Latvijas Universitāte
Datorikas fakultāte

Atbalsta bibliotēka Būla funkciju izpētei

Kvalifikācijas darbs

Autors: Ilmārs Pužulis
Studenta apliecības Nr.: ip10059

Darba vadītājs: *Dr.sc.comp.* Juris Smotrovs

Rīga, 2015

Anotācija

Būla funkcijas ir joprojām aktuāls pētījumu objekts. Tās izmanto sarežģītības teorijā, elektrisko slēgumu projektēšanā, kvantu kriptogrāfijā un citur. Šī darba mērķis ir izveidot programmatūru vienkāršāko Būla funkciju raksturotāju noteikšanai: jūtīguma, bloku jūtīguma, pārstāvošo un tuvināto polinomu aprēķināšanai. Tās galvenais pielietojums varētu būt pielietošana projektā "Kvantu algoritmi" (angliski: QALGO: Quantum Algorithmics), kas pēta arī Būla funkcijas. Motivācija šādas programmas izstrādei nāk no pētnieku puses, kas ikdienā nodarbojas ar funkciju un algoritmu sarežģītības pētīšanu, kā procesā ir nepieciešams veikt atsevišķu raksturotāju noteikšanu pastarpinātiem algoritmiem vai funkcijām.

Atslēgvārdi: Būla funkcijas, funkciju sarežģītība, tuvinātie polinomi, jūtīgums, bloku jūtīgums.

Abstract

Support Library for Analysis of Boolean Functions

Boolean functions are still topical subject of the research. They are used in complexity theory, electrical circuit design, quantum cryptography and elsewhere. This work aims to create software for calculation of the simplest Boolean function descriptors: sensitivity, block sensitivity, representing and approximated polynomial. Its main use could be in the project "QALGO: Quantum Algorithmic", which explores the Boolean functions as well. Motivation for developing such a program comes from the research community.

Keywords: Boolean functions, function complexity, approximate polynomials, sensitivity, block sensitivity.

SATURS

IEVADS

Būla funkcijas ir viens no svarīgākajiem teorētiskās datorzinātnes pētījumu objektiem. Tām ir plašs pielietojums teorētiskajā datorzinātnē, sarežģītības teorijā, elektrisko slēgumu izveidē un arī kriptogrāfijā.

Teorētiskajā datorzinātnē tiek pētīts tas cik ļoti efektīvi iespējams veikt dažādus aprēķinus. Šeit lielu lomu spēlē skaitļošanas problēmu sarežģītība. Dažādi sarežģītības mēri palīdz precīzāk noteikt algoritmu izpildes apakšējās un augšējās robežas, prognozēt iespējamos iznākumus un noteiktos gadījumos uzlabot algoritmu veiktspēju (piemēram, noskaidrojot pie kādiem nepilnīgiem vaicājumiem algoritma iznākums jau ir zināms un tos pielietojot pretstatā pilnai to izpildei). Mērķis ir atrast optimālākos algoritmus, saprast ko var optimizēt un ko nē, noskaidrot nepieciešamo un pietiekamo aprēķinu daudzumu. Kvantu algoritmu pētīšana palīdz saprast un prognozēt kuros tieši gadījumos kvantu algoritmi dos uzlabojumus veiktspējā un ātrdarbībā, kā arī kādi tieši tie būtu. Efektīvāku aprēķinu iespējas pierādīšana kvantu pasaulē arī bija viens no pirmajiem kvantu datorikas reālajiem iekustinātājiem (pievēršanās lieliem ieguldījumiem kvantu datoru izbūvē iespējama tikai tad ja ir pierādāms šo datoru pārākums noteiktu aprēķinu veikšanā).

Latvijas Universitātes Datorikas fakultātē kvantu skaitļošanas pētniecība tika aizsākta jau pagājušā gadsimta 90-to gadu nogalē profesora Rūsiņa Mārtiņa Freivalda vadībā (ar pētījumiem par kvantu galīgiem automātiem) un to veiksmīgi turpina profesors Andris Ambainis. Fakultātē ir izveidojusies pasaules līmeņa kvantu algoritmu sarežģītības pētnieku komanda kurā tiek piesaistīti pētnieki arī no citām valstīm.

Pašlaik fakultātē tiek īstenoti vairāki kvantu datorikas projekti. Viens no tiem ir arī "Kvantu algoritmi" (angliski: *QALGO: Quantum Algorithmics*). Tā ietvaros autoram tika uzdots izstrādāt programmatūru dažādu Būla funkciju raksturojošu mēru aprēķināšanai. Motivācija šādas programmas izstrādei nāk no pētnieku puses, kas ikdienā nodarbojas ar funkciju un algoritmu sarežģītības pētīšanu, kā procesā ir nepieciešams veikt atsevišķu raksturotāju noteikšanu pastarpinātiem algoritmiem vai funkcijām.

Atbilstoši šī darba mērķis bija izstrādāt programmproduktu, kas spētu aprēķināt vairākus Būla funkciju raksturotājus: jūtīgumu, bloku jūtīgumu, pārstāvošos un tuvinātos polinomus, to pakāpes.

Lai sasniegtu mērķi tika izvirzīti sekojoši uzdevumi:

- iepazīties ar teorētisko materiālu par galvenajām Būla funkcijām;

- iepazīties un apgūt konkrētu matemātisko vidi un tās valodu veikt aptuvenu darbietilpības novērtējumu;
- izveidot programmatūras prasību specifikāciju;
- pamatojoties uz programmatūras prasību specifikāciju izveidot atbilstošu programmatūras projektējuma aprakstu;
- izstrādāt pašu programmproduktu;
- veikt produkta vienībtestēšanu un uzlabošanu;
- sagatavot visu pavadošo dokumentāciju.

Izstrādātais programmprodukts ļauj aprēķināt sākotnēji uzstādītos Būla funkciju rakstu-rotājus. Izstrādes gaitā uzmanība pievērsta arī tam, lai programmatūras kods būtu viegli uztverams un papildināms.

Darbs satur programmatūras prasību specifikāciju un programmatūras projektējuma aprakstu. Tā pat ir iekļauta testēšanas dokumentācija. Papildus aprakstīta arī izstrādes organizēšana, veidi kā nodrošināta konfigurācijas pārvaldība un kvalitāte. Veikts darbietilpības novērtējums.

Kā galvenais teorētiskais materiāls par Būla funkcijām tika izmantots Ryan O`Donell grāmata, papildus tika izmantots Krišjāņa Prūša maģistra darbs un arī Scott Aaronson programma, kas ļāva salīdzināt rezultātus. Definīcijas šajā darbā veidotas pamatojoties uz teorētisko materiālu. Lai apgūtu matemātiskās vides pamatus un arī pašā izstrādes gaitā tika izmantoti Wolfram dokumentācijas un StackExchange vietnēs esošie resursi. Darbs tika sagatavots ņemot vērā prasības noslēguma darbu izstrādāšanai un aizstāvēšanai Latvijas Universitātē, Latvijas Universitātes Datorikas fakultātes kvalifikācijas darba izstrādes un aizstāvēšanas metodiskos norādījumus, kā arī atbilstošos nozares standartus LVS 68:1996 “Programmatūras prasību specifikācijas ceļvedis”, LVS 72:1996 “Ieteicamā prakse programmatūras projektējuma aprakstīšanai”.

1. Pamatdefinīcijas un saīsinājumi

Indeksu kopa (arī $[n]$) – $\{1,2,\dots,n\}$.

Bitu skaits – Šajā darbā tiek pieņemts, ka tas ir ≥ 2.

Mainīgais – Šajā darbā ar mainīgo tiek saprasts Būla funkcijas (vai ieejas virknes) konkrētā vietā esošs bits (vai elements).

Ieejas virkne vai **ievads** (arī $\{0,1\}^n$; mēdz saukt arī par Būla virkni vai vārdu) – Ar ieejas virkni garumā n tiek saprasta bitu virkne, kas sastāv no n mainīgajiem. Ieejas virknes x kārtējais elements i tiek apzīmēts ar x_i. Ieejas virkni kurā i mainīgais tiek mainīts uz pretējo apzīmēsim ar x^i. Ieejas virkni x, kur visi mainīgie, kas pieder kopai S ir mainīti uz pretējiem apzīmēsim ar x^S, kur $S \subseteq [n]$.

Būla funkcija (arī $f\colon \{0,1\}^n$) – Ar Būla funkciju uz n bitiem: $f\colon \{0,1\}^n \to \{0,1\}$ tiek saprasta funkcija, kas saņemot ieejas virkni garumā n atbild ar 0 vai 1.

Jūtīgums (arī s) – Saka, ka funkcija f ir jūtīga pret kādu ieejas virknes mainīgo x_i, ja $f(x) \neq f(x^i)$. Saka, ka funkcijas f jūtīgums s uz ieejas virkni x: $s(f,x)$ ir mainīgo x_i skaits kuriem $f(x) \neq f(x^i)$. Atbilstoši, maksimums no jūtīgumiem uz visām iespējamajām 2^n ieejas virknēm ir arī jūtīgums pašai Būla funkcijai $s(f) = \max_{x\in\{0,1\}^n} s(f,x)$ (pēc Krišjānis, 2014).

Bloku jūtīgums (arī bs) – Saka, ka funkcija f ir jūtīga pret bloku B ieejas virknē x ja $f(x) \neq f(x^B)$, kur $B \subseteq [n]$. Saka, ka funkcijas f bloku jūtīgums bs uz ieejas virkni x: $bs(f,x)$ ir maksimālais tādu nešķeļošas indeksu kopas $[n]$ apakškopas komplektu skaits, kur ieejas virkne ir jūtīga pret katru no šiem komplektiem. Atbilstoši, maksimums no bloku jūtīgumiem uz visām iespējamajām 2^n ieejas virknēm ir arī jūtīgums pašai Būla funkcijai $bs(f) = \max_{x\in\{0,1\}^n} bs(f,x)$ (pēc Krišjānis, 2014).

Pārstāvošais polinoms – Par funkcijas f pārstāvošo polinomu p sauc tādu polinomu, ka vienmēr pie vienādām ieejas virknēm, tas izdos tādu pašu vērtību kādu izdod funkcija: $p(x_1,x_2,\dots,x_n) = f(x_1,x_2,\dots,x_n)$

Tuvinātais polinoms – Par tuvināto polinomu p sauc tādu polinomu, kam vērtība no funkcijas f vērtības neatšķirsies vairāk kā par ε: $|p(x_1,x_2,\dots,x_n) - f(x_1,x_2,\dots,x_n)| \leq \varepsilon$.

Patiesumvērtības – Šajā darbā ar patiesumvērtībām tiek saprastas $false, true$.

Reprezentācija – Kāds no veidiem kā tiek uzdoti biti: $false, true$, 0,1 vai $1, -1$.

2. Programmatūras prasību specifikācija

2.1. Ievads

2.1.1. Nolūks

Programmatūras prasību specifikācija ir paredzēta izstrādājamās atbalsta bibliotēkas Būla funkciju izpētei prasību aprakstīšanai. Šīs prasības būs par pamatu tālākai programmprodukta izstrādei. Šajā dokumentā tiek formulētas izstrādājamās bibliotēkas prasības un raksturota tās funkcionalitāte.

Šis dokuments ir paredzēts izstrādājamās atbalsta bibliotēkas pasūtītājam un izstrādātājam. Pasūtītājs to izmanto, lai viennozīmīgi, skaidri un precīzi noteiktu izstrādājamās sistēmas prasības un atsauktos uz tām, bet izstrādātājs to izmantos, lai realizētu pasūtītāja noteiktās prasības. Atbilstoši gan pasūtītājs, gan izstrādātājs varēs pārliecināties par dokumentā iekļauto prasību realizācijas gaitu un pilnību.

2.1.2. Darbības sfēra

Programmatūras moduļa nosaukums ir "Atbalsta bibliotēka Būla funkciju izpētei" (angliski: *Support Library for Analysis of Boolean Functions* vai saīsināti *bfslib*).

Programmproduktam jāļauj lietotājam iegūt lietotāja definēto vai izvēlēto Būla funkciju mērus, kas definēti prasībās.

Programmproduktam jābūt kā izmantojamam modulim vai bibliotēkai, kuru lietotājs var izmantot ikdienā, nosakot atsevišķu Būla funkciju mērus, iestrādāt savā programmproduktā, kā arī, iespējams, papildināt tā iespējas.

2.1.3. Definīcijas un saīsinājumi

Dokumentā lietotās definīcijas un saīsinājumi definēti darba 1. sadaļā.

2.1.4. Saistība ar citiem dokumentiem

Programmatūras prasību specifikācija ir kvalifikācijas darba "Atbalsta bibliotēka Būla funkciju izpētei" daļa un lietojama kopā ar Programmatūras projektējuma aprakstu.

Detalizētākas informācijas un izpratnes iegūšanai, šo dokumentu ieteicams izmantot kopā ar materiālu par Būla funkcijām un to analīzi.

Šis dokuments izstrādāts saskaņā ar standartu LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis". Par pamatu šī dokumenta 3. nodaļas struktūrai izmantots minētajā standartā dotais 1. prototips.

2.1.5. PĀRSKATS

Dokuments ir daļa no kvalifikācijas darba.

Dokuments sastāv no 3 nodaļām.

Pirmajā nodaļā sniegta vispārīga informācija par dokumenta nolūku un programmprodukta darbības sfēru.

Otrā nodaļa sniedz vispārīgu ieskatu produkta funkcionalitātē un uz to attiecināmos ierobežojumos.

Trešā nodaļa apraksta ar programmproduktu saistītās funkcionālās, saskarņu, veiktspējas prasības un ierobežojumus uz aparatūru.

2.2. Vispārējais apraksts

2.2.1. Produkta perspektīva

Produktam jābūt izstrādātam kā funkciju bibliotēkai priekš Būla funkciju izpētes. Produkts ir paredzēts: - lietošanai izstrādājamo Būla funkciju mēru noteikšanai; - izmantošanai lielāku programmproduktu sastāvā nodrošinot funkcionalitāti, kas saistīta ar šeit aprakstītajiem ieviešamajiem funkciju mēriem; - tas var kalpot kā pamats turpmākai jaunu Būla funkciju mēru ieviešanai un izstrādājamā programmprodukta papildināšanai un attīstīšanai pēc tālākām pasūtītāja vajadzībām.

2.2.2. Produkta funkcijas

Produktam ir jārealizē vienkāršu Būla funkciju mēru notikšana. Tam ir jāspēj noteikt sekojošie Būla funkciju mēri: jūtīgums, bloku jūtīgums, pārstāvošā un tuvinātā polinoma pakāpes un paši polinomi.

Produktam ir jāspēj darboties ar Būla funkcijām $f: \{0,1\}^n$, kur bitu skaits n ir patvaļīgs.

2.2.3. Lietotāja raksturiezīmes

Produkta lietotāji būs kvantu algoritmu un Būla funkciju pētnieki ar atbilstošām priekšzināšanām.

Lielākoties produktu paredzēts lietot ar ārējo grafisko saskarni. Tā pat produktu lietos izvēlētajā matemātiskajā vidē, lai izmatojot iekšējās matemātiskās vides programmēšanas saskarnes darbībām un manipulācijām, kas ietver produktā izstrādājamo funkciju mērus. Atsevišķi lietotāji var izmantot produkta pirmkodu, lai to papildinātu un uzlabotu.

Lietotāji komunikācijai izmanto angļu valodu.

2.2.4. Vispārējie ierobežojumi

Produkts jāizstrādā angļu valodā. Tas attiecas gan uz kodu, gan komentāriem, gan arī lietotāja dokumentāciju.

