TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutisüsteemide instituut

Madis Sander Maddison 134471

LOOGIKA FUNKTSIOONIDE SÜSTEEMI TÄPNE MINIMEERIJA

Bakalaurusetöö

Juhendaja: Prof. Peeter Ellervee

Tallinna
Tehnikaülikool,
Arvutisüsteemide
instituut, Ph.D.

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Madis Sander Maddison

21.05.2017

Annotatsioon

Käesoleva lõputöö teemaks on Quine-McCluskey minimeerija. Selle töö eesmärgiks oli programmeerida rakendus Java programmeerimiskeeles, mis suudaks lahendada Boole'i funktsiooni kasutades Quine-McCluskey minimeerimismeetodit. Programm peab samal ajal olema lihtsalt kasutatav ning näitama visuaalselt Quine-McCluskey minimeerimist. Samuti peab programmil olema graafiline liides, kust kasutaja saaks kontrollida programmi tööd.

Selleks, et paremini kirjeldada kuidas programm töötab, on lõputöös välja toodud programmis rakendatud meetodid ning kirjeldatud seoseid nende vahel. Samuti on toodud näide sellest kuidas Quine-McCluskey meetodit rakendada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 6 peatükki, 13 joonist, 6 tabelit.

Abstract

Exact Minimizer of Logic Functions System

This Bachelor's thesis is about a program minimizing boolean logic functions. The aim of this thesis was to create a program, which would be capable of minimizing boolean logic functions using the Quine-McCluskey method. The two main parts of the Quine-McCluskey method are separated into two distinct stages. The stages had to be representable in a visual way, in which the user could easily work out, what the program was doing.

Each stage was implemented seperately, since the second stage of the minimization method can't be called out, before the first part is finished.

The user can access the Quine McCluskey minimzation program functions through a graphical interface, which allows the user to easily input the boolean functions through text form, which is highly convenient.

As part of this thesis, the effectiveness of different minimization programs found on the web were compared against eachother and the solution programmed during the course of this thesis.

The thesis is in Estonian and contains 47 pages of text, 6 chapters, 13 figures, 6 tables.

Lühendite ja mõistete sõnastik

QMM	Quine-McCluskey Meetod

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Töö teemaks on realiseerida Quine-McCluskey Minimeerimismeetod loogikafunktsioonide süsteemile kasutades programmeerimiskeelt Java. Quine-McCluskey algoritm on meetod, mille leiutas Willard V. Quine ja laiendas Edward J. McCluskey. [1] [2]

Meetodit kasutatakse Boole'i funktsioonide minimeerimiseks, funktsionaalselt sarnaneb meetod Karnaugh kaardile, aga tabeli põhiline vorm teeb ta tõhusamaks arvuti algoritmides kasutamiseks. Meetod töötab kahel astmel, kus esiteks leitakse funktsiooni lihtimplikandid. Teiseks kasutame leitud lihtimplikante, et leida minimaalne arv vajalikke implikante, et katta ära funktsioonis kehtestatud vektorid. Meetodit kasutatakse tihti ainetes Diskreetne Matemaatika kui ka Digitaalsüsteemid, seega tuleks programm kasuks nendes ainetes.

Lõputöö eesmärgiks on luua Java programm, mis demonstreerib Quine-McCluskey meetodit visuaalselt ja annab võimalikult täpse vastuse vastava meetodi alusel.

Programmi arendamisel peamiseks probleemiks osutus meetodi arendamine viisil, mis võimaldaks näidata minimeerimismeetodi tööd samm-haaval.

Ülesanded mis lahendasin lõputöö käigus:

- Kirjutada programm, mis demonstreerib Quine-McCluskey meetodit Javas.
- Anda programmile graafiline liides, mille kaudu on võimalik meetodit rakendada.
- Lasta programmil näidata visuaalselt seoseid erinevate etappide vahel.

Realisatsioon peab võimaldama algoritmi täitmist samm-haaval koos võimalusega näidata seoseid sammude vahel. Seoste näitamine ja sammude juhtimine toimub graafilise liidese abil.

2 Lahendused

2.1 Quine McCluskey baaslahenduse näide

Quine-McCluskey meetod koosneb kahest etapist. Esiteks tuleb leida lihtimplikandid kasutades süstemaatiliselt kleepimisseaduseid. Selles töös on kasutatud intervall-meetodit. Teise etapi käigus minimeeritakse leitud loetelu.

f(x1,x2,x3,x4) = $\sum(0,2,4,5,6,7,10,12,13,15)_1$

Esimene etapp – lihtimplikantide hulga leidmine.

Esiteks kirjutame kõik funktsiooni ühtede piirkonna numbrid ümber binaarsele kujule: 0 = 0000, 2 =0010, 4 = 0100, 5= 0101, 6= 0110, 7= 0111, 10= 1010 , 12 = 1100,13= 1101, 15 = 1111.

Seejärel jaotame nad intervallideks kahendvektorite nn. Indeksite järgi. Boole'i vektori indeks on ühtede arv selles vektoris. [3]

Tabel 1. QMM vektorid.

Indeks	**Intervall**	**Märge**
0	0000	
1	0010 0100	
2	0101 0110 1010 1100	
3	0111 1101	
4	1111	

Ilmselt on omavahel kleebitavad vaid need kahendvektorid, mille indeksid erinevad täpselt ühe võrra Seejuures langevad (n-1) argumendi väärtused kokku ja ühe argumendi väärtus on kleebitavates vektorites erinev.

Pärast indeksite määramist toimub kleepmisseaduse alusel intervallide tabelite koostamine.

Esimese etapi lõpuks saadakse kõigi antud funktsiooni lihtimplikantide loetelu. [3]

Tabel 2. QMM esimene etapp.

Indeks	**Intervall**	**Märge**	**Indeks**	**Intervall**	**Märge**	**Indeks**	**Intervall**	**Märge**
0	0000	x	0-1	00-0 0-00	x x	0-1-1-2	0--0 0--0	A2 A3
1	0010 0100	x x	1-2	0-10 -010 010- 01-0 -100	x A1 x x x	1-2-2-3	01-- -10-	A4 A5
2	0101 0110 1010 1100	x x x x	2-3	01-1 011- -101 110-	x x x x	2-3-3-4	-1-1	A6
3	0111 1101	x x	3-4	-111 11-1				
4	1111	x						

Teine etapp

Teise etapi käigus seda loetelu minimeeritakse s.t. valitakse minimaalne alamhulk lihtimplikantidest, mis võimaldavad katta antud funktsiooni ühtede piirkonna (s.o. tüüpiline nn. katteülesanne). 0,2,4,5,6,7,10,12,13,15

Tabel 3. QMM teine etapp.

