 13, 48, 66, 30, 126, 588, 1128, 1161, 685, 220, 30, 0, 0]
r11: [1, 8, 18, 12, 70, 441, 1128, 1548, 1233, 550, 110, 0, 0, 1]
74:
r000: [0, 12, 66, 135, 109, 8, 28, 182, 250, 164, 58, 11, 1]
r001: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r010: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r011: [1, 9, 26, 40, 98, 322, 644, 736, 485, 173, 26, 0, 0]
r100: [1, 12, 40, 47, 10, 56, 280, 470, 397, 180, 40, 3, 0]
r101: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r110: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r111: [0, 1, 8, 56, 245, 610, 896, 784, 386, 85, 0, 0, 1]
75:
r000: [0, 12, 66, 135, 109, 8, 28, 182, 250, 164, 58, 11, 1]
r001: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r010: [1, 12, 41, 56, 46, 140, 406, 596, 481, 216, 49, 4, 0]
r011: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r100: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r101: [1, 8, 17, 4, 14, 196, 518, 652, 449, 164, 25, 0, 0]
r110: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r111: [0, 1, 8, 56, 245, 610, 896, 784, 386, 85, 0, 0, 1]
79:
r000: [0, 12, 66, 135, 112, 28, 84, 266, 320, 192, 58, 7, 0]
r001: [1, 12, 41, 53, 26, 84, 322, 526, 453, 216, 53, 5, 0]
r010: [1, 12, 41, 53, 26, 84, 322, 526, 453, 216, 53, 5, 0]
r011: [1, 7, 13, 4, 42, 266, 602, 708, 469, 167, 25, 0, 0]
r100: [1, 12, 41, 53, 26, 84, 322, 526, 453, 216, 53, 5, 0]
r101: [1, 7, 13, 4, 42, 266, 602, 708, 469, 167, 25, 0, 0]
r110: [1, 7, 13, 4, 42, 266, 602, 708, 469, 167, 25, 0, 0]
r111: [1, 5, 8, 28, 175, 526, 840, 764, 383, 85, 0, 0, 1]
84:
r000: [0, 12, 66, 135, 109, 8, 28, 182, 250, 164, 58, 11, 1]
r001: [1, 12, 41, 56, 46, 140, 406, 596, 481, 216, 49, 4, 0]
r010: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r011: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r100: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r101: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r110: [1, 8, 17, 4, 14, 196, 518, 652, 449, 164, 25, 0, 0]
r111: [0, 1, 8, 56, 245, 610, 896, 784, 386, 85, 0, 0, 1]
85:
```

```
r000: [0, 12, 66, 136, 116, 29, 63, 217, 271, 171, 59, 11, 1]
r001: [1, 12, 40, 49, 25, 105, 371, 575, 474, 215, 49, 4, 0]
r010: [1, 12, 40, 49, 25, 105, 371, 575, 474, 215, 49, 4, 0]
r011: [1, 8, 17, 5, 21, 217, 553, 687, 470, 171, 26, 0, 0]
r100: [1, 12, 40, 49, 25, 105, 371, 575, 474, 215, 49, 4, 0]
r101: [1, 8, 17, 5, 21, 217, 553, 687, 470, 171, 26, 0, 0]
r110: [1, 8, 17, 5, 21, 217, 553, 687, 470, 171, 26, 0, 0]
r111: [0, 1, 7, 49, 224, 575, 861, 763, 379, 84, 0, 0, 1]
89:
r000: [0, 12, 66, 136, 117, 36, 84, 252, 306, 192, 66, 12, 1]
r001: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r010: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r011: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r100: [1, 12, 40, 48, 18, 84, 336, 540, 453, 208, 48, 4, 0]
r101: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r110: [1, 8, 18, 12, 42, 252, 588, 708, 477, 172, 26, 0, 0]
r111: [0, 0, 0, 28, 189, 540, 840, 756, 378, 84, 0, 0, 1]
```

Maģistra darbs

"Zemas pakāpes polinomu Būla funkciju vaicājumu sarežģītība"

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai. Piekrītu sava darba publicēšanai internetā.

Autors: _____

(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augšminēto maģistra darbu un atzīstu to par **piemērotu/nepiemērotu** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu bakalaura studiju programmas gala pārbaudījuma komisijas sēdē.

Darba vadītājs: _____

(Vadītāja paraksts)

Darbs iesniegts Datorikas fakultātē _____

(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Metodiķe: _____

(Metodiķes paraksts)

Recenzents: _____

(Recenzenta paraksts)

Darbs aizstāvēts datorzinātņu maģistra pārbaudījumu komisijas sēdē

_____ prot. Nr. _____, vērtējums_____

(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(Sekretāra paraksts)