2.2.5. Pieņēmumi un atkarības

Tiek pieņemts, ka produkts tiks izmantots uz datoriem kuros būs pieejama atbilstošā izstrādātāja izvēlētā matemātiskā vide, bet lai izmantotu grafisko saskarni matemātiskā vide nav obligāti vajadzīga. Tiek pieņemts, ka produkts galvenokārt tiks izmantots ar Būla funkciju un kvantu algoritmu izpēti saistītos uzdevumos un saistītos pētniecības projektos. Tiek pieņemts, ka iekšējo matemātiskās vides programmēšanas saskarni izmantos lietotājs ar zināšanām un izpratni par izstrādātāja izvēlēto matemātisko vidi.

2.3. Konkrētās prasības

2.3.1. Funkcionālās prasības

Ja funkcijai ir iespējamas vairākas ievades, tad ar to jāsaprot, ka aprakstīti vairāki funkciju prototipi (skatoties no ievades viedokļa), kam jāveic līdzīga aprakstītā ievaddatu apstrāde un jāizdod viens un tas pats norādītais rezultāts. Ja konkrētai ievadei ir norādītas dažādas reprezentācijas, tad arī šajā gadījumā (atkarībā no izvēlētās matemātiskās vides iespējām) tās var nākties uztvert kā pamatu vairāku funkciju prototipiem.

Ievades piemēri norādīti aptuveni un tie adaptējami pēc izvēlētās matemātiskās vides (piem. vai masīvs ir kvadrātiekavās – [ ] vai figūriekavās – { }, vai apaļajās iekavās – ( ) u.tml.).

2.3.1.1. Būla funkciju ievade un izvade

2.3.1.1.1. Būla funkcijas uzdošana

Ievads	Lai lietotājs varētu ērti strādāt ar dažādām funkcijām un neuztraukties par sarežģītiem vēlamo mēru iegūšanai nepieciešamajiem citu funkciju izsaukumiem jābūt iespējai pašu Būla funkciju lietot kā argumentu. Tādēļ nepieciešams ģenerēt ievadē aprakstītu Būla funkciju, ko var padot citām funkcijām.
Ievade	Iespējamie ievadi: 1. Funkcijas definīcija (ja matemātiskajā vidē ir iebūvētas Būla funkcijas vai iespēja tās ieviest no lietotāja ievades, citādāk dinamiski ģenerēt darbības laikā; piem. $If\ \ VisiArgumentiVienādi == true, false, true$) vai *NotEqual*; 2. Atbilžu vektors $0,1,\ 1,-1$ vai $false, true$ reprezentācijā (piem. [0,0,0,1], [1,1,1,−1] vai [*false*, *false*, *false*, *true*]; papildus pirms tam var būt norādīts funkcijas bitu skaits); 3. Patiesību tabula $0,1,\ 1,-1$ vai $false, true$ reprezentācijā (piem. $[0,0] \rightarrow 0, [0,1] \rightarrow 1, [1,0] \rightarrow 1, [1,1] \rightarrow 1)$, $[1,1] \rightarrow 1, [1,-1] \rightarrow -1, [-1,1] \rightarrow -1, [-1,-1] \rightarrow -1)$ vai $[false, false] \rightarrow false, [false, true] \rightarrow true, [true, false] \rightarrow true, [true, true] \rightarrow true$)). Argumentu skaits un funkcijas definīcija var atšķirties izejot no bitu skaita uz kuru aprakstītā funkcija darbojas. 2. ievades gadījumā atbilžu skaitam jābūt 2^n, kur n ir bitu skaits aprakstītajai funkcijai. 3. ievades gadījumā patiesību tabulai jāsastāv no 2^n piekārtojumiem, kur n ir bitu skaits aprakstītajai funkcijai. Katram no šiem piekārtojumiem funkcijas vērtībai atbilstošajam ievadam jāsastāv no n mainīgajiem.
Apstrāde	Balstoties uz ievadi dinamiski jāizveido ievadē aprakstītā funkcija vai objekts.
Izvade	Jāatgriež piešķirama (piem., $a = Iegūtā Funkcija$) un izsaucama (piem. $Iegūtā Funkcija\ True, False$) vai jau no piešķirtās $a\ True, False$)) funkcija vai objekts.

2.3.1.1.2. Būla funkcijas atbilžu saraksta ģenerēšana

Ievads	Lietotājam nepieciešama iespēja ievadītās vai izvēlētās iebūvētās Būla funkcijas arī eksportēt. Funkcija sagatavo Būla funkciju raksturojošo patiesumvērtību atbilžu komplektu. Tas ļauj izmantot funkcijas atbilžu tabulu, lai to saglabātu vai arī varētu tālāk padot pasūtītāja esošajās lietojumprogrammās cita veida analīzēm.
Ievade	Iespējamie ievadi: 1. Būla funkcija un bitu skaits (piem. $And, 4$); 2. Būla funkcija, bitu skaits un reprezentācija (piem. $And, 4, -1$). Ja reprezentācija nav norādīta, tad tā ir 0,1.
Apstrāde	Padotajai Būla funkcijai jāuzdod aprēķināt atbildes uz visām iespējamajām ieejas vērtībām.
Izvade	Jāatgriež atbilstošais patiesumvērtību komplekts pret izsaukumu (piem., $And, 2$ gadījumā – 0,0,0,1) vai $And, 2, -1$ gadījumā – 1,1,1, –1)).

2.3.1.2. Būla funkcijas raksturojošu mēru aprēķināšana

2.3.1.2.1. Būla funkcijas jūtīguma aprēķināšana

Ievads	Jūtīgums parāda to cik atsevišķu bitu maiņa uz pretējiem, sliktākajā gadījumā, mainīs visas Būla funkcijas vērtību.
Ievade	Būla funkcija un bitu skaits (piem. $Majority, 3$).
Apstrāde	Pārbauda katras ieejas virknes katra mainīgā, nomainīta uz pretējo vērtību, ietekmi uz funkcijas rezultātu. Skatās maksimālo nomainīto mainīgo daudzumu, kas mainījuši funkcijas rezultātu konkrētai ievades virknei.
Izvade	Atgriež maksimālo mainīgo skaitu, kas ietekmē kādas no virknēm rezultātu (piem., $Majority, 3$ gadījumā – 2).

2.3.1.2.2. Būla funkcijas bloku jūtīguma aprēķināšana

Ievads	Bloku jūtīgums parāda to cik atsevišķu bloku bitu maiņa uz pretējiem, sliktākajā gadījumā, mainīs visas Būla funkcijas vērtību, kur bloki pieder nešķeļošai indeksu kopas apakškopai.
Ievade	Būla funkcija un bitu skaits (piem. $Xor, 3$).
Apstrāde	Pārbauda katras ieejas virknes katra bloka, nomainīta uz pretējo vērtību, ietekmi uz funkcijas rezultātu, kur bloki pieder nešķeļošai indeksu kopas apakškopai. Skatās maksimālo vienas nešķeļošas indeksu kopas apakškopas nomainīto bloku daudzumu, kas mainījuši funkcijas rezultātu konkrētai ievades virknei.
Izvade	Atgriež maksimālo nešķeļošu indeksu kopas apakškopas bloku skaitu, kuru mainīgo maiņa ietekmē kādas no virknēm rezultātu (piem., $Xor, 3$ gadījumā – 3).

2.3.1.2.3. Būla funkciju pārstāvošā polinoma ģenerēšana

Ievads	Pārstāvošais polinoms ir svarīgs tādēļ, ka tas, tā pat kā funkcijas atbilžu saraksts, ir unikāls priekš katras Būla funkcijas.

Ievade	Iespējamie ievadi: 1. Būla funkcija un bitu skaits (piem. $Majority, 4$); 2. Būla funkcija, bitu skaits un reprezentācija (piem. $Majority, 4, -1$). Ja reprezentācija nav norādīta, tad tā ir $0,1$.
Apstrāde	Jāaprēķina pārstāvošais polinoms izmantojot lineāro programmu vai arī algoritmu.
Izvade	Jāatgriež atbilstošais Būla funkciju pārstāvošais polinoms (piem., $Majority, 3$ gadījumā – $x_1x_2 + x_1x_3 + x_2x_3 - 2x_1x_2x_3$).

2.3.1.2.4. TUVINĀTA POLINOMA APRĒĶINĀŠANA AR 2-PUSĒJU KĻŪDU

Ievads	Polinoms, kas ar noteiktu ticamību dos tādu pašu rezultātu kā pārstāvošais polinoms ir noderīgs tādos gadījumos ja tā pakāpe pie dotās kļūdas ir mazāka nekā pārstāvošajam, kas var atvieglot aprēķinus.
Ievade	Iespējamie ievadi: 1. Būla funkcija, bitu skaits un pieļaujamās kļūdas lielums (piem. $Xor, 4, 1/3$); 2. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un reprezentācija (piem. $Xor, 4, 1/3, -1$); 3. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un vēlamā maksimālā pakāpe (piem. $Xor, 4, 1/3, 2$); 4. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums, vēlamā maksimālā pakāpe un reprezentācija (piem. $Xor, 3, 1/3, 2, 0$). Ja reprezentācija nav norādīta, tad tā ir $0,1$.
Apstrāde	Ar 2-pusēju kļūdu jāsaprot, ka katru ievades virkni pārstāvošais multilineārais polinoms ir no $FunkcijasVērtība - Kļūdas Lielums$ līdz $FunkcijasVērtība + Kļūdas Lielums$. Izmantojot lineāro programmu noteikt kādu no tuvinātajiem polinomiem. Ja tuvinātais polinoms ar mazāku pakāpi pie dotās pieļaujamās kļūdas nav atrodams, tad atgriezt pārstāvošo. 2. un 4. ievades gadījumā ja nav iespējams atgriezt polinomu ar doto maksimālo pakāpi pie dotās pieļaujamās kļūdas, tad to darīt zināmu lietotājam.
Izvade	Atgriež tuvinātu polinomu (piem., $Xor, 2, 1/3$ gadījumā – $x_1 + x_2 - 2x_1x_2$).

2.3.1.2.5. TUVINĀTA POLINOMA APRĒĶINĀŠANA AR 2-PUSĒJU KĻŪDU IEKŠPUSĒ

Ievads	Polinoms, kas ar noteiktu ticamību dos tādu pašu rezultātu kā pārstāvošais polinoms ir noderīgs tādos gadījumos ja tā pakāpe pie dotās kļūdas ir mazāka nekā pārstāvošajam, kas var atvieglot aprēķinus. Funkcija ir identiska tuvinātā polinoma aprēķināšanai ar 2-pusēju kļūdu vienīgi šoreiz netiek pieļautas kļūdas līdz nepatiesībai un no patiesības.

Ievade	Iespējamie ievadi: 1. Būla funkcija, bitu skaits un pieļaujamās kļūdas lielums (piem. $Xor, 4, 1/3$); 2. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un reprezentācija (piem. $Xor, 4, 1/3, -1$); 3. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un vēlamā maksimālā pakāpe (piem. $Xor, 4, 1/3, 2$); 4. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums, vēlamā maksimālā pakāpe un reprezentācija (piem. $Xor, 3, 1/3, 2, 0$). Ja reprezentācija nav norādīta, tad tā ir 0,1.
Apstrāde	Ar 2-pusēju kļūdu jāsaprot, ka katru ievades virkni pārstāvošais multilineārais polinoms ir no $FunkcijasVērtība + KļūdasLielums$ nepatiesības gadījumā līdz $FunkcijasVērtība - KļūdasLielums$ patiesības gadījumā. Izmantojot lineāro programmu noteikt kādu no tuvinātajiem polinomiem. Ja tuvinātais polinoms ar mazāku pakāpi pie dotās pieļaujamās kļūdas nav atrodams, tad atgriezt pārstāvošo. 2. un 4. ievades gadījumā ja nav iespējams atgriezt polinomu ar doto maksimālo pakāpi pie dotās pieļaujamās kļūdas, tad to darīt zināmu lietotājam.
Izvade	Atgriež tuvinātu polinomu (piem., $Xor, 2, 1/3$ gadījumā – $x_1 + x_2 - 2x_1x_2$).

2.3.1.2.6. TUVINĀTA POLINOMA APRĒĶINĀŠANA AR 1-PUSĒJU KĻŪDU 1-PUSĒ

Ievads Šī metode dos novirzi līdz patiesībai pret pārstāvošo polinomu.

Ievade	Iespējamie ievadi: 1. Būla funkcija, bitu skaits un pieļaujamās kļūdas lielums (piem. $Xor, 4, 1/3$); 2. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un reprezentācija (piem. $Xor, 4, 1/3, -1$); 3. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un vēlamā maksimālā pakāpe (piem. $Xor, 4, 1/3, 2$); 4. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums, vēlamā maksimālā pakāpe un reprezentācija (piem. $Xor, 3, 1/3, 2, 0$). Ja reprezentācija nav norādīta, tad tā ir 0,1.
Apstrāde	Ar 1-pusēju kļūdu jāsaprot, ka katru ievades virkni pārstāvošais multilineārais polinoms ir no $FunkcijasVērtība - KļūdasLielums$ līdz $FunkcijasVērtība$ tikai gadījumā ja funkcijas vērtība tajā ir patiesa, citādāk tas vienāds ar funkcijas vērtību. Izmantojot lineāro programmu noteikt kādu no tuvinātajiem polinomiem. Ja tuvinātais polinoms ar mazāku pakāpi pie dotās pieļaujamās kļūdas nav atrodams, tad atgriezt pārstāvošo. 2. un 4. ievades gadījumā ja nav iespējams atgriezt polinomu ar doto maksimālo pakāpi pie dotās pieļaujamās kļūdas, tad to darīt zināmu lietotājam.

Izvade	Atgriež tuvinātu polinomu (piem., $Xor, 2, {}^{1}/_{3}$ gadījumā – $x_1 + x_2 - 2x_1x_2$).

2.3.1.2.7. TUVINĀTA POLINOMA APRĒĶINĀŠANA AR 1-PUSĒJU KĻŪDU 0-PUSĒ

Ievads	Šī metode dos novirzi no nepatiesības pret pārstāvošo polinomu.
Ievade	Iespējamie ievadi: 1. Būla funkcija, bitu skaits un pieļaujamās kļūdas lielums (piem. $Xor, 4, {}^{1}/_{3}$); 2. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un reprezentācija (piem. $Xor, 4, {}^{1}/_{3}, -1$); 3. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un vēlamā maksimālā pakāpe (piem. $Xor, 4, {}^{1}/_{3}, 2$); 4. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums, vēlamā maksimālā pakāpe un reprezentācija (piem. $Xor, 3, {}^{1}/_{3}, 2, 0$). Ja reprezentācija nav norādīta, tad tā ir 0,1.
Apstrāde	Ar 1-pusēju kļūdu jāsaprot, ka katru ievades virkni pārstāvošais multilineārais polinoms ir no $FunkcijasVērtība + KļūdasLielums$ līdz $FunkcijasVērtība$ tikai gadījumā ja funkcijas vērtība tajā ir aplama, citādāk tas vienāds ar funkcijas vērtību. Izmantojot lineāro programmu noteikt kādu no tuvinātajiem polinomiem. Ja tuvinātais polinoms ar mazāku pakāpi pie dotās pieļaujamās kļūdas nav atrodams, tad atgriezt pārstāvošo. 2. un 4. ievades gadījumā ja nav iespējams atgriezt polinomu ar doto maksimālo pakāpi pie dotās pieļaujamās kļūdas, tad to darīt zināmu lietotājam.
Izvade	Atgriež tuvinātu polinomu (piem., $Xor, 2, {}^{1}/_{3}$ gadījumā – $x_1 + x_2 - 2x_1x_2$).