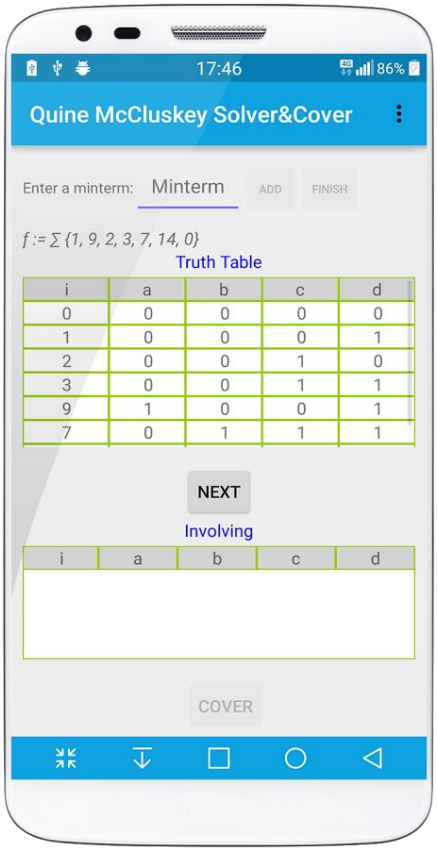
Implikandid	**0**	**2**	**4**	**5**	**6**	**7**	**10**	**12**	**13**	**15**
A1		x				x				
A2	x		x		x					
A3			x	x	x			x		
A4			x	x			x		x	
A5				x				x	x	x

Lahendis osalevad lihtimplikandid peavad katma funktsiooni ühtede piirkonna. Lihtimplikandid A1, A2, A4 ja A5 on igal juhul vajalikud, kuna nad võimaldavad ainsatena katta vektoreid 0, 2, 7 ja 15 (tabelis halli taustaga). Peale seda on näha , et kui kattes ära implikandid A1, A2, A4 ja A5 pole implikanti A3 enam vaja ,kuna vektorid 4, 5, 6 ja 12 on kaetud juba teiste lihtimplikantide poolt ning lõpptulemuseks jääb:

$$f(x_1, x_2, x_3, x_4) = A1 \vee A2 \vee A4 \vee A5 = \overline{x_2} x_3 \overline{x_4} \vee \overline{x_1 x_4} \vee x_2 \overline{x_3} \vee x_2 x_4$$

2.2 Olemasolevad lahendused

Quine-McCluskey Solver&Cover, loodud MC-Soft poolt. [4]



Joonis 1. Quine-McCluskey Solver&Cover MC-Soft.

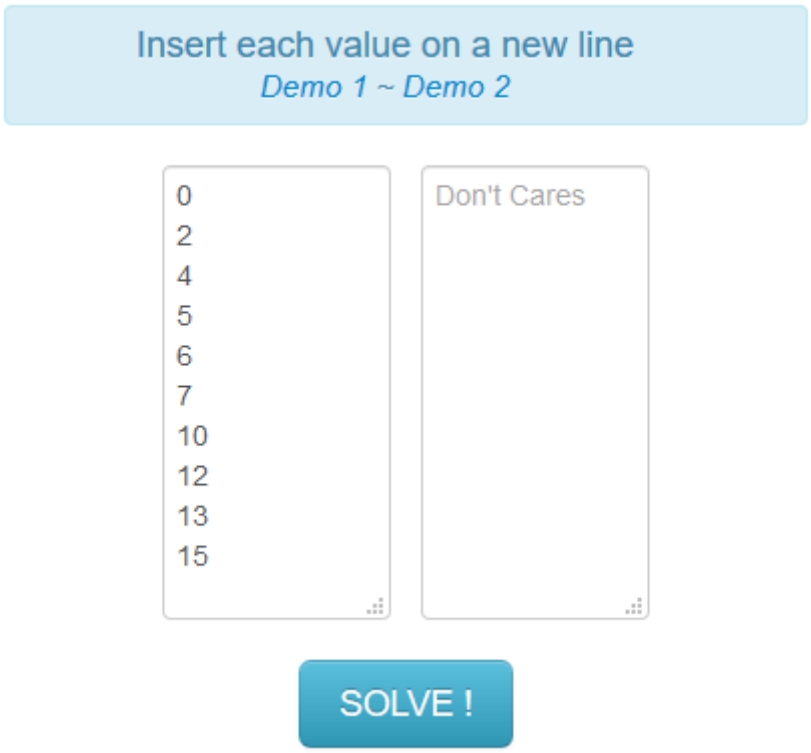
Head omadused: lihtsalt kasutatav, on võimalik kasutada androidil põhinevatel mobiiltelefonidel.

Halvad omadused: Täisversioon on tasuline.

Lõpp vastus , kasutades ülaltoodud näidet:

$$f(a,b,c,d) = P1 \vee P0 \vee P3 \vee P4 = \overline{b}c\overline{d} \vee \overline{a}\overline{d} \vee b\overline{c} \vee bd$$

QuineMcCluskeySolver, loodud Hatem Hassani poolt. [5]



Joonis 2. QuineMcCluskey Solver

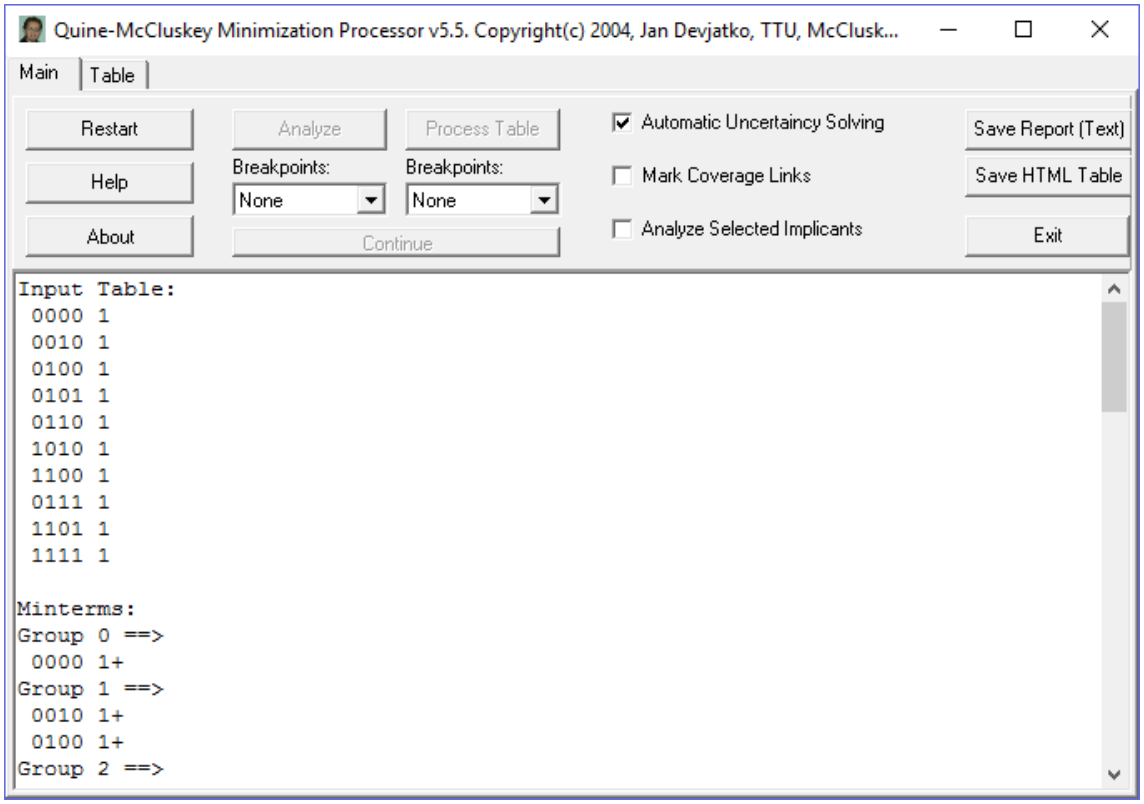
Head omadused: visuaalselt lihtne ning arusaadav algajale, on näha samm-haaval kuidas jõuti vastuseni.