2.3.1.2.8. REDUCĒT POLINOMU PĒC MODUĻA 2

Ievads	Ja polinoms ir reducēts pēc moduļa 2, tad iespējams to ērtāk izmantot 0,1 reprezentācijā un tas var sastāvēt no mazāk monomiem, tai skaitā būt ar mazāku pakāpi.
Ievade	Polinoms (piem. $x_1x_2 + x_1x_3 + x_2x_3 - 2x_1x_2x_3$).
Apstrāde	Polinoms jāreducē pēc moduļa 2.
Izvade	Atgriež reducētu polinomu (piem., $x_1 + x_2 - 2x_1x_2$ gadījumā – $x_1 + x_2$).

2.3.1.3. PALĪGFUNKCIJAS

2.3.1.3.1. POLINOMU PĀRVEIDE STARP REPREZENTĀCIJĀM

Ievads	Atkarībā no lietotāja vajadzībām ir nepieciešama iespēja polinoma pārveidot starp reprezentācijām, lai varētu strādāt ar vēlamo.

Ievade	Iespējamie ievadi: 1. Polinoms (piem. $x_1x_2 + x_1x_3 + x_2x_3 - 2x_1x_2x_3$); 2. Polinoms un vēlamā reprezentācija (piem. $x_1x_2 + x_1x_3 + x_2x_3 - 2x_1x_2x_3, -1$). Gadījumā ja norādīts tikai bitu skaits, tad uztvert, ka vēlamā reprezentācija ir $1, -1$. Nav iespējams zināt ka polinoms pārstāv vienu vai otru reprezentāciju, tādēļ tiek pieņemts, ka lietotājs veicot ievadi ir izvēlējies tieši pretējo reprezentāciju pārveidojamajam polinomam.
Apstrāde	Pārveidot polinomu uz pretējo reprezentāciju.
Izvade	Atgriež pretējo reprezentāciju pārstāvošu polinomu (piem., $x_1x_2 + x_1x_3 + x_2x_3 - 2x_1x_2x_3, -1$ gadījumā – $1 - 4x_1x_2 - 4x_1x_3 - 4x_2x_3 + 16x_1x_2x_3$).

2.3.1.3.2. POLINOMA PAKĀPES NOTEIKŠANA

Ievads	Funkcija nodrošina pakāpes atrašanu jau iegūtam polinomam (piem. pārstāvošajam, tuvinātajam vai reducētam polinomam).
Ievade	Polinoms (piem. $x_1x_2 + x_1x_3 + x_2x_3 - 2x_1x_2x_3$).
Apstrāde	Jānosaka polinoma pakāpe.
Izvade	Atgriež polinoma pakāpi (piem., $x_1x_2 + x_1x_3 + x_2x_3 - 2x_1x_2x_3$ gadījumā – 3).

2.3.2. ĀRĒJĀS SASKARNES PRASĪBAS

2.3.2.1. GRAFISKĀ LIETOTĀJA SASKARNE

Atbalsta bibliotēkas Būla funkciju izpētei manuāli lietojamajām funkcijām jābūt pieejamām no matemātiskās vides caur failu ar to definīcijām, caur ielādējamu moduli vai pielāgotu matemātiskās vides instalāciju.

Atbalsta bibliotēkas Būla funkciju izpētei grafiskā lietotāja saskarnes izveidei ir šādi ierobežojumi:

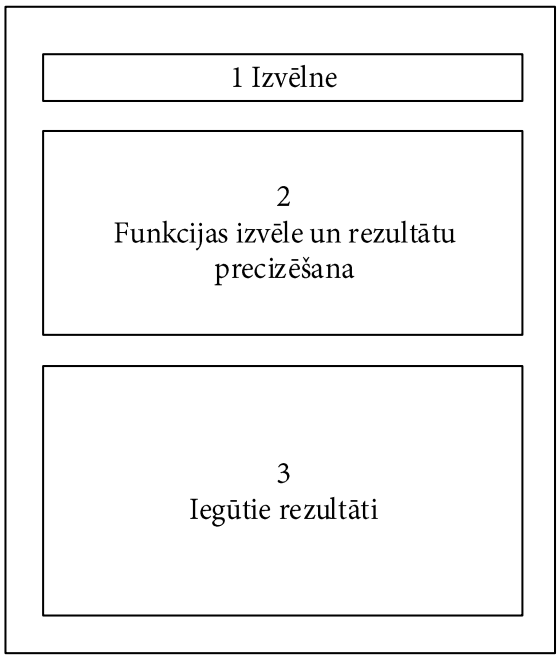
- jābūt angļu valodā;
- nosaukumiem vai saīsinājumiem jābūt viennozīmīgi saprotamiem un atbilstošiem lietotāju nozarē pieņemtajiem;
- tai jābūt izmantojamai bez pilnas matemātiskās vides (ja izvēlēta komerciāla versija) vai ar pieejamas bezmaksas palīgprogrammu.

Lietotāja saskarnei jābūt veidotai no trim blokiem (skat. 2.1. attēlu):

- izvēļņu joslas (1), kurai jāpiedāvā:
  - ielasīt Būla funkciju no faila;
  - saglabāt esošo Būla funkciju failā;
  - saglabāt iegūtos rezultātus failā;
- ievades bloka (2), kuram jāpiedāvā:
  - manuāli ievadīt Būla funkciju sniedzot secīgas atbildes;

- izvēlēties kādu no definētajām Būla funkcijām;
- izvēlēties vai ievadīt pieļaujamās kļūdas lielumu atbilstošajiem mēriem;
- izvēlēties vai rādīt polinomus;

- rezultātu bloka (3), kurā jābūt redzamiem mēram un tam atbilstošajai vērtībai.



2.1. attēls. **Lietotāja saskarne**

2.3.2.2. PROGRAMMATŪRAS SASKARNE

Programmproduktam jāstrādā gan uz datoriem ar Windows 8.1, gan uz datoriem ar Ubuntu 14.04 LTS.

Produkts jāizstrādā matemātiskā vidē (tādā kā MATLAB, Mathematica, Maple, SageMath vai līdzvērtīgā), lai lietotājam būtu iespējams ērti izmantot un kombinēt citas matemātiskās analīzes metodes Būla funkciju izpētē. Iespēju robežās (ar to saprotot – kur iespējams), izstrādes posmā jāizmanto iebūvētās metodes. Izstrādātājam jāvienojas ar pasūtītāju par izvēlētās matemātiskās vides izmantošanu izejot no pašreizējās pasūtītāja puses zināšanām / pieredzes / gatavības apgūt izstrādātāja izvēlēto matemātisko vidi un tās nākotnes izmantošanas iespējām atbilstoši pasūtītāja plānotajai izmantošanai. Gadījumā ja matemātiskā vide nav brīvpieejas, tad izstrādātājam ar pasūtītāju jāvienojas par tās izmantošanu izejot no tās izmaksām, pieejamības.

Produktam, papildus matemātiskās vides saskarnei, jānodrošina lietotāja saskarne, kas ļauj viennozīmīgi un lietotājam saprotami (izmantojot lietotāja sfēras terminoloģiju un pieņemtus saīsinājumus) iegūt produktā iestrādātos mērus.

2.3.3. Veiktspējas prasības

Katram raksturotājam, kas strādā ar Būla funkcijām līdz 7 bitiem, jāiekļaujas aprēķinu laikā līdz 5 minūtēm.

2.3.4. Aparatūras ierobežojumi

Programmproduktam jābūt spējīgam darboties uz datora ar 2GHz procesoru un 2GB operatīvo atmiņu, kam papildus ir jāatbilst izvēlētās matemātiskās vides minimālajām prasībām.

3. Programmatūras projektējuma apraksts

3.1. Ievads

3.1.1. Nolūks

Programmatūras projektējuma apraksts paredzēts programmatūras prasību specifikācijā atbalsta bibliotēkai Būla funkciju izpētei minēto prasību aprakstīšanai un konkretizēšanai.

Šis dokuments ir paredzēts projektētājiem, ieviesējiem, uzturētājiem un testētājiem. Tas ir paredzēts, lai sekmētu programmprodukta plānošanu, ieviešanu un analīzi.

3.1.2. Darbības sfēra

Atbalsta bibliotēkas Būla funkciju izpētei mērķis ir nodrošināt vienkāršu Būla funkciju mēru noteikšanu. Programmprodukts ļaus lietotājam vienkāršos mērus apstrādāt grafiski un arī nodrošinās iespēju izmantot ieviestās metodes, lai tās papildinātu vai izmantotu matemātiskajā vidē darbā ar sarežģītākiem mēriem.

3.1.3. Definīcijas un saīsinājumi

Dokumentā lietotās definīcijas un saīsinājumi definēti darba 1. sadaļā.

3.1.4. Saistība ar citiem dokumentiem

Programmatūras projektējuma apraksts veidots balstoties uz programmatūras prasību specifikāciju Tas ir kvalifikācijas darba "Atbalsta bibliotēka Būla funkciju izpētei" daļa un lietojams kopā ar Programmatūras prasību specifikāciju un Testēšanas dokumentāciju.

Detalizētākas informācijas un izpratnes iegūšanai, šo dokumentu ieteicams izmantot kopā ar materiālu par Būla funkcijām un to analīzi.

Šis dokuments izstrādāts saskaņā ar standartu LVS 72-1996 "Ieteicamā prakse programmatūras projektējuma aprakstīšanai".

3.2. Matemātiskā vide

Balstoties uz programmatūras prasību specifikācijā uzliktajiem ierobežojumiem uz izstrādes vidi un programmatūras saskarni nolemts apskatīt iespējamos variantus.

Tika nolemts apskatīt sekojošās matemātiskās vides: Maple, Mathcad, Mathematica, MATLAB, SageMath.

Mathcad nav izmantojams, jo neatbalsta programmatūras saskarnē noteikto iespēju darbam Linux vidē. SageMath nav izmantojams, jo tas neatbalsta programmatūras saskarnē noteikto iespēju darbam Windows vidē. Mathcad un SageMath var izmantot caur virtuālo mašīnu, tomēr, tas nespēs nodrošināt salīdzināmu ar platformai pielāgotu programmu ātrumu lielu bitu funkciju gadījumā.

Maple nav izmantojams, jo pasūtītāja rīcībā nav produkta lietošanas licenču un pasūtītājs ir noraidījis iespēju pašlaik tās iegādāties.

Atlikušajām vidēm izejot no programmatūras prasību specifikācijas veikta salīdzināšana (skat. 3.1. tabulu).

3.1. tabula. **MATLAB un Mathematica PPS programmatūras saskarnes uzstādījumu salīdzinājums**

	MATLAB	**Mathematica**
Brīvpieejas	- (pasūtītājam ir licences)	- (pasūtītājam ir licences)
Windows 8.1 atbalsts	+	+
Ubuntu 14.04 LTS atbalsts	+	+
Iespēja veidot interaktīvu grafisko saskarni	+	+
Implementēta Būla funkcijas uzdošana	-	+

Pamatojoties uz Mathematica jau ieviesto Būla funkciju ģenerēšanu un palīgfunkcijām kā matemātiskā vide priekš šī programmprodukta izstrādes izvēlēta Mathematica.

3.3. Dekompozīcijas apraksts

Balstoties uz programmatūras prasību specifikāciju un izvelēto matemātisko vidi tika nolemts funkcionālās prasības skatīt kā entītijas. Atbilstoši programmproduktu nolemts veidot sekojoši:

- izveidot entītiju *BooleanFunctionCreate*, kas uzturēs Būla funkciju izveidi;
- izveidot entītiju *BooleanFunctionTable*, kas nodrošinās iespēju Būla funkcijas eksportēt;
- izveidot entītiju *BooleanFunctionSensitivity*, kas uzturēs jūtīguma aprēķināšanu;
- izveidot entītiju *BooleanFunctionBlockSensitivity*, kas uzturēs bloku jutīguma aprēķināšanu;
- izveidot entītiju *BooleanFunctionRepresentingPolynomial*, kas nodrošinās pārstāvošā polinoma atrašanu;
- izveidot entītiju *BooleanFunction2ApproximatePolynomial*, kas nodrošinās iespēju atrast tuvinātu polinomu pie divpusējas kļūdas;
- izveidot entītiju *BooleanFunction2InApproximatePolynomial*, kas nodrošinās iespēju atrast tuvinātu polinomu pie divpusējas kļūdas no nepatiesības un līdz patiesībai;
- izveidot entītiju *BooleanFunction11ApproximatePolynomial*, kas nodrošinās iespēju atrast tuvinātu polinomu pie vienpusējas kļūdas līdz patiesībai;
- izveidot entītiju *BooleanFunction10ApproximatePolynomial*, kas nodrošinās iespēju atrast tuvinātu polinomu pie vienpusējas kļūdas no nepatiesības;
- izveidot entītiju *BooleanFunctionPolynomialModulus2*, kas uzturēs iespēju polinomu reducēt pēc moduļa 2;
- izveidot entītiju *BooleanFunctionPolynomialConvert*, kas uzturēs polinomu konversāciju starp reprezentācijām;
- izveidot entītiju *BooleanFunctionPolynomialDegree*, kas nodrošinās iespēju noteikt polinoma pakāpi.

Var pamanīt, ka vairākām funkcijām būs nepieciešami līdzīgi funkcionālī aprēķini. Atbilstoši nolemts tos izdalīt sekojošās vispārīgās funkcijās: *boolConvertToIntegerValues*, *boolConvertToTruthValues*, *boolGenerateBinaryStrings*, *boolGenerateMultivariatePolynomial*, *boolGenerateVariableArray* un *boolPolynomial*.

Papildus Mathematica jau iebūvētajiem Būla operatoriem (*And*, *Or*, *Not*, *Nand*, *Nor*, *Xor*, *Xnor* un *Majority*) nolemts papildus ieviest vēl sekojošos vienkāršos bieži izmantojamos operatorus: *Equal*, *Parity*, *Threshold*(k), *Exact*(k) un *NEqual*.

3.3.1. Entītiju dekompozīcija

Par entītijas identificējumu tiek uzskatīts tās nosaukums.

Tā kā visām entītijām tips ir funkcija, tad tas nav ticis norādīts.

3.3.1.1. BooleanFunctionCreate

Nolūks	Sagatavot Būla funkciju.
Funkcija	Ģenerē Būla funkciju.
Pakļautība	boolConvertToTruthValues

3.3.1.2. BooleanFunctionTable

Nolūks	Nodrošināt funkcijas identificēšanai nepieciešamo informāciju.
Funkcija	Aprēķina funkcijas vērtību katrai iespējamajai ievadei.
Pakļautība	boolGenerateBinaryStrings, boolConvertToIntegerValues

3.3.1.3. BooleanFunctionSensitivity

Nolūks	Nodrošināt Būla funkcijas jūtīguma aprēķinu.
Funkcija	Aprēķina funkcijas katra mainīgā ietekmi uz katru ievadi skatoties vai konkrētā mainīgā maiņa uz pretējo kādai ievadei ir mainījusi funkcijas vērtību. Noskaidro maksimālo mainīgo skaitu kuru maiņa uz pretējo konkrētajai ievadei ir mainījusi funkcijas vērtību.
Pakļautība	BooleanFunctionTable, boolGenerateBinaryStrings

3.3.1.4. BooleanFunctionBlockSensitivity

Nolūks	Nodrošināt Būla funkcijas bloku jūtīguma aprēķinu.
Funkcija	Katrai ievadei, kur blokam atbilstošie ievades elementi tiek mainīti uz pretējiem skatās vai funkcijas vērtība nav mainījusies, kur bloks pieder kādai nešķeļošas indeksu kopas apakškopai. Noskaidro maksimālo kādas nešķeļošas indeksu kopas apakškopas blokiem atbilstošo kādas ievades daļu maiņas uz pretējām vērtībām skaitu, kur šī maiņa ir mainījusi funkcijas vērtību.
Pakļautība	BooleanFunctionTable, boolGenerateBinaryStrings

3.3.1.5. BooleanFunctionRepresentingPolynomial

Nolūks	Aprēķināt Būla funkciju pārstāvošo polinomu.
Funkcija	Ģenerē pārstāvošo polinomu.
Pakļautība	BooleanFunctionTable, BooleanFunctionPolynomialConvert, boolConvertToIntegerValues, boolGenerateBinaryStrings, boolGenerateVariableArray

3.3.1.6. BooleanFunction2ApproximatePolynomial

Nolūks	Aprēķināt Būla funkcijai pārstāvošajam polinomam tuvināto polinomu ar divpusēju kļūdu.
Funkcija	Ģenerē polinomu, kas argumentu aizvietošanas gadījumā ar ievades bitiem izdod rezultātu ar ticamību kļūdas robežās uz abām pusēm no Būla funkcijas vērtības.

Pakļautība	boolPolynomial

3.3.1.7. BooleanFunction2InApproximatePolynomial

Nolūks	Aprēķināt Būla funkcijai pārstāvošajam polinomam tuvināto polinomu ar divpusēju kļūdu no nepatiesības un līdz patiesībai.
Funkcija	Ģenerē polinomu, kas argumentu aizvietošanas gadījumā ar ievades bitiem izdod rezultātu ar ticamību kļūdas robežās no nepatiesības un līdz patiesībai no Būla funkcijas vērtības.
Pakļautība	boolPolynomial

3.3.1.8. BooleanFunction11ApproximatePolynomial

Nolūks	Aprēķināt Būla funkcijai pārstāvošajam polinomam tuvināto polinomu ar vienpusēju kļūdu.
Funkcija	Ģenerē polinomu, kas argumentu aizvietošanas gadījumā ar ievades bitiem izdod pareizu rezultātu nepatiesajām Būla funkcijas vērtībām un rezultātu ar ticamību kļūdas robežās līdz patiesībai pret patiesajām Būla funkcijas vērtībām.
Pakļautība	boolPolynomial

3.3.1.9. BooleanFunction10ApproximatePolynomial

Nolūks	Aprēķināt Būla funkcijai pārstāvošajam polinomam tuvināto polinomu ar vienpusēju kļūdu.
Funkcija	Ģenerē polinomu, kas argumentu aizvietošanas gadījumā ar ievades bitiem izdod pareizu rezultātu patiesajām Būla funkcijas vērtībām un rezultātu ar ticamību kļūdas robežās līdz nepatiesībai pret nepatiesajām Būla funkcijas vērtībām.
Pakļautība	boolPolynomial

3.3.1.10. BooleanFunctionPolynomialModulus2

Nolūks	Nodrošināt polinoma pārveidi pēc moduļa 2.
Funkcija	Reducēt polinomu pēc moduļa 2.
Pakļautība	-

3.3.1.11. BooleanFunctionPolynomialConvert

Nolūks	Uzturēt iespēju pāriet no vienas reprezentācijas citā.
Funkcija	Pārveido polinomu starp reprezentācijām
Pakļautība	-

3.3.1.12. BooleanFunctionPolynomialDegree

Nolūks	Realizēt polinoma pakāpes noteikšanu.
Funkcija	Aprēķina polinoma pakāpi.

Pakļautība	-

3.3.1.13. boolConvertToIntegerValues

Nolūks	Ļaut funkciju rezultātus pārveidot eksportēšanai vai aprēķiniem derīgā formā.
Funkcija	Pārveido patiesumvērtības par skaitliskajām.
Pakļautība	-

3.3.1.14. boolConvertToTruthValues

Nolūks	Nodrošināt iespēju ievadi pārveidot funkcijai saprotamā veidā.
Funkcija	Pārveido patiesumvērtības par skaitliskajām.
Pakļautība	-

3.3.1.15. boolGenerateBinaryStrings

Nolūks	Realizēt visu iespējamo ievadu iegūšanu.
Funkcija	Ģenerē visus iespējamos ievadus.
Pakļautība	-

3.3.1.16. boolGenerateMultivariatePolynomial

Nolūks	Nodrošināt polinomu aprēķināšanai nepieciešamo datu sagatavošanu.
Funkcija	Ģenerē simbolisku multilineāru polinomu ar koeficientiem.
Pakļautība	boolGenerateVariableArray

3.3.1.17. boolGenerateVariableArray

Nolūks	Nodrošināt izvadei un polinomu aprēķināšanai nepieciešamo datu sagatavošanu.
Funkcija	Izveido mainīgo sarakstu.
Pakļautība	-

3.3.1.18. boolPolynomial

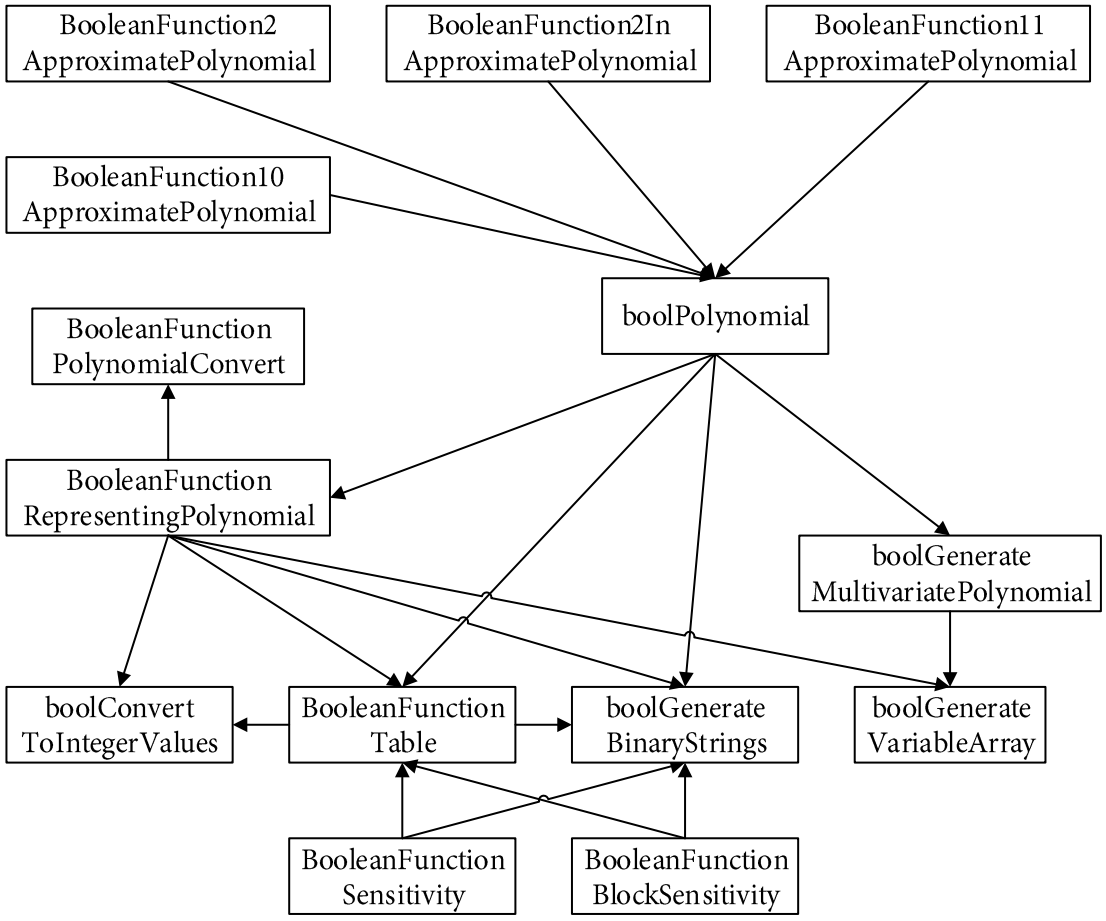
Nolūks	Nodrošināt polinomu aprēķinu.
Funkcija	Ģenerē polinomu izmantojot lineāro programmu.
Pakļautība	boolGenerateMultivariatePolynomial, boolGenerateBinaryStrings, BooleanFunctionTable, BooleanFunctionRepresentingPolynomial

3.3.2. Datu dekompozīcija

Par Būla funkciju saturošu failu uzskatāms fails, kur sākumā norādīts funkcijas bitu skaits un pēc tā seko atdalītas atbildes līdz pat 2^bitu skaitam. Atbildes ir vai nu 0 vai 1. Atstarpes starp simboliem ir vai nu Windows vai Unix jaunas rindas simboli vai kādas no atstarpēm.

3.4. Atkarības apraksts

Tālāk attēlotas atkarības starp dekompozīcijā aprakstītajām entītijām, kur no vienas entītijas novilkta šķautne uz nākošo, tad ja pirmā entītija ir atkarīga no otrās (skat. 3.1. attēlu, kur grafā redzami atkarību atribūti entītijām, to identificējumi ir redzamie nosaukumi, tipi visos gadījumos ir funkcijas, resursi entītijām ir matemātiskā vide – šajā gadījumā Mathematica, entītiju nolūki jau ir minēti 3.3.1. nodaļā). Entītijas *BooleanFunctionPolynomialDegree* un *BooleanFunctionPolynomialModulus2* nav attēlotas grafā, jo tās nav pakārtotas citām entītijām, tāpat arī *BooleanFunctionCreate*, kas atkarīga no boolConvertToTruthValues.



3.1. attēls. **Entītiju atkarību grafs**

3.5. Saskarnes apraksts

Programmatūras prasību specifikācijas 2.3.2.1. nodaļā aprakstītās ārējās lietotāja saskarnes prasības projektēšanas gaitā ir pielāgotas sekojoši:

- atsevišķa izvēļņu josla netiek izdalīta;
- izvēļņu joslas iespēja ielasīt funkciju no faila pievienota ievades blokam – nolemts, ka specifiskajam pielietojumam šāds apvienojums būs uztveramāks un ērtāks (lietotājs vienmēr zinās kādā veidā ievadītu funkciju viņš pašreiz izmanto pretstatā gadījumam, kad caur izvēļņu joslu viņš ir ielasījis Būla funkciju no faila);
- izvēļņu joslas iespēja saglabāt funkciju un rezultātus failā pievienota rezultātu blokam – nolemts, jo lietotājs var izvēlēties saglabāt funkciju vai tās analīzes rezultātus tikai pēc tam, kad jau ir redzējis pašus rezultātus (atbilstoši, lietotājs rezultātus lasa no augšas uz leju un pēc tam, kad lietotājs ir pabeidzis lasīt konkrētās Būla funkcijas analīzes rezultātus, viņš jau zinās vai viņš vēlēsies saglabāt tikai funkciju vai arī visus rezultātus, ko viņš varēs izdarīt tieši pēc pašiem rezultātiem nepaceļot acis uz augšu).

Būla funkciju mēru noteikšanas funkcionalitāte tiks izsaukta tikai pēc atbilstošas aprēķināšanas pogas izsaukšanas.

Programmatūras prasību specifikācijas funkcionālajās prasībās norādītās prasības ir tādas pašas mēru noteikšanai, atbilstoši, lai neatkārtotos tās šeit nav vēlreiz aprakstītas (skat. Dekompozīcijas aprakstu).

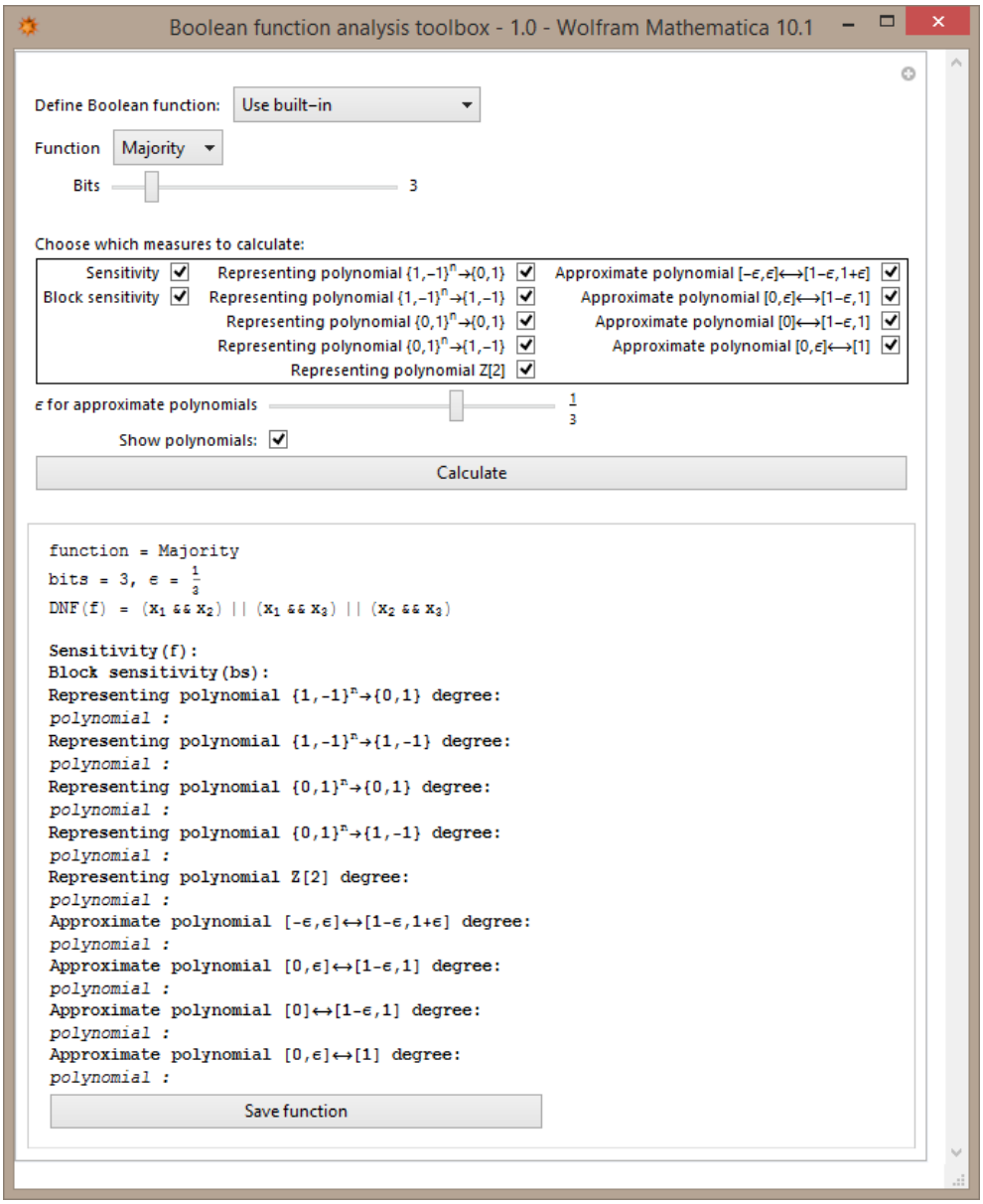
Standartā prasītais saskarnes atribūts ir norādīts dabiskā valodā.