Halvad omadused: ei sobi keerulistemate Boole'i funktsioonide jaoks, kus on vaja mitut väljundit.

Lõpp vastus , kasutades ülaltoodud näidet:

$$f(A,B,C,D) = P1 \vee P0 \vee P3 \vee P4 = \mathrm{B'CD'} + \mathrm{BD} + \mathrm{BC'} + \mathrm{A'D'}$$

QMMinPro, loodud Jan Devjatko poolt. [6]



Joonis 3. QMMinPro Jan Devjatko

Head omadused: hästi kasutatav, õpetused olemas, sammhaaval protsess, visuaalne.

Halvad omadused: kergelt aegunud, ei tööta kõikidel operatsioonisüsteemidel, jääb „hängima“ suuremate funktsioonide puhul.

Kuna see variant oli kõige parem ning juhendaja poolt soovitatud, jäi ülesandeks taolise programmi kirjutamine Java baasil.

Lõpp vastus kasutades ülaltoodud näidet:

```
0--0 1 B
-010 1 A
-10- 1 C
-1-1 1 E
```

3 Tarkvaraline realisatsioon

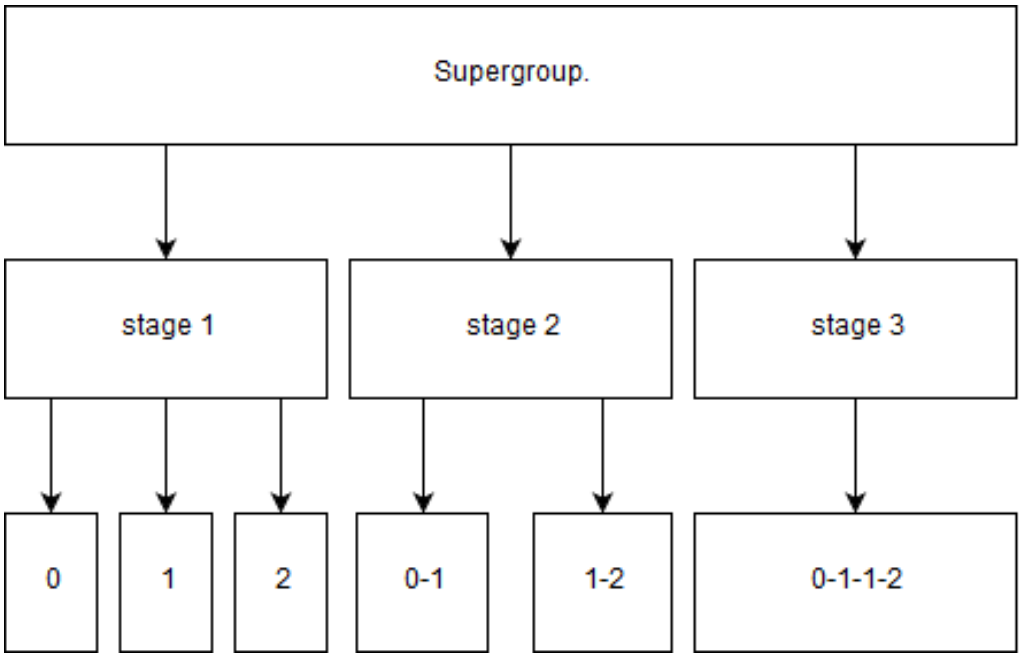
Lahendus on jaotatud peamiselt kahte erinevasse ossa, esimene osa tegeleb Quine-McCluskey meetodi minimeerimisega ja teine osa ehk graafiline liides.

Järgnevalt on näidatud sammud, mille järgi programm rakendab Quine-McCluskey Minimeerimismeetodit.

1. Loeb kasutaja poolt sisestatud vektorid programmi mälusse ning kontrollib nende vastavust ette antud mallile.

2. Programm kirjutab vektori lahti väljundi alusel. Nt: xxxx 11- läheb xxxx 100, xxxx 010, xxxx 001.

3. Lisab vektorid supergruppi, kus hakkavad kõik implikandid olema. Supergrupp hoiab kõiki esimese etapi samme eraldi gruppides(stage), mille sees omakorda on kõik vektorid, mis on indeksite kaudu gruppideks jaotatud.



Joonis 4. Supergrupi ehitus.

4. Programm kontrollib omavahel kahte indeksite gruppi vektoreid, et leida implikante. Kontrollimis meetod teeb kaks eraldi kontrolli, üks sisendite ning teine väljundite jaoks.

   Sisendite kontroll käsitleb vektori esimest poolt, ehk sisendit, leides vektoreid, mis erinevad üksteisest ainult ühe koha võrra, näiteks(-110 01 ja -100 01). Samal ajal kontrollib ta, et vektori väljundid oleks võrdsed. Leides sellise paari moodustab programm uue vektori (-1-0 01), mille programm salvestab ajutisse gruppi, kus ta samal ajal kontrollib, et varem ei oleks mingi teine vektorite paar samasugust vektorit moodustanud, näiteks (11-0 01 ja 01-0 01 moodustavad samuti -1-0 01 vektori).

   Väljundite kontroll käsitleb vektori teist poolt ehk väljundit. Kuigi kontrollimisviis sarnaneb sisendite kontrollile, moodustab meetod uusi vektoreid teistmoodi. Programm kontrollib, et vektorid erineksid üksteisest ainult kahe koha võrra, näiteks(1100 01 ja 1100 10). Leides sellise paari, moodustab programm uue vektori (1100 11), aga erinevalt sisendkontrollist, ei saa väljund olla implikantide seas „don’t care“ ehk (-). Seetõttu loob programm vektori, kus mõlemad väljundid on tõesed. Ning sarnaselt sisendite kontrollile, kontrollib programm, et varasemalt pole samasugust vektorit moodustatud

   Peale mõlema kontrolli tegemist kombineerib programm mõlemad ajutised grupid ühte gruppi kontrollides, et ei leiduks duplikaate. Peale seda programm salvestab uue grupi Supergruppi järgmise grupi(stage) algusesse, ning võtab järgmised kaks indeksite gruppi, kui võimalik.

5. Programm loob algsete vektorite ja lihtimplikantide alusel tabeli ning leiab milliseid vektoreid implikandid katavad.

6. Kasutades eelnevalt leitud lihtimplikante ning vektoreid mida neid katavad, hakkab programm leidma implikante, mida on vaja, et minimaalsete arvu implikantidega, katta ära kõik algvektorid. Programm leiab implikante niiöelda tähtsuse järjekorras.