3.5.1. Iebūvēto Būla funkciju saskarne

Būla funkciju analīzes lietotāja grafiskajā saskarnē iespējams izmantot iebūvētās Būla funkcijas un mainīt to parametrus (skat. 3.2. attēlu) atbilstoši Programmatūras prasību specifikācijas 2.3.2.1. nodaļā noteiktajam:

- lai izmantotu iebūvētās Būla funkcijas pie Būla funkciju definīcijas (angliski: *Define Boolean function*) nepieciešams izvēlēties lietot iebūvētās (angliski: *Use built-in*);
- lai izvēlētos konkrētu funkciju nepieciešams pie funkcijām (angliski: *Function*) izvēlēties vēlamo funkciju;
- lai izvēlētos bitu skaitu kādam Būla funkciju izmantot pie bitiem (angliski: *Bits*) nepieciešams slīdni novietot uz vēlamā bitu skaita;
- lai aprēķinātu tikai konkrētus raksturotājus sadaļā pie izvēlēties kādus raksturotājus aprēķināt (angliski: *Choose which measures to calculate*);

- lai izvēlētos kādu ticamību lietot tuvinātajiem polinomiem slīdni pie epsilon vērtības tuvinātajiem polinomiem (angliski: *ε for approximate polynomials*) jānovieto uz vēlamās kļūdas lieluma;
- lai izvēlētos vai rādīt polinomus nevis tikai to pakāpes pie polinomu rādīšanas (angliski: *Show polynomials*) jāatzīmē izvēlnes rūtiņa;
- lai aprēķinātu Būla funkcijas mērus pie uzstādītajiem parametriem nepieciešams
- noklikšķināt uz aprēķināt (angliski: *Calculate*);
- lai zinātu vai pašreiz notiek aprēķini pēc noklikšķēšanas uz aprēķināt tieši zem tā parādās josla, kas pārvietojas kamēr vien tiek veikti aprēķini;
- lai priekšlaicīgi apturētu aprēķinu gaitu (gadījumā ja izvēlēta nepareizā funkcija vai tās analīzes parametri, vai arī aprēķini lielāka bitu skaita gadījumā aizņem pārlieku lielu laika apjomu) nepieciešams noklikšķināt uz pārtraukt operācijas (angliski: *Abort operations*), kas parādās pēc noklikšķēšanas uz aprēķināt, bet pazūd, kad aprēķini pabeigti;
- lai saglabātu pašu Būla funkcijas definīciju loga apakšējā daļā nepieciešams noklikšķināt uz saglabāt funkciju (angliski: *Save function*); ja funkcija pie izvēlētās Būla funkcijas definīcijas veida vēl nav definēta, tad par to tiek izdots atbilstošs paziņojums.

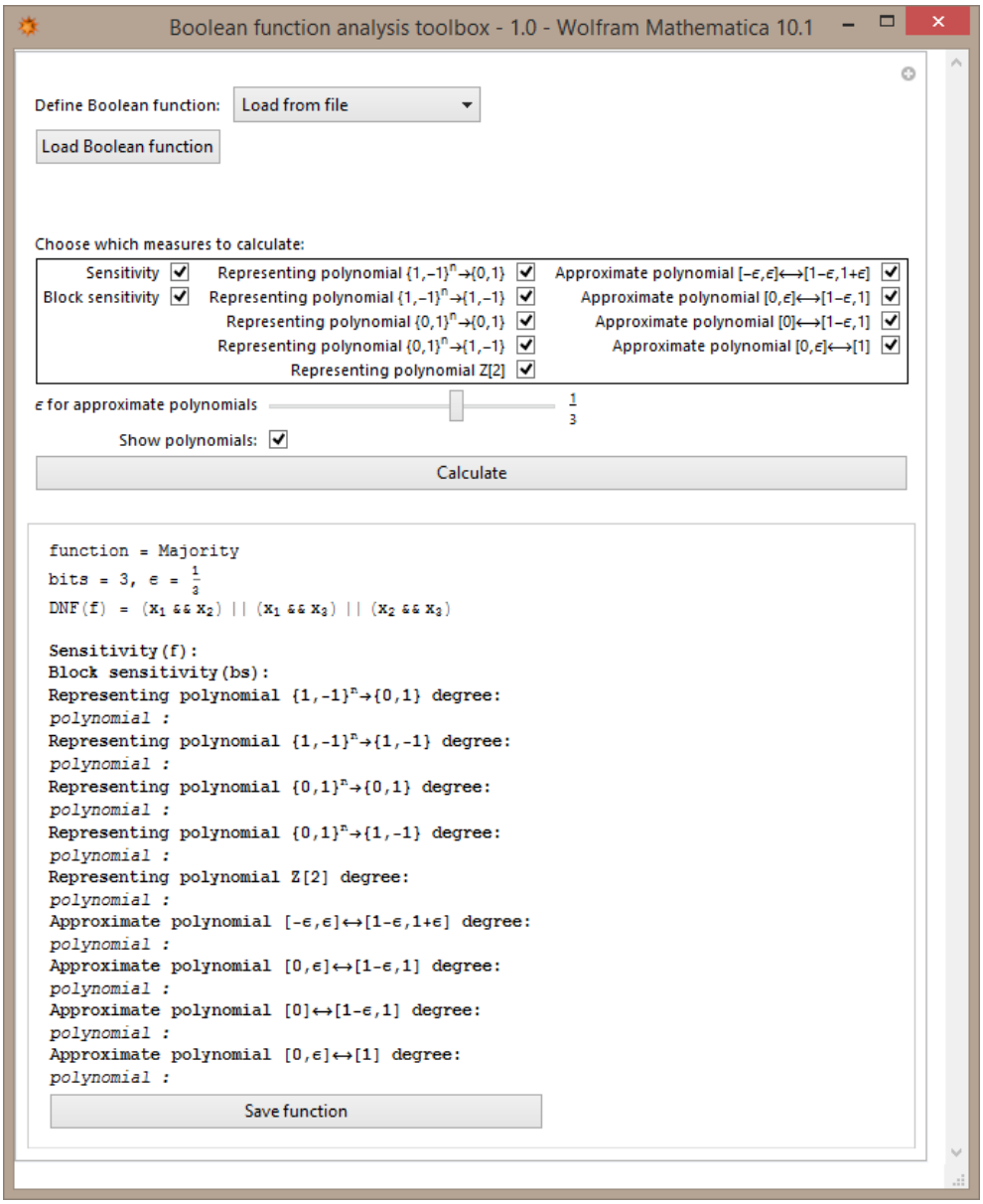


3.2. attēls. **Iebūvēto Būla funkciju izvēles saskarne**

3.5.2. BŪLA FUNKCIJU IEGUVES NO FAILA SASKARNE

Saskarnes daļā, kas paredzēta, lai ielasītu Būla funkciju no faila papildus 3.5.1. nodaļā noteiktajām iespējām neaprakstītās darbības ir sekojošas (skat. 3.3. attēlu):

- lai ielasītu Būla funkcijas no faila pie Būla funkciju definīcijas nepieciešams izvēlēties ielasīt no faila (angliski: *Load from file*);
- lai izvēlētos konkrētu funkciju nepieciešams noklikšķināt uz ielādēt Būla funkciju (angliski: *Load Boolean function*) un izlecošajā logā no failu sistēmas izvēlēties atbilstošo failu kurā ir Būla funkcijas definīcija; ja Būla funkcijas ielase ir bijusi veiksmīga, tad tas tiek atbilstoši norādīts, bet ja nav bijusi veiksmīga, tad par to tiek izdots atbilstošs paziņojums.

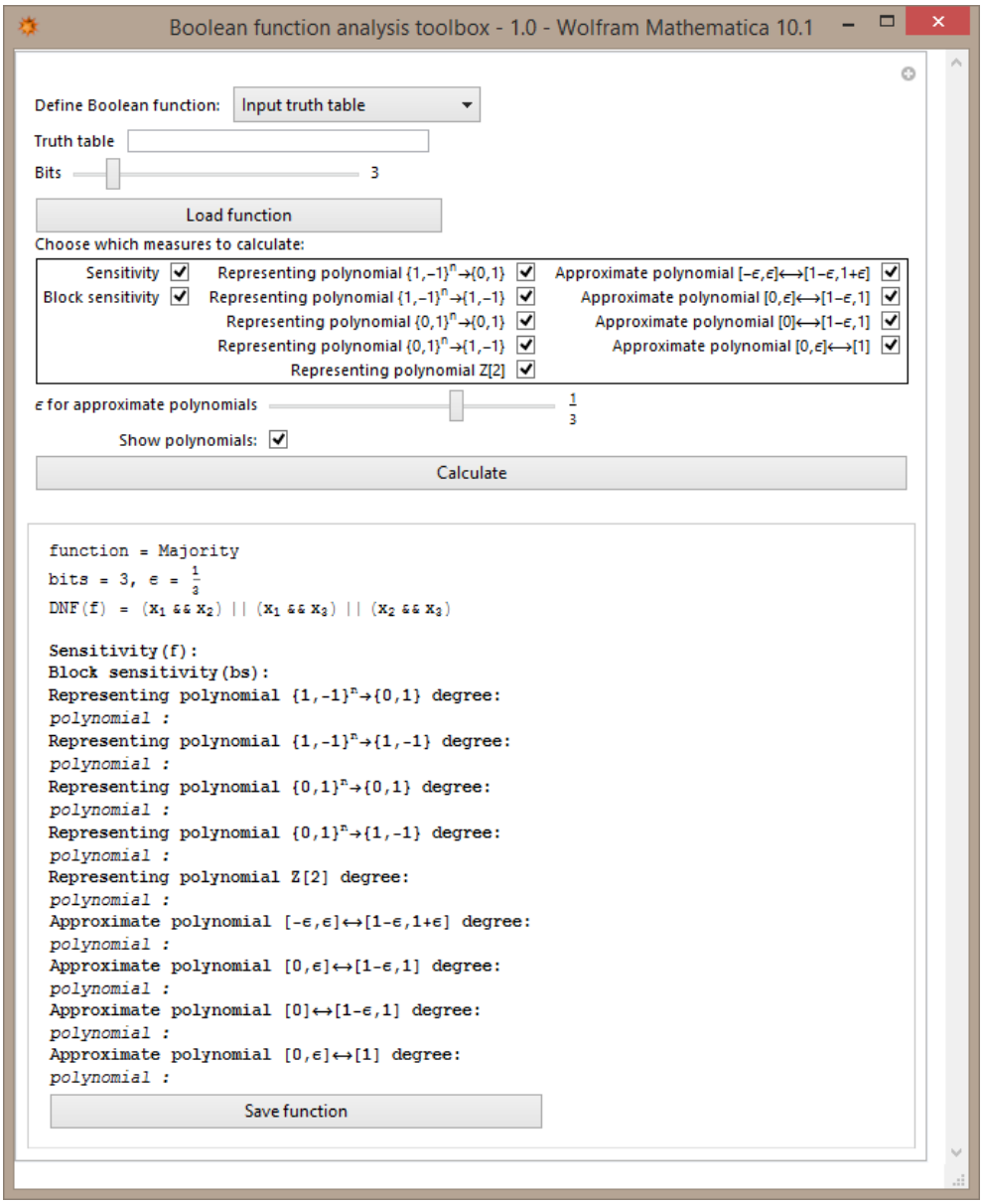


3.3. attēls. **Būla funkciju iegūšanas no faila saskarne**

3.5.3. BŪLA FUNKCIJU MANUĀLAS IEVADES SASKARNE

Saskarnes daļā, kas paredzēta, lai manuāli ievadītu Būla funkciju pašā saskarnē papildus 3.5.1. nodaļā noteiktajām iespējām neaprakstītās darbības ir sekojošas (skat. 3.4. attēlu):

- lai manuāli ievadītu Būla funkciju pie Būla funkciju definīcijas nepieciešams izvēlēties ievadīt patiesību tabulu (angliski: *Input truth table*);
- lai ievadītu konkrētu funkciju pie patiesību tabulas (angliski: *Truth table*) nepieciešams ievadīt $2^{bitu\ skaits}$ secīgas funkcijas atbildes 0,1, 1, −1 vai $True, False$ reprezentācijā, atdalītas ar komatiem;
- lai apstiprinātu ievadīto funkcijas definīciju nepieciešams noklikšķināt uz ielasīt funkciju (angliski: *Load function*); ja Būla funkcijas ielase ir bijusi veiksmīga, tad tas tiek atbilstoši norādīts, bet ja nav bijusi veiksmīga, tad par to tiek izdots atbilstošs paziņojums.

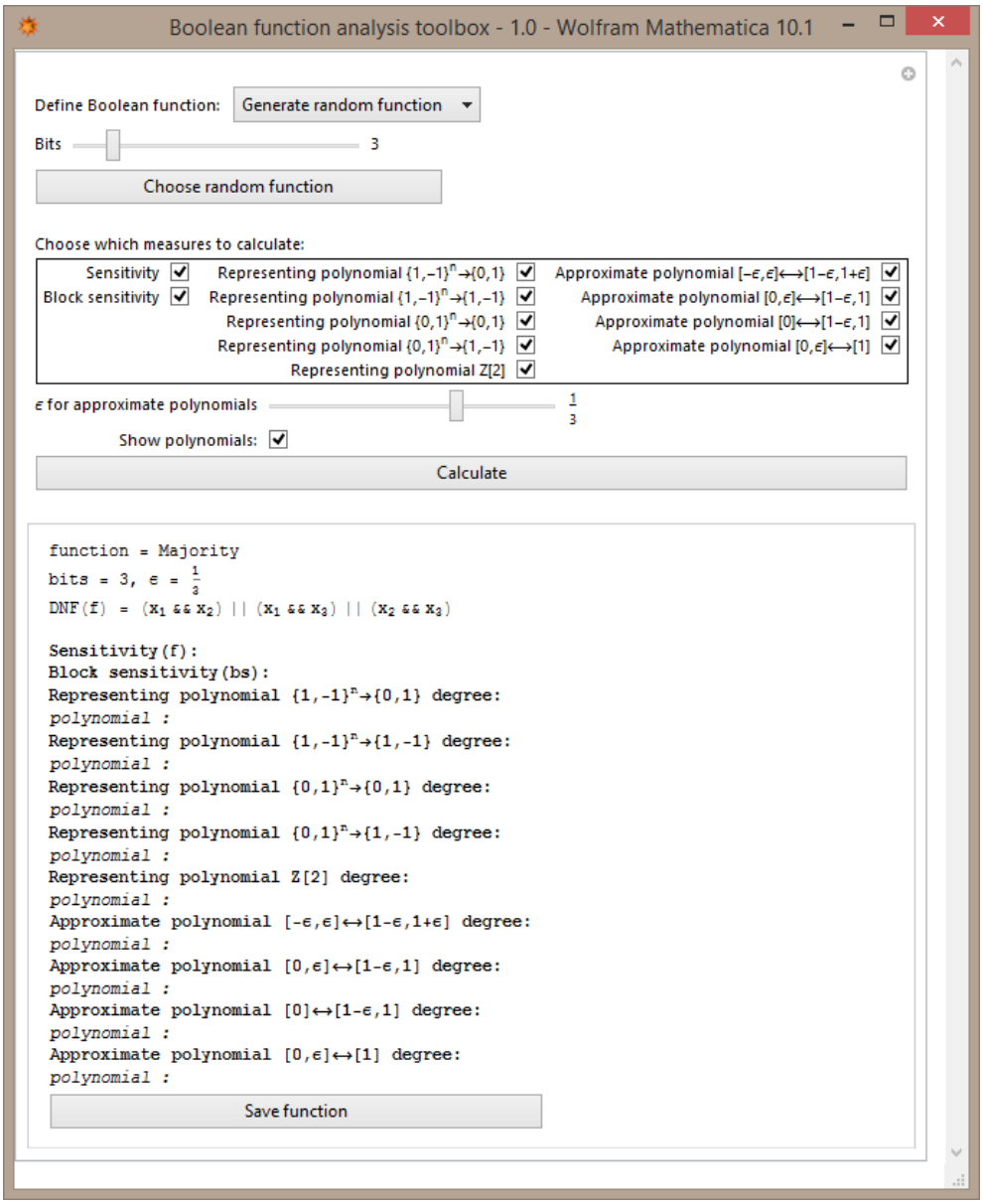


3.4. attēls. **Būla funkciju manuālas ievades saskarne**

3.5.4. BŪLA FUNKCIJU ĢENERĒŠANAS SASKARNE

Saskarnes daļā, kas paredzēta, lai ģenerētu nejaušas Būla funkcijas pie noteiktā bitu skaita papildus 3.5.1. nodaļā noteiktajām iespējām neaprakstītās darbības ir sekojošas (skat. 3.5. attēlu):

- lai uzģenerētu nejaušu Būla funkciju pie Būla funkciju definīcijas nepieciešams izvēlēties ģenerēt nejaušu funkciju (angliski: *Generate random function*);
- lai pie izvēlētā bitu skaita ģenerētu pašu funkciju nepieciešams noklikšķināt uz izvēlēties nejaušu funkciju (angliski: *Choose random function*).



3.5. attēls. **Nejaušu Būla funkciju ģenerēšanas saskarne**

3.6. Detalizētais projektējums

Detalizētais projektējums aptver matemātiskās vides paplašinājuma funkciju saskarnes un tās grafiskās saskarnes daļas, kas balstās uz minēto funkcionalitāti un veic mēru aprēķinus aprakstu.

3.6.1. Entītiju detalizētais projektējums

Par entītijas identificējumu tiek uzskatīts tās nosaukums. Entītijas nosaukums arī ir jāizmanto kā konkrētās funkcijas nosaukums.

Līdzīgi kā Programmatūras prasību specifikācijā, tad arī šeit, ja funkcijai ir iespējamas vairākas ievades, tad ar to jāsaprot, ka aprakstīti vairāki funkciju prototipi (skatoties no ievades viedokļa). Ja konkrētai ievadei ir norādītas dažādas reprezentācijas, tad arī šajā gadījumā tās jāuztver kā vairāku funkciju prototipus. Katrs prototips nav norādīts kā atsevišķa funkcija, jo tādā gadījumā informācija dublētos.

3.6.1.1. BooleanFunctionCreate

Ievade	Iespējamie ievadi: 1. Atbilžu vektors 0,1, 1, −1 vai $false, true$ reprezentācijā uzdots kā masīvs; ja vēlas izveidot funkciju n bitiem, tad masīvā jābūt 2^n elementiem (atbildēm); 2. Patiesību tabula 0,1, 1, −1 vai $false, true$ reprezentācijā uzdots kā piekārtojums; ja vēlas izveidot funkciju n bitiem, tad jābūt 2^n elementiem, kur katrs piekārtojums sastāv no masīva ar n ieejas bitiem, kam piekārtota atbilstošā funkcijas vērtība ieejas bitu reprezentācijā.
Apstrāde	Ja ievade nav patiesumvērtību reprezentācijā, tad to pārveido par tādu izmantojot tālāk definēto boolConvertToTruthValues. Lai izveidotu Būla funkciju izmanto BooleanFunction.
Izvade	Būla funkcija, kas saņemot n patiesumvērtību argumentus izdod ievadē definēto atbildi.

3.6.1.2. BooleanFunctionTable

Ievade	Būla funkcija, bitu skaits un reprezentācija (reprezentācija nav obligāta, tā var būt: 0,1, 1, −1 vai $false, true$; ja nav norādīts, tad tā ir 0,1).
Apstrāde	Izsauc funkciju boolGenerateBinaryStrings ar ievadē norādīto bitu skaitu patiesumvērtību reprezentācijā. Iegūtajā masīvā katru 1. līmeņa masīvu izmanto Būla funkcijā, lai noteiktu tās vērtību uz to. Visu iegūto Būla funkciju vērtības uz 1. līmeņa masīvu elementiem tālāk secīgi pārveido ievadē norādītajā reprezentācijā ar funkcijas boolConvertToIntegerValues palīdzību vai arī atstāj patiesumvērtību reprezentācijā.
Izvade	Masīvs ar ievadē noteikto bitu skaitu atbilžu, ievadē noteiktajā reprezentācijā.

3.6.1.3. BooleanFunctionSensitivity

Ievade	Būla funkcija un bitu skaits.
Apstrāde	Ar boolGenerateBinaryStrings palīdzību padodot ievadē norādīto bitu skaitu un patiesumvērtību reprezentāciju iegūst visus iespējamos ievadus. Tālāk pēc kārtas skatās katru ievadu. Katram apskatāmajam ievadam noskaidro ievadē padotās Būla funkcijas vērtību ar BooleanFunctionTable palīdzību, un tad pēc kārtas izveido konkrētā ievada pagaidu kopijas, katrai kopijai nomainot pēc kārtas citu ievades bitu uz pretējo. Kamēr pagaidu kopija eksistē skatās vai to padodot Būla funkcijai tās rezultāts atšķiras no rezultāta uz nemainītu ievadu. Visiem pagaidu ievadiem, kas veidoti no sākotnējā ievada, uz kuriem Būla funkcija dod savādāku rezultātu skaita cik reizes ievada maiņa ir atšķirusies no orģinālā; to atkārto katram pašā sākumā uzģenerētajam ievadam. Beigās atgriež maksimumu no visiem ievadiem reģistrētajiem mainīto bitu skaitiem uz kuriem funkcijas vērtība ir mainījusies no pašas ievades.
Izvade	Skaitlis, kas atspoguļo Būla funkcijas jūtīgumu.

3.6.1.4. BooleanFunctionBlockSensitivity

Ievade	Būla funkcija un bitu skaits.
Apstrāde	Izmantojot boolGenerateBinaryStrings, tai padodot ievadē norādīto bitu skaitu patiesumvērtību reprezentācijā, iegūst visus iespējamos ievadus, kurus tad tālāk apstrādās atsevišķi. Tāpat sagatavo nešķeļošo indeksu kopas kopu, kuras katras apakškopas elementi tiks izmantoti, lai pārbaudītu vai ievados tos mainot mainās Būla funkcijas vērtība. Tad pēc kārtas skatās katru ievadu. Apskatāmo ievadu padot Būla funkcijai un noskaidro tās vērtību, lai vēlāk to varētu salīdzināt izmantojot BooleanFunctionTable. Tālāk, apskatāmajam ievadam, pēc kārtas, pielietos katru nešķeļošos indeksu kopu. Tiek izveidoti pagaidu ievadi, kur katrā citā nešķeļošajai indeksu kopai piederīgajā blokā atbilstošās ievades vērtības tiek mainītas uz pretējām pēc kā skatās vai Būla funkcija mainītajam pagaidu ievadam izdod citu rezultātu. Tiek skaitīti visi gadījumi kuros konkrētās nešķeļošās indeksu kopas blokiem atbilstošo ievades bitu maiņa uz pretējiem mainījusi funkcijas vērtību konkrētajam ievadam. Skatās maksimālo šādu gadījumu skaitu konkrētam ievadam un beigās atgriež maksimālo šādu gadījumu skaitu no visiem ievadiem.
Izvade	Skaitlis, kas atspoguļo Būla funkcijas bloku jūtīgumu.

3.6.1.5. BooleanFunctionRepresentingPolynomial

Ievade	Būla funkcija, bitu skaits un polinoma iegūšanas veids (polinoma iegūšanas veids nav obligāts, tas var būt −10,−11, 0 vai −1; ja tas nav norādīts, tad tas ir 0). Vēlamais polinoma iegūšanas veids atspoguļo to kādā veidā polinoms veidots: - −10 gadījumā: $f\{1,-1\}^n \to \{0,1\}$,−11 gadījumā: $f\{1,-1\}^n \to \{1,-1\}$, 0 gadījumā: $f\{0,1\}^n \to \{0,1\}$ un −1 gadījumā: $f\{0,1\}^n \to \{1,-1\}$.

Apstrāde	Ar boolGenerateBinaryStrings palīdzību, tam padodot ievadē norādīto bitu skaitu un patiesumvērtības iegūst visus iespējamos ievadus. Tālāk ar Būla funkciju noskaidro funkcijas vērtību uz katru iesējamo ievadu izmantojot funkciju BooleanFunctionTable. Rezultātus pārveido atpakaļ ar boolConvertToIntegerValues, kā reprezentāciju izmantojot $1, -1$, ja polinoma iegūšanas veids ir norādīts -10 vai -11 un $1,0$, ja polinoma iegūšanas veids ir -1 vai 0. Tādā pašā veidā pārveido arī sākotnēji iegūtos iespējamos ievadus. Ja ir dota patvaļīga funkcija $f\{1,-1\}^n \to \{1,-1\}$, tad var izmantot zināmu metodi, kas interpolē 2^n vērtības, kuras f piešķir punktiem $\{1,-1\}^n \subset \mathbb{R}^n$. Katram punktam $a = (a_1, \cdots, a_n) \in \{1,-1\}^n$ rādītāj polinoms $1_{\{a\}}(x) = \left(\frac{1+a_1x_1}{2}\right) \cdots \left(\frac{1+a_nx_n}{2}\right)$ pieņem vērtību 1, kad $x = a$ un vērtību 0, kad $x \in \{1,-1\}^n \backslash \{a\}$. Atbilstoši f iegūst pārstāvošo polinomu $f(x) = \sum_{a\in\{1,-1\}^n} f(a)1_{\{a\}}(x)$. Nezināmos metodē izveido ar boolGenerateVariableArray. Ja polinoma iegūšanas veids ir -10 vai -11, tad šo metodi arī izmanto, vienīgi -10 gadījumā beigās veic pārveidojumu uz $0,1$ reprezentāciju ar BooleanFunctionPolynomialConvert. Ja polinoma iegūšanas veids ir -1 vai 0, tad metodi pielāgo tās iekšienē katram nezināmajam veicot aizvietošanu: $x_i = \frac{1-x_i}{2}$. Atbilstoši rezultātu arī izmanto, vienīgi -1 gadījumā beigās veic pārveidojumu uz $1, -1$ reprezentāciju ar BooleanFunctionPolynomialConvert.
Izvade	Polinoms.

3.6.1.6. BooleanFunction2ApproximatePolynomial

Ievade	Iespējamie ievadi: 1. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un reprezentācija (reprezentācija nav obligāta); 2. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums, vēlamā maksimālā pakāpe (pozitīvs skaitlis no 1) un reprezentācija (reprezentācija nav obligāta). Ja reprezentācija nav norādīta, tad tā ir $0,1$.
Apstrāde	Lai iegūtu rezultātu izsauc funkciju boolPolynomial ar sekojošiem ievadē norādītajiem argumentiem: Būla funkcija, bitu skaits, pieļaujamās kļūdas vērtība, pieļaujamās kļūdas vērtība, pieļaujamās kļūdas vērtība, pieļaujamās kļūdas vērtība, reprezentācija un vēlamā maksimālā pakāpe (ja tāda ir norādīta).
Izvade	Atgriež tuvinātu polinomu ar pēc iespējas mazāku pakāpi. Ja norādīta maksimālā pakāpe un tai nav iespējams iegūt atrisinājumu, tad par to lietotāju atbilstoši informē.

3.6.1.7. BooleanFunction2InApproximatePolynomial

Ievade	Iespējamie ievadi: 1. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un reprezentācija (reprezentācija nav obligāta); 2. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums, vēlamā maksimālā pakāpe (pozitīvs skaitlis no 1) un reprezentācija (reprezentācija nav obligāta). Ja reprezentācija nav norādīta, tad tā ir 0,1.
Apstrāde	Lai iegūtu rezultātu izsauc funkciju boolPolynomial ar sekojošiem ievadē norādītajiem argumentiem: Būla funkcija, bitu skaits, 0, pieļaujamās kļūdas vērtība, pieļaujamās kļūdas vērtība, 0, reprezentācija un vēlamā maksimālā pakāpe (ja tāda ir norādīta).
Izvade	Atgriež tuvinātu polinomu ar pēc iespējas mazāku pakāpi. Ja norādīta maksimālā pakāpe un tai nav iespējams iegūt atrisinājumu, tad par to lietotāju atbilstoši informē.

3.6.1.8. BooleanFunction11ApproximatePolynomial

Ievade	Iespējamie ievadi: 1. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un reprezentācija (reprezentācija nav obligāta); 2. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums, vēlamā maksimālā pakāpe (pozitīvs skaitlis no 1) un reprezentācija (reprezentācija nav obligāta). Ja reprezentācija nav norādīta, tad tā ir 0,1.
Apstrāde	Lai iegūtu rezultātu izsauc funkciju boolPolynomial ar sekojošiem ievadē norādītajiem argumentiem: Būla funkcija, bitu skaits, 0, 0, pieļaujamās kļūdas vērtība, 0, reprezentācija un vēlamā maksimālā pakāpe (ja tāda ir norādīta).
Izvade	Atgriež tuvinātu polinomu ar pēc iespējas mazāku pakāpi. Ja norādīta maksimālā pakāpe un tai nav iespējams iegūt atrisinājumu, tad par to lietotāju atbilstoši informē.

3.6.1.9. BooleanFunction10ApproximatePolynomial

Ievade	Iespējamie ievadi: 1. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums un reprezentācija (reprezentācija nav obligāta); 2. Būla funkcija, bitu skaits, pieļaujamās kļūdas lielums, vēlamā maksimālā pakāpe (pozitīvs skaitlis no 1) un reprezentācija (reprezentācija nav obligāta). Ja reprezentācija nav norādīta, tad tā ir 0,1.
Apstrāde	Lai iegūtu rezultātu izsauc funkciju boolPolynomial ar sekojošiem ievadē norādītajiem argumentiem: Būla funkcija, bitu skaits, 0, pieļaujamās kļūdas vērtība, 0, 0, reprezentācija un vēlamā maksimālā pakāpe (ja tāda ir norādīta).
Izvade	Atgriež tuvinātu polinomu ar pēc iespējas mazāku pakāpi. Ja norādīta maksimālā pakāpe un tai nav iespējams iegūt atrisinājumu, tad par to lietotāju atbilstoši informē.

3.6.1.10. BooleanFunctionPolynomialModulus2

Ievade	Polinoms.
Apstrāde	Izmantojot funkciju PolynomialMod reducē ievadē padotu polinomu pēc moduļa 2.
Izvade	Pārveidotais polinoms.

3.6.1.11. BooleanFunctionPolynomialConvert

Ievade	Polinoms un vēlamā reprezentācija (reprezentācija nav obligāta, tā var būt: $0,1$ vai $1,-1$; ja nav norādīts, tad tā ir $1,-1$). Polinoms kā nezināmos izmanto x_i.
Apstrāde	Katru nezināmo aizstāj: ja nepieciešama $0,1$ reprezentācija, tad $x_i = \frac{1-x_i}{2}$, bet ja nepieciešama $1,-1$ reprezentācija, tad $x_i = 1 - 2x_i$.
Izvade	Pārveidotais polinoms.

3.6.1.12. BooleanFunctionPolynomialDegree

Ievade	Polinoms.
Apstrāde	Skatās cik monomu polinomā ir. Katram monomam skatās kādi nezināmie tajā ir un kādas ir viņu pakāpes. Atgriež maksimālo pakāpi starp monomiem.
Izvade	Skaitlis, kas atspoguļo polinoma pakāpi.

3.6.1.13. boolConvertToIntegerValues

Ievade	Masīvs ar patiesumvērtībām un vēlamā reprezentācija (reprezentācija nav obligāta, tā var būt: $0,1$ vai $1,-1$; ja nav norādīts, tad tā ir $0,1$).
Apstrāde	Veic patiesumvērtību aizvietošanu uz vēlamo reprezentāciju.
Izvade	Masīvs kam vērtības dotas vēlamajā reprezentācijā.