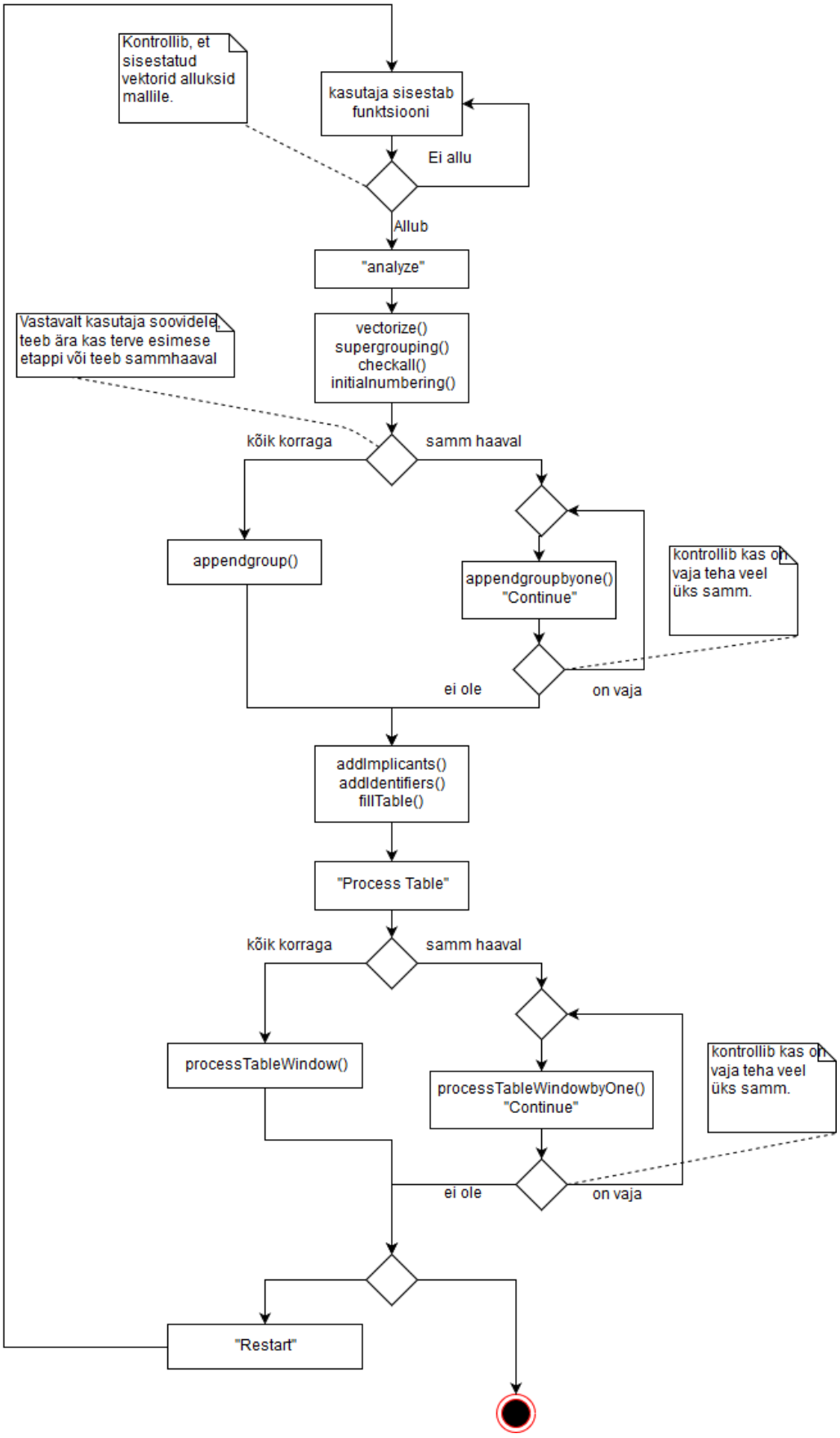
   Esimeseks sammuks on eemaldada valikutest kõik implikandid, mis on teiste implikantide poolt kaetud.

Teiseks valib programm implikandid, mis üksikuna katavad mingit vektorit,.

Peale igat implikandi valikut eemaldab programm tabelist kõik vektorid, mida implikant kattis. Ning kui implikant on valitud, proovib programm uuesti leida, kas on tekkinud implikante, mis on teiste poolt kaetud peale vektorite eemaldamist.

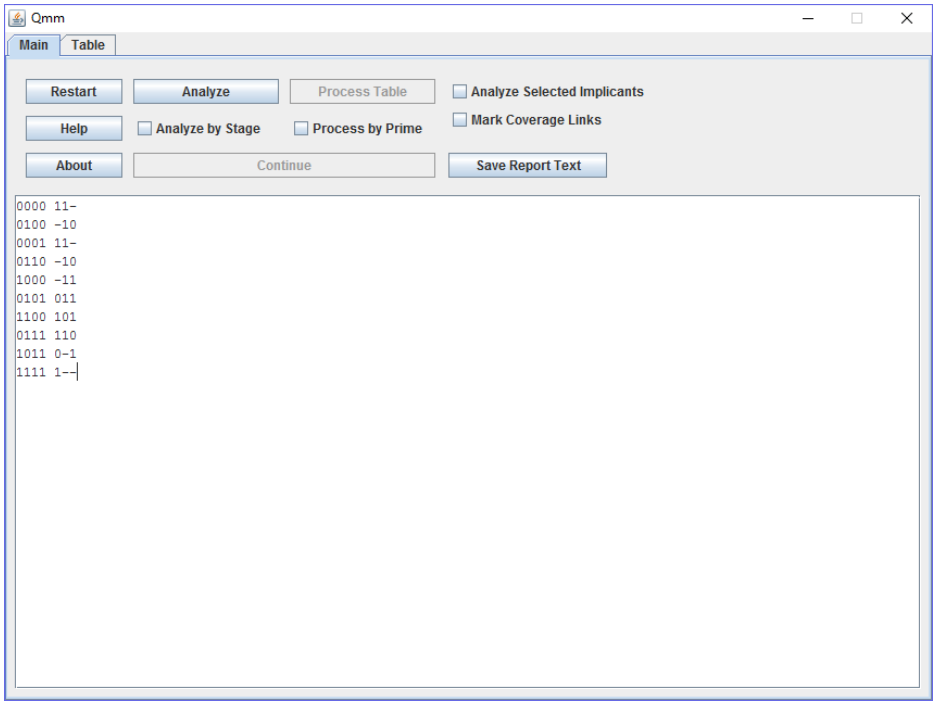
7. Peale implikantide leidmist väljastab programm vastavad implikandid teksti väljale, peale mida saab kasutaja kontrollida, milliseid eelnevaid vektoreid implikant kattis ning soovi korral salvestada terve programmi tegevuse eraldi faili.

8.

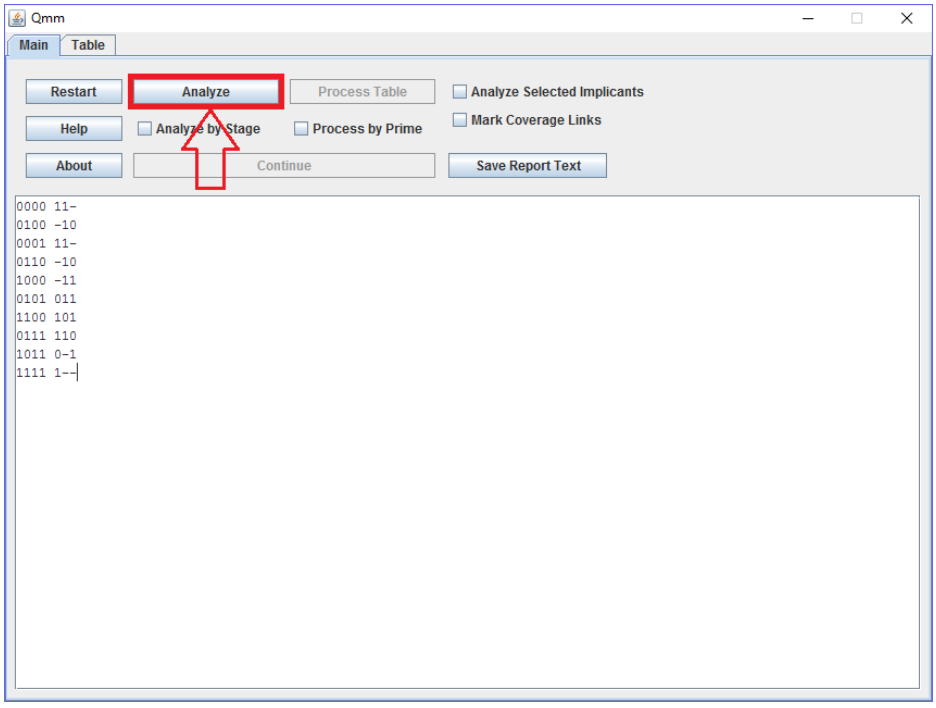


Joonis 5. Voodiagramm

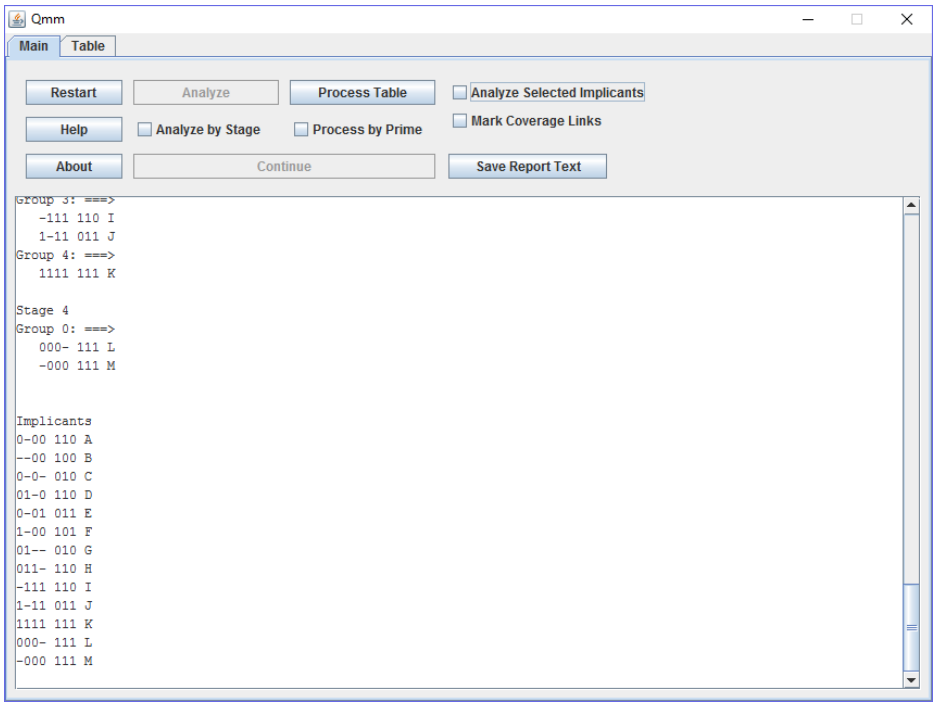
3.1 Kasutusjuhend



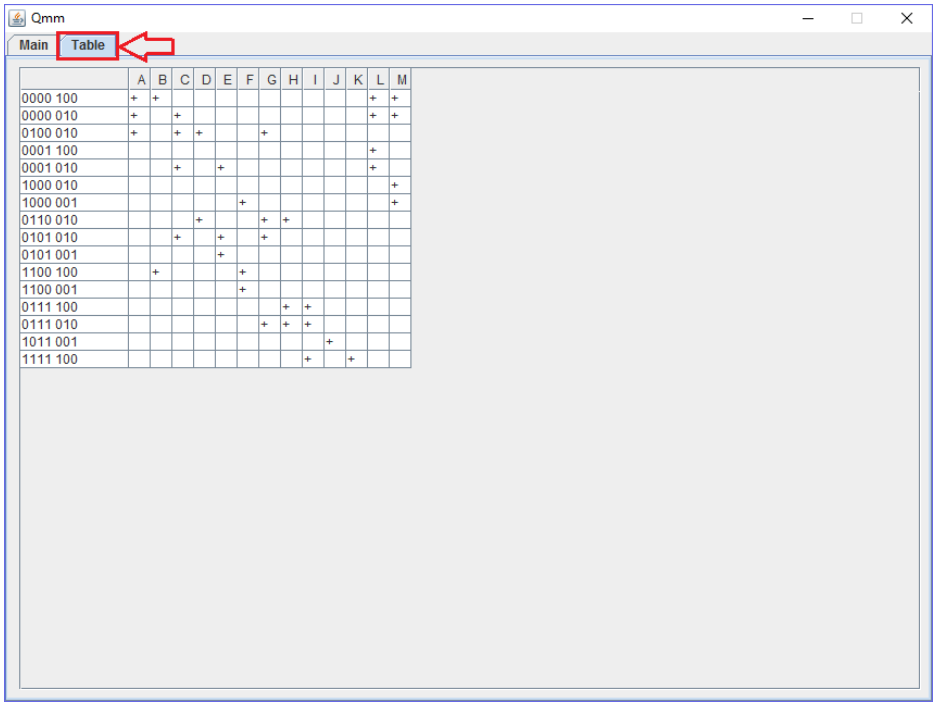
Kasutaja sisestab teksti välja oma funktsiooni algandmed.

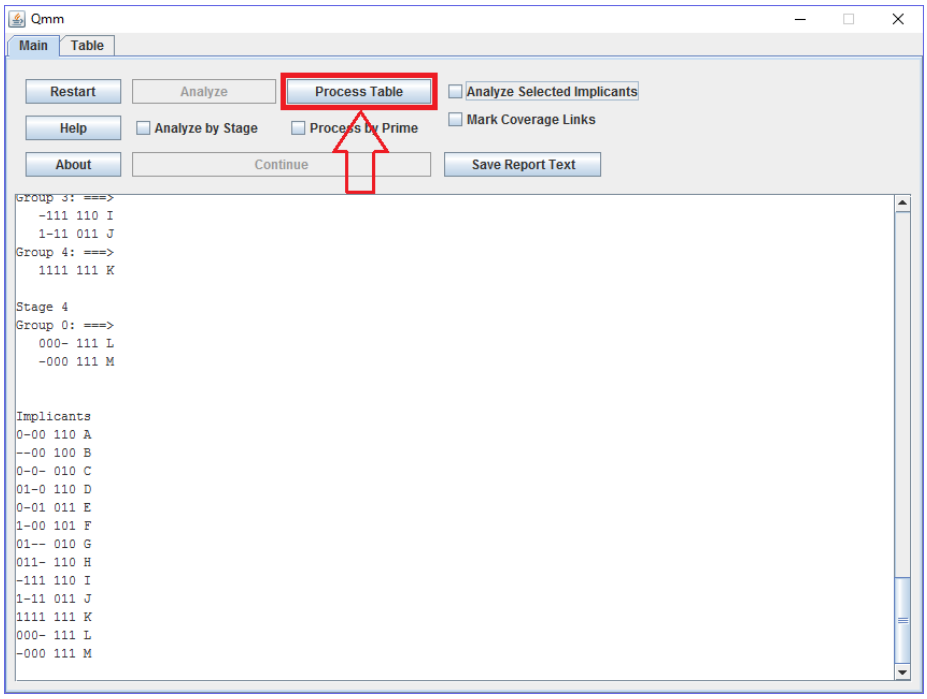


Kasutaja vajutab „Analyze“ nuppu, et alustada minimeerimisfunktsiooni esimest etappi.