3.6.1.14. boolConvertToTruthValues

Ievade	Masīvs ar skaitliskām vērtībām (reprezentācija var būt: $0,1$ vai $1,-1$).
Apstrāde	Veic skaitlisko vērtību aizstāšanu ar patiesumvērtībām.
Izvade	Masīvs kam vērtības dotas patiesumvērtību reprezentācijā.

3.6.1.15. boolGenerateBinaryStrings

Ievade	Elementu daudzums un reprezentācija (reprezentācija var būt: $0,1$, $1,-1$ vai $false, true$).
Apstrāde	Izmantojot funkciju Tuples sagatavo visus iespējamos ievadē norādītā elementu daudzuma garuma kortežus.
Izvade	Masīvs ar kortežiem.

3.6.1.16. boolGenerateMultivariatePolynomial

Ievade	Elementu daudzums.

Apstrāde	Sagatavo nezināmos mainīgos ar boolGenerateVariableArray palīdzību, kam padod elementu daudzumu. Sagatavo arī koeficientu mainīgos. Tālāk izveido iespējamos monomus un katram pievieno savu koeficientu.
Izvade	Multilineārs polinoms ar koeficientiem.

3.6.1.17. boolGenerateVariableArray

Ievade	Elementu daudzums.
Apstrāde	Sagatavo masīvu ar mainīgajiem x_i ievadē norādītajā daudzumā.
Izvade	Masīvs ar mainīgajiem.

3.6.1.18. boolPolynomial

Ievade	Būla funkcija, bitu skaits, pieļaujamās kļūdas vērtība līdz nepatiesajai atbildei, kļūdas vērtība no nepatiesās atbildes, kļūdas vērtība līdz patiesajai atbildei, kļūdas vērtība no patiesās atbildes, reprezentācija un vēlamā maksimālā pakāpe (vēlamā maksimālā pakāpe nav obligāta).
Apstrāde	Ar boolGenerateBinaryStrings palīdzību sagatavo iespējamos ievadus pie dotā bitu skaita norādītajā reprezentācijā. Ar BooleanFunctionTable noskaidro atbildes dotajiem ievadiem. Ar boolGenerateMultivariatePolynomial palīdzību sagatavo multilineāru polinomu aprēķinu veikšanai. Tad sagatavo vienādojumus lineārajai programmai un ar Solve palīdzību tos mēģina atrisināt sākot ar polinomiem ar mazākajām monomu pakāpēm. Ja tādu nevar atrisināt pakāpi palielina, līdz atrod (vienmēr būs atrisinājums, jo ir pārstāvošais polinoms, tad jāizmanto BooleanFunctionRepresentingPolynomial, kas veiks aprēķinus ātrāk nekā lineārā programma). Ja norādīta vēlamā maksimālā pakāpe, tad mēģina atrisināt līdz tai. Ja atrisinājuma nav tad par to izdod atbilstošu paziņojumu.
Izvade	Polinoms ar ievadei atbilstošiem parametriem.

3.7. Prasību trasējamības matrica

Programmatūras prasību specifikācijā noteiktās funkcionālās un nefunkcionālās prasības un to realizācija redzama tabulā (skat. 3.2. tabulu). Lai redzētu realizēto prasību atkarību no Programmatūras prasību specifikācijā ieviestajām papildus funkcijām skatīties atkarību grafu (skat. 3.1. attēlu).

3.2. tabula. **Prasības un to realizācija**

Prasība	**Realizācija**
2.3.1.1.1. Būla funkcijas uzdošana	BooleanFunctionCreate
2.3.1.1.2. Būla funkcijas atbilžu saraksta ģenerēšana	BooleanFunctionTable
2.3.1.2.1. Būla funkcijas jūtīguma aprēķināšana	BooleanFunctionSensitivity
2.3.1.2.2. Būla funkcijas bloku jūtīguma aprēķināšana	BooleanFunctionBlockSensitivity
2.3.1.2.3. Būla funkciju pārstāvošā polinoma ģenerēšana	BooleanFunctionRepresentingPolynomial
2.3.1.2.4. Tuvināta polinoma aprēķināšana ar 2-pusēju kļūdu	BooleanFunction2ApproximatePolynomial
2.3.1.2.5. Tuvināta polinoma aprēķināšana ar 2-pusēju kļūdu iekšpusē	BooleanFunction2InApproximatePolynomial
2.3.1.2.6. Tuvināta polinoma aprēķināšana ar 1-pusēju kļūdu 1-pusē	BooleanFunction11ApproximatePolynomial
2.3.1.2.7. Tuvināta polinoma aprēķināšana ar 1-pusēju kļūdu 0-pusē	BooleanFunction10ApproximatePolynomial
2.3.1.2.8. Reducēt polinomu pēc moduļa 2	BooleanFunctionPolynomialModulus2
2.3.1.3.1. Polinomu pārveide starp reprezentācijām	BooleanFunctionPolynomialConvert
2.3.1.3.2. Polinoma pakāpes noteikšana	BooleanFunctionPolynomialDegree
2.3.2.1. Grafiskā lietotāja saskarne	skat. 3.5. Saskarnes apraksts
2.3.2.2. Programmatūras saskarne	Tiks nodrošināts programmprodukta kodēšanas laikā pamatojoties uz pieejamo dokumentāciju.
2.3.3. Veiktspējas prasības	Tiks nodrošināts programmprodukta kodēšanas un testēšanas laikā.
2.3.4. Aparatūras ierobežojumi	Tiks nodrošināts programmprodukta kodēšanas un testēšanas laikā.

4. Testēšanas dokumentācija

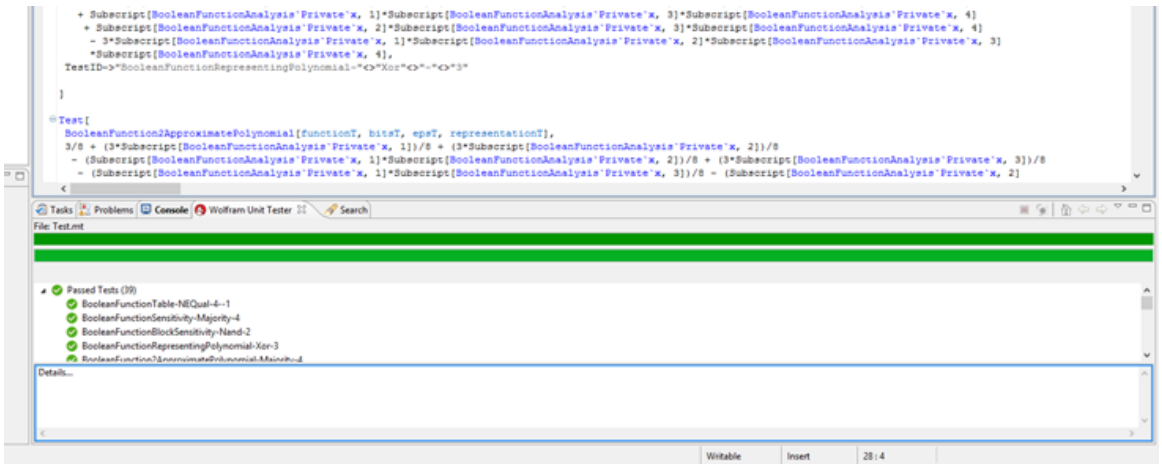
4.1. Ievads

Šajā dokumentā ir aprakstīta Atbalsta bibliotēkas Būla funkciju izpētei vienībtestēšanas process un tā galvenie rezultāti. Darbā tika veikta arī funkcionālā testēšana.

Šeit testēšana tiek veikta tikai uz iespējamām padodamām vērtībām, jo tiek izmantota simboliskā programmatūras saskarne. Bet lietotāja grafiskajā saskarnē kļūdu pārbaudes un ziņojumi jau ir iekļauti. Lietotāja grafiskā saskarne paļaujas uz vienām un tām pašām funkcijām, tādēļ to pareizība šeit tiek pārbaudīta. Pašas saskarnes funkcionālā testēšana bija ļoti vienkārša. Tika atrasta kļūda, kas aptur programmu, ja funkcijas saglabāšanas logā nospiež Atcelt – tomēr kā tika noskaidrots, tad tā ir Mathematica plaši zināma kļūda. Bibliotēkas iekšējās funkcijas tika pārbaudītas manuāli. Tas pats attiecas arī uz iekšējo funkciju izsaukumiem un to korektumu, kas pārbaudīts vizuāli. Tuvināto polinomu funkcionalitāte nav tikusi testēta, jo vispārīgajā gadījumā ar roku var pārbaudīt tikai jau esošu rezultātu, bet jaunu izrēķināt ir grūti. Atsevišķām funkcijām testēšanu iebūvēt nemaz nevar (piemēram, funkcija BooleanFunctionPolynomialConvert nevar zināt kādas reprezentācijas polinoms viņai ir padots, to var tikai pieņemt no otrā argumenta u.tml.).

Ir veikts 100% testu pārklājums visām programmatūras saskarnes funkcijām, taču tikai ar maz bitu virknēm, jo vairāk ir grūti ar roku pārrēķināt (te domāts, ka izieti visi ceļi šajās funkcijās, taču nav pārbaudītas citas iekšēji pakārtotas funkcijas).

Vienībtestēšanas laikā kļūdas netika atrastas (skat. 4.1. attēlu), jo visas līdzšinējās tika atrastas vienkārši pārskatot kodu. Iespējams, vienībtestēšanas sagatavošana būtu atvieglojusi kļūdu izķeršanu pašā sākumā (kaut gan funkcijas un to prototipi mainījās), tomēr šajā projektā, diemžēl, vienībtestēšana tika veikta tikai gala produktam.



4.1. attēls. **Būla funkciju vienībtestēšanas rezultāti**

4.2. BooleanFunctionCreate

Ievads	Sagaidāmais rezultāts	Rezultāts
{0,1,0,1,1,1,0,1}	3 bitu Būla funkcija	+
{1,1,0,1}	2 bitu Būla Funkcija	+
$\{\{0,0\} \to 1, \{1,1\} \to 1, \{1,0\} \to 0, \{0,1\} \to 1\}$	2 bitu Būla Funkcija	+

4.3. BooleanFunctionTable

Ievads	Sagaidāmais rezultāts	Rezultāts
3,-1	$\{\{1,1,1\},\{1,1,-1\},\{1,-1,1\},\{1,-1,-1\},$ $\{-1,1,1\},\{-1,1,-1\},\{-1,-1,1\},\{-1,-1,-1\}\}$	+
2,True	{{False, False}, {False, True}, {True, False}, {True, True}}	+
2,0	{{0,0}, {0,1}, {1,0}, {1,1}}	+

4.4. BooleanFunctionSensitivity

Ievads	Sagaidāmais rezultāts	Rezultāts
Xor,2	2	+
Majority, 3	3	+
And, 3	3	+

4.5. BooleanFunctionBlockSensitivity

Ievads	Sagaidāmais rezultāts	Rezultāts
Xnor,2	3	+
Nand,2	2	+
Majority,4	3	+

4.6. BooleanFunctionRepresentingPolynomial

Ievads	Sagaidāmais rezultāts	Rezultāts
Majority,4,-11	$\frac{3}{8}+\frac{3x_1}{8}+\frac{3x_2}{8}-\frac{x_1x_2}{8}+\frac{3x_3}{8}-\frac{x_1x_3}{8}-\frac{x_2x_3}{8}-\frac{1}{8}x_1x_2x_3+\frac{3x_4}{8}-\frac{x_1x_4}{8}-\frac{x_2x_4}{8}-\frac{1}{8}x_1x_2x_4-\frac{x_3x_4}{8}-\frac{1}{8}x_1x_3x_4-\frac{1}{8}x_2x_3x_4+\frac{3}{8}x_1x_2x_3x_4$	+
Or,3,-11	$-\frac{3}{4}+\frac{x_1}{4}+\frac{x_2}{4}+\frac{x_1x_2}{4}+\frac{x_3}{4}+\frac{x_1x_3}{4}+\frac{x_2x_3}{4}+\frac{1}{4}x_1x_2x_3$	+
Majority,3,-11	$\frac{x_1}{2}+\frac{x_2}{2}+\frac{x_3}{2}-\frac{1}{2}x_1x_2x_3$	+

4.7. BooleanFunctionPolynomialModulus2

Ievads	Sagaidāmais rezultāts	Rezultāts
$x_1+x_2-2x_1x_2+x_3-2x_1x_3-2x_2x_3+4x_1x_2x_3$	$x_1+x_2+x_3$	+
$1-x_1-x_2+2x_1x_2-x_3+2x_1x_3+2x_2x_3-4x_1x_2x_3$	$1+x_1+x_2+x_3$	+
$1-x_1-x_2+2x_1x_2-x_3+2x_1x_3+2x_2x_3-4x_1x_2x_3-x_4+2x_1x_4+2x_2x_4-4x_1x_2x_4+2x_3x_4-4x_1x_3x_4-4x_2x_3x_4+8x_1x_2x_3x_4$	$1+x_1+x_2+x_3+x_4$	+

4.8. BooleanFunctionPolynomialConvert

Ievads	Sagaidāmais rezultāts	Rezultāts
$1-x_1-x_2+2x_1x_2$,-1	$-1+2x_1+2(1-2x_1)(1-2x_2)+2x_2$	+
x_1x_2,-1	$(1-2x_1)(1-2x_2)$	+
$\frac{1}{2}-\frac{x_1}{2}-\frac{x_2x_3}{2}-\frac{1}{2}x_1x_2x_3$,0	$\frac{1}{2}+\frac{1}{4}(-1+x_1)-\frac{1}{8}(1-x_2)(1-x_3)-\frac{1}{16}(1-x_1)(1-x_2)(1-x_3)$	+

4.9. BooleanFunctionPolynomialDegree

Ievads	Sagaidāmais rezultāts	Rezultāts
$1+x_1+x_2+x_1x_2+x_3+x_2x_3$	2	+
$\frac{1}{2}-\frac{x_2}{2}-\frac{x_1x_3}{2}-\frac{1}{2}x_1x_2x_3$	3	+
$1-x_2+x_1x_2$	2	+

4.10. Veiktspējas testēšana

Pārbaudīts, ka visas funkcijas uz 7 bitiem, atsevišķi, izpildās laikā līdz 5 minūtēm, kas noteikts Programmatūras prasību specifikācijā. Darbības ātrums samazinās eksponenciāli, kā jau bija sagaidāms, jo iespējamo funkciju skaits kāpj eksponenciāli, kāpjot bitu skaitam.