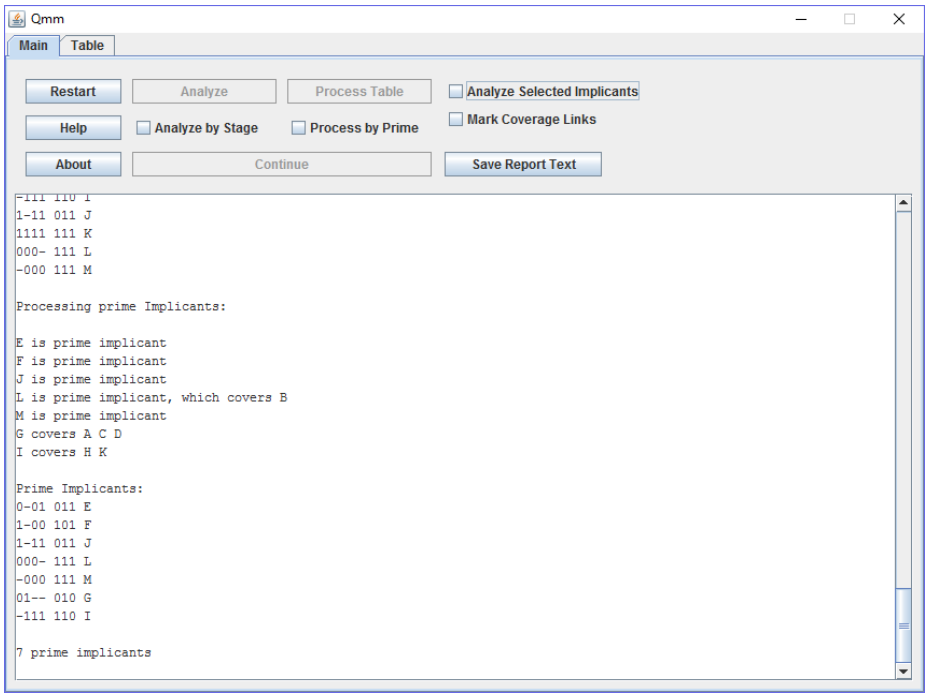


Kasutaja saab esimese etapi käigus leitud implikandid ning mooduse, kuidas neid leiti. Kasutaja saab leitud implikante analüüsida, mis näitab kuidas selle implikandini jõuti ning kui võimalik siis näha, milliste implikantide koostamisel valitud implikanti vaja on.





Kasutaja vajutab „Process Table“ nuppu, et alustada minimeerimisfunktsiooni teist etappi.



Kasutaja saab teise etappi käigul leidud lihtimplikandid ning mooduse, kuidas neid leiti. Nüüd võib kasutaja veel analüüsida implikante, salvestada kasutades „Save Report Text“ nuppu, või alustada uue funktsiooniga kasutades „Restart“ nuppu.

3.2 Programmi struktuur

Programmi valmistamisel loodi mitmeid erinevaid meetodeid. Programmi peamiseks osaks on Checker Java class, kus asuvad kõik peamised funktsioonid, mis on seotud Quine-McCluskey Minimeerimismeetodiga.

Järgnevalt on kirjeldatud programmi põhilisi meetodeid.

1. isNumeric()
   Vaatab kas vektorid vastavad kujule xxxx yyyy, kus x võib olla 1 või 0 ning y võib olla 1,0 või -. **Vt** Joonis 8, **lk** 44.

2. multiply()
   Loob ühest algvektorist mitu erinevat vektorit, nt kujult xxxx 01- kujule xxxx 010 ja xxxx 001, kus viimane on samuti märgitud, et on „don't care". **Vt** Joonis 9 **lk** 44.

3. checkCoverage()
   Töötleb tabelit, leiab millised implikandid katavad milliseid vektoreid. **Vt** Joonis 10 **lk** 45.

4. getCovers()
   Leiab kõik implikandid, mis on kaetud või katavad valitud implikanti „mark coverage links" jaoks. **Vt** Joonis 11**lk** 45.

5. check()
   Kasutades functionChecki ja outputChecki leiab, mis vektorid katavad teineteist kleepimisseaduste alusel. **Vt** Joonis 12 **lk** 46.

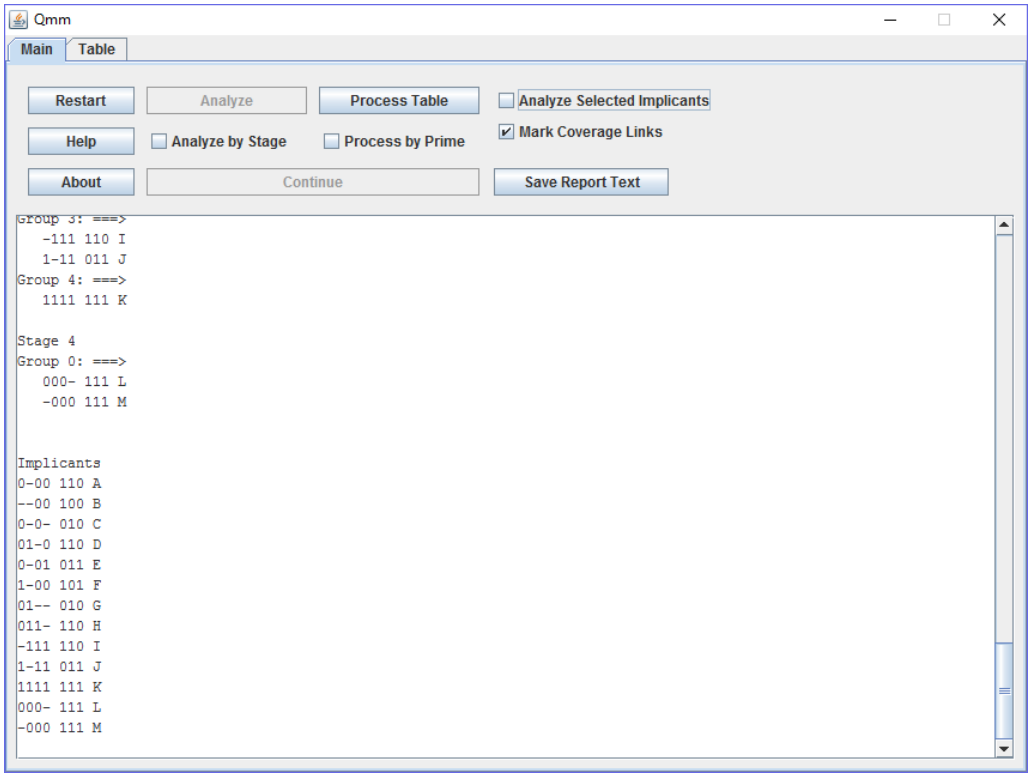
6. functionCheck()
   Kontrollib vektori sisendit teiste vektorite suhtes.

7. outputCheck()
   Kontrollib vektori väljundit teiste vektorite suhtes.

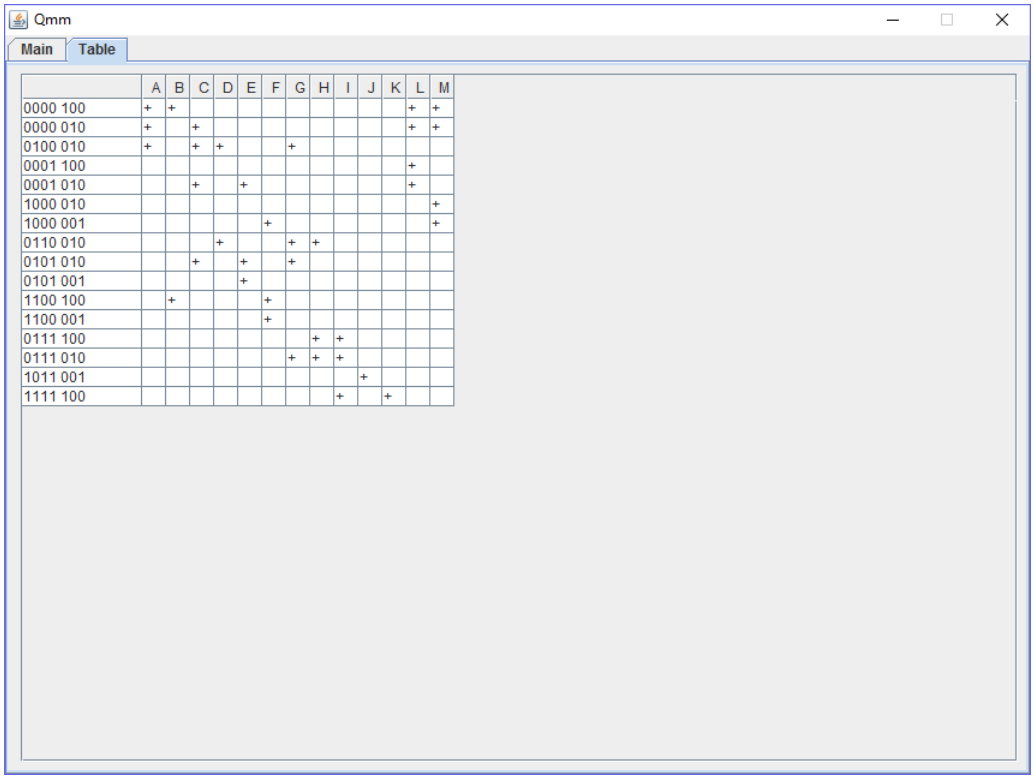
8. getPrime()

   Leiab kõik lihtimplikandid mida on vaja, et minimaalse arvu implikantidega katta kõik vajalikud vektorid. **Vt** Joonis 13 **lk** 47.

3.3 Graafiline kasutajaliides



Joonis 6. QMM paneel 1.



Joonis 7. QMM paneel 2.

Rakendus koosneb kahest paneelist. Esimesel paneelil (Joonis 6. QMM paneel 1.), toimub peamine programmi rakendus. Teisel paneelil (Joonis 7. QMM paneel 2.) asub tabel vastavate vektorite ja liht implikantidega.

Ülaosas asub menüü, kus asuvad kõik nupud programmi kasutamiseks.

„Restart“ seab programmi algseisukorda, kust on võimalik uut funktsiooni minimeerida.

„Help“ toob ette akna, kus on juhend programmi kasutamiseks.

„About“ avab akna, kus on infot programmi kohta.

„Analyse“ alustab analüüsimisprotsessi ja olenevalt valikutest, programm kas leiab kõik implikandid korraga, või laseb kasutajal gruppide kaupa läbi käia.

„Process Table“ käivitab Quine-McCluskey Minimeerimismeetodis teise etappi, mis leiab lihtimplikante, mis kokku kõik katavad algvektorid.

Lahendusväli on lihtne text väli, kuhu programm mitme erineva print() funktsiooni abil programmi tööd väljastab. „Save Report Text“ salvestab kõik tekstiväljas oleva info tekstifaili.

3.4 Võrdlus

Võrreldes QMMinPro-ga töötleb minu programm samu andmeid palju kiiremini, eriti esimeses etapis, kus QMMinPro on mõnikord kümneid kordi aeglasem, aga see eest on QMMinPro teise etapi töötlus natuke kiirem kui minu programmis.

Kuigi QuineMcCluskeySolver ja Quine-McCluskey Solver&Cover töötavad hästi omas skoobis, on nende üleüldine jõudlus võrreldes teiste lahendustega väiksem , kuna nad toetavad samaaegselt ainult ühe väljundi leidmist, ning seetõttu ei sobi nad analüüsimiseks.

4 Analüüs

Töö käigus testiti lõputöös esitatud programmi effektiivsust võrreldes teiste lahendustega. Testimiseks valitud lähteülesanded valiti erinevate raskusastmetega. Alates lihtsama ülesandega, kus sisendeid oli neli ning väljundeid seitse (**Vt** Lähteülesanne 1 **lk** 33). Lõpetades lähteülesandega, kus sisendeid oli kaheksa ning väljundeid kaheksa (**Vt** Lähteülesanne 3 **lk** 36),

Kuigi veebilehekülgedel asuvad minimeerisrakendused töötavad ning on lihtsalt kasutavad, on nende effektiivsus suhteliselt väike, kuna enamasti on neil maksimaalne väljundite arv üks. Seetõttu ei ole nad sobilikud lähteülesande lahendamiseks. Lõputöös võrreldi QMMinPro'd [6], enda lahendust ning Espressot [7] ise, kui kontroll-lahendust.

Analüüsimiseks kasutan espressot, mille arendas Robert Brayton IBM'is [7]. Espresso kasutab heuristikat ja algoritme, et „rohkelt" minimeerida loogika funktsioone. Espresso programmiga tegin iga lähteülesande jaoks kontroll-lahenduse, kasutades käsku –Dexact, sest juhul kui kontroll-ülesanne ei tule õigesti välja, ei saa kindel olla espresso kontrolli õigsuses. Kuigi espresso kontrolltulemusel on väljundid erinevad, ei muuda see Quine-McCluskey minimeerimise õigsust, kuna espresso kasutab lõpus veel nn. hõrendamist - kui mõne väljundi puhul on implikant juba täielikult kaetud mõne teise (ja suurema) implikandi poolt, siis vastav väljund eemaldatakse. Klassikaline Quine-McCluskey meetod seda ei kasuta, sest eesmärgiks on ainult implikantide arvu minimeerimine.

Testimiseks kasutan lähteülesannet erinevate programmide peal ning kontrollimiseks kasutan espresso käsku –Dverify, et näha, kas lähteülesanne ja minimeeritud ülesanne on võrdsed.