5. PROGRAMMATŪRAS PIRMKODA FRAGMENTI

```
If[Not@ValueQ[BooleanFunction2InApproximatePolynomial::usage],
BooleanFunction2InApproximatePolynomial::usage = "BooleanFunction2InApproximatePolynomial[function,bits,error] will
                                find polynomial that will provide the same answer the original function does. As this polyno-
                                mial will always give results in between 0 and 1 it can be easily used to show complexity and
                                correctness probability of algorithms or functions."];

boolConvertToTruthValues[integerArray_]:=
    If[MemberQ[integerArray,-1,2],
(* If this would not be internal function than levelspec 2 should not be used as a result of what function should be overlooked.
                                *)
        integerArray/.{_?(#==-1&)->True,_?(#==1&)->False},
        integerArray/.{_?(#==1&)->True,_?(#==0&)->False}]

boolPolynomial::maxDegree = "Approximate polynomial with degree smaller or equal to the one demanded was not found!"
boolPolynomial[booleanFunction_,bits_,errorFalse_,falseError_,errorTrue_,trueError_,representation_,maxDe-
                                gree_:bits_,over_:Reals]:=
    Module[{
        inputs=boolGenerateBinaryStrings[bits,representation],
        variables=boolGenerateVariableArray[bits,x],
        answers=BooleanFunctionTable[booleanFunction,bits,representation],
        polynomial=boolGenerateMultivariatePolynomial[bits],
        inequalitySystem},
        (* Initialization (code reccurance) with starting values needed for inequalitySystem otherwise Linear program wont
                                recognize and solve it for it to be initialized other way. *)
        If[representation===-1,
            If[answers[[1]]===1,
                inequalitySystem=(answers[[1]]-errorFalse<=(polynomial/.Table[variables[[j]]-
                                >inputs[[1]][[j]],{j,bits}])<=answers[[1]]+falseError),
                inequalitySystem=(answers[[1]]-errorTrue<=(polynomial/.Table[variables[[j]]-
                                >inputs[[1]][[j]],{j,bits}])<=answers[[1]]+trueError)],
            If[answers[[1]]===0,
                inequalitySystem=(answers[[1]]-errorFalse<=(polynomial/.Table[variables[[j]]-
                                >inputs[[1]][[j]],{j,bits}])<=answers[[1]]+falseError),
                inequalitySystem=(answers[[1]]-errorTrue<=(polynomial/.Table[variables[[j]]-
                                >inputs[[1]][[j]],{j,bits}])<=answers[[1]]+trueError)]];
        Do[
            inequalitySystem=
                inequalitySystem&&If[representation===-1,
                    If[answers[[i]]===1,
                        (answers[[i]]-errorFalse<=(polynomial/.Table[variables[[j]]-
                                >inputs[[i]][[j]],{j,bits}])<=answers[[i]]+falseError),
                        (answers[[i]]-errorTrue<=(polynomial/.Table[variables[[j]]-
                                >inputs[[i]][[j]],{j,bits}])<=answers[[i]]+trueError)],
                    If[answers[[i]]===0,
                        (answers[[i]]-errorFalse<=(polynomial/.Table[variables[[j]]-
                                >inputs[[i]][[j]],{j,bits}])<=answers[[i]]+falseError),
                        (answers[[i]]-errorTrue<=(polynomial/.Table[variables[[j]]-
                                >inputs[[i]][[j]],{j,bits}])<=answers[[i]]+trueError)]],
            {i,2,2^bits,1}];
        Module[{
            degreeMembers=Table[Binomial[bits,k],{k,0,bits}],
            result={},
            triggerMax=False},
            Do[
                result=FindInstance[
                    inequalitySystem/.Table[Subscript[a, j]-> 0,{j,Sum[degreeMembers[[i]],{i,k}]+1,2^bits}],
```

```
                    boolGenerateVariableArray[Sum[degreeMembers[[i]],{i,k}],a],
                    over];
                If[Length[result]!=0,
                    polynomial=polynomial/.Table[Subscript[a, j]-> 0,{j,Sum[degreeMembers[[i]],{i,k}]+1,2^bits}];
                    Break[]];
                If[k>=maxDegree+1,
                    triggerMax=True;
                    Break[]],
                {k, 1,Length[degreeMembers]-1}];
            If[Length[result]!=0,
                polynomial/.Flatten[result],
                If[triggerMax,
                    Message[boolPolynomial::maxDegree],
                    If[representation==-1,
                        BooleanFunctionRepresentingPolynomial[booleanFunction,bits,-11],
                        BooleanFunctionRepresentingPolynomial[booleanFunction,bits,0]]]]]]

BooleanFunctionRepresentingPolynomial[booleanFunction_,bits_,whatWay_:0]/;
    If[FunctionQ[booleanFunction],*)
        (*If[TrueQ[whatWay==-10||whatWay==-11||whatWay==0||whatWay==-1],
            If[TrueQ[bits>=1]&&Positive[bits]&&IntegerQ[bits],
                True,
                Message[BooleanFunctionRepresentingPolynomial::arg3err,whatWay];False],
            Message[BooleanFunctionRepresentingPolynomial::arg2err,bits];False],
        Message[BooleanFunctionRepresentingPolynomial::arg1err,booleanFunction];False]:=
(* whatWay shows in which way to create polynomial: -10 means f{1,-1}^n\[RightArrow]{0,1},-11 means f{1,-1}^n\[RightAr-
                            row]{1,-1}, 0 means f{0,1}^n\[RightArrow]{0,1} and -1 means f{0,1}^n\[RightArrow]{1,-1}. *)
    Module[{
        representation},
        If[whatWay==-10||whatWay==-11,
            representation=-1,
            representation=0];
        Module[{
            inputs=boolGenerateBinaryStrings[bits,representation],
            answers=BooleanFunctionTable[booleanFunction,bits,representation],
            variables=boolGenerateVariableArray[bits,x],
            polynomial},
            If[representation==-1,
                polynomial=
                    Expand[
                        Sum[answers[[i]]*Product[(1+inputs[[i]][[j]]*variables[[j]])/2,
                        {j,1,bits}],
                        {i,1,2^bits}]],
                Module[{
                    polynomialHelper=boolGenerateMultivariatePolynomial[bits],
                    inequalitySystem},
                    inequalitySystem=((polynomialHelper/.Table[variables[[j]]->inputs[[1]][[j]],{j,bits}])==answers[[1]]);
                    Do[
                        inequalitySystem=inequalitySystem&&((polynomialHelper/.Table[variables[[j]]-
                                >inputs[[i]][[j]],{j,bits}])==answers[[i]]),
                        {i,2,2^bits,1}];
                    polynomial=polynomialHelper/.Flatten[FindInstance[inequalitySystem,boolGenerateVariableAr-
                                ray[2^bits,a],Integers]]]];
            If[whatWay==-10||whatWay==-1,
                If[whatWay==-10,
                    BooleanFunctionPolynomialConvert[polynomial,0],
                    BooleanFunctionPolynomialConvert[polynomial,-1]],
                polynomial]]]
```

6. Projekta organizācija

Projekts tika izstrādāts pēc spirāles dzīvescikla modeļa. To noteica tas, ka uzsākot projektu nebija apgūts nepieciešamais teorētiskais materiāls un izstrādes vide, kā arī nebija pilnībā zināms tas cik tieši un kuras no vēlamajam prasībām nepieciešamas (ar nepieciešamību, šeit, tiek saprasts, tas cik ļoti kāda funkcionalitāte ir vajadzīgāka pret kādu citu izstrādei atvēlētajā laikā).

Projektu izstrādāja viens cilvēks, veicot prasību specificēšanu, to projektēšanu, izstrādi un arī testēšanu. Testēšanā pastarpināti piedalījās speciālists no malas ar nepieciešamajām teorētiskajām zināšanām par Būla funkcijām.

Sākumā autors iepazinās ar teorētiskajiem materiāliem un tika izvirzītas galvenās prasības. Sākumā tika izveidots pirmais prototips. Balstoties uz to tika gūts sākotnējais priekšstats par prasībām un izstrādāta Programmatūras prasību specifikācija. Tālāk tika izstrādāts nākošais prototips, jau ņemot vērā jauniegūtās teorētiskās un izstrādes vides zināšanas. Tad tika konkretizētas prasības un atbilstoši pielāgota Programmatūras prasību specifikācija uz kā pamata tālāk tika izstrādāts Programmatūras projektējuma apraksts. Tika izveidots vēl viens prototips pēc kā tika precizēts projektējuma apraksts un sagatavots detalizētais projektējums. Pēc kā jau tika sagatavota produkta gala versija, kas arī tika testēta.

7. Kvalitātes nodrošināšana

Lai izstrādātais programmprodukts nodrošinātu lietotājiem nepieciešamo funkcionalitāti un būtu kvalitatīvi tika veiktas šādas darbības:

- izejot no vēlamajām prasībām tika noteikta nepieciešamā funkcionalitāte un tās papildināšanas iespējas nepieciešamās ieviešanas gadījumā;
- ar pasūtītāju tika aptuveni atrunātas komunikācijas detaļas un ļoti vispārīgi ieskicēta darba izstrāde;
- balstoties uz standartiem un komunikāciju ar pasūtītāju tika izstrādāta programmatūras prasību specifikācija, testēšanas dokumentācija un vēlāk arī programmatūras projektējuma apraksts;
- tika uzsākta programmatūras koda izstrāde, ņemot vērā izstrādes vides dokumentāciju un programmēšanas labo praksi; kods tika izstrādāts angļu valodā, norādot nolūku katrai metodei; jau izstrādes laikā manuāli tika pārbaudīta metožu daļu pareizība un novērstas nepilnības;
- cikliski produkta versijas tika atrādītas pasūtītājam, lai noskaidrotu iespējamos produkta attīstības virzienus;
- cikliski tika pārskatīta izstrādātā daļa, tās pareizība un uzlabošanas iespējas;
- programmatūras saskarnes daļai tika veikta arī vienībtestēšana, lai vēlreiz pārliecinātos par tās pareizību.

8. Konfigurācijas pārvaldība

Lai nodrošinātu kvalitātes prasības tika veikta versiju kontrole. Sakarā ar to, ka salīdzinošais izmaiņu daudzums un izkliede izstrādājamajā programmproduktā bija paredzami mazāka nekā tas būtu citu projektu gadījumā, tad no sākuma tika nolemts konfigurāciju pārvaldību nodrošināt manuāli. Izstrādes gaitā programmprodukts tika iedalīts versijās pēc funkcionālas paplašināšanās vai arī pirms sarežģītākas ieviešanas. Versiju kontrole programmatūras kodam tika nodrošināta manuāli veicot tā kopijas failu sistēmā. Paralēli, nedēļā reizi, notika versiju glabāšana mākonī, drošības nolūkos. Katrai jaunai versijai tika norādītas veiktās izmaiņas pret iepriekšējo. Dublēšana tika veikta arī dokumentācijai, taču tās versijas netika izdalītas pamatojoties uz to, ka prasības tika izstrādātas jau sākumā un projekta specifiku, bet katru reizi, kad tika veiktas izmaiņas projektējumā, trasējamības nolūkā, tās tika veiktas pilnībā, ieskaitot programmatūras koda pielāgošanu.

9. Darbietilpības novērtējums

Uzsākot projektu tika noteikts, ka visa projekta īstenošanai atvēlētais laiks būs 4 kalendārie mēneši (atbilstoši ap 3,5 personmēneši). Pamatojoties uz termiņu darba vadītājs izklāstīja iespējamos uzdevumus un to būtību. Balstoties uz šo informāciju autors vēlējās veikt aptuvenu darbietilpības novērtējumu. Lai to izdarītu tika mēģināts meklēt rakstus par līdzīgu projektu izstrādes gaitu un ilgumu apskatītajās matemātiskajās vidēs tomēr šādu informāciju atrast neizdevās.

Tā kā autoram šis bija pirmais projekts un iepriekšējas pieredzes, lai zinātu kāda varētu būt aptuvenā darbietilpība vai produktivitāte (kādi svari būtu kuriem koeficientiem jāpiemēro un pēc kādas metrikas), nebija, tad tika nolemts, ka izvēlēto daļu norādīto uzstādījumu (kuri arī ir aprakstīti Programmatūras prasību specifikācijā) varētu aptuveni paveikt atvēlētajā laikā.

Darbietilpības novērtēšanai netika izmantotas tādas metodes kā COCOMO vai funkcipunkti, jo arī to izmantošanai nepieciešama pieredze, atvēlētais laiks bija mazāk kā 6 mēneši un izstrāde bija jāveic vienai personai.

Lielu daļu laika aizņēma teorētiskā materiāla apguve un iepazīšanās ar matemātisko vidi Mathematica (sākotnēji tika apskatīti arī MATLAB pamati). Augsta līmeņa valodu izmantošanas pieredzes un zināšanu autoram uzsākot darbu nebija.

Tā pat arī darba gaitā funkcijas tika pārstrādātas vai uzlabotas atklātu kļūdu vai jaunu iegūto zināšanu rezultātā. Ne tik daudz pieredzes vai zināšanu, atvēlētā laika trūkums, autora prāt, padarītu darbietilpības novērtēšanu ar tādām metodēm kā COCOMO apšaubāmu, cik tieši mācību process un funkciju pārstrādāšana.

Ar teorētiskā materiāla apguvi, dokumentācijas sagatavošanu, programmēšanu un testēšanu kopā patērētais laiks tuvu sakrita ar darba īstenošanai atvēlēto laiku.

Rezultāti un secinājumi

Darba izstrādes rezultātā ir veiksmīgi izveidots funkcionējošs, noteiktajai prasību specifikācijai atbilstošs programmprodukts. Tas ļauj noteikt Būla funkciju jūtīgumu, bloku jūtīgumu, dažādi iegūtus pārstāvošos un vairākus tuvinātos polinomus.

Autors darba rezultātā ieguva vispārēju priekšstatu par Būla funkcijām un to pielietojumiem, iepazinās ar projekta dzīves cikla posmiem, apguva jaunu programmēšanas valodu un simbolisko programmēšanas paradigmu.

Var secināt, ka specifisku programmproduktu izstrāde zinātniskām vajadzībām ir sarežģīta. Tā ir grūti programmējama un laikietilpīga.

Tāpat var secināt, ka arī lielas komerciālās matemātiskās vides dokumentācija un implementācija var būt kļūdaina, nepilnīga un neprecīza.

Šāda programmprodukta pilnīga izstrāde ir pārāk plaša, lai aptvertu kvalifikācijas darbu un ietver tikai vienkāršākos Būla funkciju raksturotājus. Pamatojoties uz to un aizvien pastāvīgo nepieciešamību pēc algoritmu pētniecības un novērtēšanas, kā arī līdzvērtīgas programmatūras ierobežojumiem darbu būtu nepieciešams turpināt attīstot iespēju noteikt sarežģītākus Būla funkciju raksturotājus un ieviest atbalstu arī pusnoteiktām funkcijām.

Pateicības

Autors izsaka pateicību darba vadītājam profesoram Jurim Smotrovam par interesantās tematikas piedāvājumu, konsultācijām, ieteikumiem un palīdzību izstrādei nepieciešamās teorētiskās daļas izpratnes veidošanā. Tā pat autors izsaka pateicību Jānim Iraidam par ar tematiku saistītiem skaidrojumiem un ieteikumiem.

Atsauces

Aaronson, Scott, The boolean function wizard 1. Nepublicēts materiāls. 08/31/2000.

O`Donell, Ryan, Analysis of Boolean Functions. Cambridge University Press, 2014.

Prūsis, Krišjānis. Būla funkciju bloku jūtīgums. LU DF Maģistra darbs. 2014.

Darba sagatavošanai izmantoti šādi dokumenti no e-studijas vides Latvijas Universitātes Datorikas fakultātei kursam "Metakurss : Kvalifikācijas darbs"
(Pieejams: http://estudijas.lu.lv/course/view.php?id=77):

- Prasības noslēguma darbu izstrādāšanai un aizstāvēšanai Latvijas Universitātē
- LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis"
- LVS 72:1996 "Ieteicamā prakse programmatūras projektējuma aprakstīšanai"

Darba gaitā, vides apgūšanai, plaši izmantoti resursi:

- http://reference.wolfram.com/language/?source=nav
- http://mathematica.stackexchange.com/

Kvalifikācijas darbs „A***tbalsta bibliotēka Būla funkciju izpētei***" izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: **Ilmārs Pužulis** ..... 01.06.2015.

Rekomendēju darbu aizstāvēšanai:

Darba vadītājs: *Dr.dat.* **Juris Smotrovs** ..... 01.06.2015.

Recenzents: *Dr.dat.* **Juris Rāts**

Darbs iesniegts 01.06.2015.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: Darja Solodovņikova .....

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2015. prot. Nr. _____

Komisijas sekretārs(-e): .....