Testimise käigus selgus, et kõik programmid minimeerivad, kuigi QMMinPro ei leia alati kõige minimaalsemat varianti. Seda on näha kolmandas lähteülesandes, kus espresso ja minu lahendus mõlemad minimeerisid 76 implikandini (**Vt** Tulemus 7,Tulemus 9 **lk** 41,43) ja QMMinPro minimeeris ainult 77 implikandini (**Vt** Tulemus 8 **lk** 42). Samuti tekkis suur erinevus minimeerimiseks kulunud aja peale, kus espresso oli tohutult kiirem mõlemast variandist (**Vt** Tabel 4,Tabel 5 ja Tabel 6 **lk** 30). Aga võrreldes peamiselt minu lahendust ja QMMinPro, on kulunud aegade vahe siiski suur,

olles esimesel kahel juhul (**Vt** Tabel 4 ja Tabel 5 **lk** 30) keskmiselt 3 korda aeglasem ning ligi 8.5 korda aeglasem kõige raskemal lähteülesandel (**Vt** Tabel 6 **lk** 30). Suurim ajaerinevus tuli esimese etapi töötluses ning teise etappi puhul, oli minu lahendus ainult natukene kiirem.

Tabelis kasutatud lähteülesanded (**Vt** Lähteülesanne 1,Lähteülesanne 2 ja Lähteülesanne 3 **lk** 33, 34 ja 36) on näha lisades.

Tabel 4. Lähteülesanne 1.

	Sisendite arv	**Väljundite arv**	**Lihtimplikante**	**Kulunud aeg**	**-Dverify**
QMMinPro	4	7	9	3 sek	PLA's compared equal
Minu lahendus	4	7	9	1 sek	PLA's compared equal
espresso	4	7	9	0.04 sek	PLA's compared equal

Tulemused näidatud lisas (**Vt** Tulemus 1,Tulemus 2 ja Tulemus 3 **lk** 38).

Tabel 5. Lähteülesanne 2.

	Sisendite arv	**Väljundite arv**	**Lihtimplikante**	**Kulunud aeg**	**-Dverify**
QMMinPro	8	5	57	30 sek	PLA's compared equal
Minu lahendus	8	5	57	9 sek	PLA's compared equal
espresso	8	5	57	0.1 sek	PLA's compared equal

Tulemused näidatud lisas (**Vt** Tulemus 4, Tulemus 5 ja Tulemus 6 **lk** 39 ja 40).

Tabel 6, Lähteülesanne 3.

	Sisendite arv	**Väljundite arv**	**Lihtimplikante**	**Kulunud aeg**	**-Dverify**
QMMinPro	8	8	77	216 sek	PLA's compared equal
Minu lahendus	8	8	76	25 sek	PLA's compared equal
espresso	4	8	76	0.16 sek	PLA's compared equal

Tulemused näidatud lisas (Vt Tulemus 7, Tulemus 8 ja Tulemus 9 lk 41, 42 ja 43).

5 Kokkuvõte

Lõputöö käigus sai välja arendatud lihtne tarkvara, millega on võimalik minimeerida etteantud Boole'i funktsioon. Rakendus võiks leida kasutust inimeste seas, kellel on vaja minimeerida Boole'i funktsioone ning soovivad kontrollida enda lahendust või kontrollida, kuidas Quine-McCluskey meetod töötab. Rakendusest on eemaldatud arenduse käigus mitmed ebastabiilsused ning kõik silumise käigus leitud tõrked on parandatud. Seega võib töö eesmärke lugeda õnnestunuks.

6 Kasutatud kirjandus

[1] Willard Van Orman Quine, ""A Way to Simplify Truth Functions"," *The American Mathematical Monthly*, pp. 627–631, November 1955.

[2] Willard Van Orman Quine, ""The Problem of Simplifying Truth Functions"," *The American Mathematical Monthly*, pp. 521–531, October 1952.

[3] Margus Kruus. www.pld.ttu.ee/~kruus/diskmat/Loeng_3_1.doc.

[4] MC-Soft. (2017, May) https://play.google.com/store/apps/details?id=com.mcsoft.quine_mccluskeysolvercover&hl=en.

[5] Hatem Hassan. (2017, May) http://www.quinemccluskey.com/.

[6] Jan Devjatko. Digitaalsüsteemid / Digital Systems. [Online]. http://mini.pld.ttu.ee/~lrv/IAY0150/

[7] Robert Brayton. espresso. [Online]. http://mini.pld.ttu.ee/%7Elrv/espresso/

Lisa 1 – Lähteülesanded

Lähteülesanne 1

```
.i 4
.o 7
0000 1111110
0001 0110000
0010 1101101
0011 1111001
0100 0110011
0101 1011011
0110 1011111
0111 1110000
1000 1111111
1001 1111011
1010 0000000
1011 0000000
1100 0000000
1101 0000000
1110 0000000
1111 0000000
```

Lähteülesanne 2

```
.i 8
.o 5
00000000 00000
00000001 00001
00000010 00001
00000011 00010
00000100 00010
00000101 00010
00000110 00010
00000111 00011
00001000 00011
00001001 00011
00001010 00011
00001011 00011
00001100 00011
00001101 00100
00001110 00100
00001111 00100
00010000 00100
00010001 00100
00010010 00100
00010011 00100
00010100 00100
00010101 00101
00010110 00101
00010111 00101
00011000 00101
00011001 00101
00011010 00101
00011011 00101
00011100 00101
00011101 00101
00011110 00101
00011111 00110
00100000 00110
00100001 00110
00100010 00110
00100011 00110
00100100 00110
00100101 00110
00100110 00110
00100111 00110
00101000 00110
00101001 00110
00101010 00110
00101011 00111
00101100 00111
00101101 00111
00101110 00111
00101111 00111
00110000 00111
00110001 00111
00110010 00111
00110011 00111
00110100 00111
00110101 00111
00110110 00111
00110111 00111
00111000 00111
00111001 01000
00111010 01000
00111011 01000
00111100 01000
00111101 01000
00111110 01000
00111111 01000
01000000 01000
01000001 01000
01000010 01000
01000011 01000
01000100 01000
01000101 01000
01000110 01000
01000111 01000
01001000 01000
01001001 01001
01001010 01001
01001011 01001
01001100 01001
01001101 01001
01001110 01001
01001111 01001
01010000 01001
01010001 01001
01010010 01001
01010011 01001
01010100 01001
01010101 01001
01010110 01001
01010111 01001
01011000 01001
01011001 01001
01011010 01001
01011011 01010
01011100 01010
01011101 01010
01011110 01010
01011111 01010
01100000 01010
01100001 01010
01100010 01010
01100011 01010
01100100 01010
01100101 01010
01100110 01010
01100111 01010
01101000 01010
01101001 01010
01101010 01010
01101011 01010
01101100 01010
01101101 01010
01101110 01010
01101111 01011
01110000 01011
01110001 01011
01110010 01011
01110011 01011
01110100 01011
01110101 01011
01110110 01011
01110111 01011
01111000 01011
01111001 01011
01111010 01011
01111011 01011
01111100 01011
01111101 01011
01111110 01011
01111111 01011
10000000 01011
```

```
10000001 01011
10000010 01011
10000011 01011
10000100 01011
10000101 01100
10000110 01100
10000111 01100
10001000 01100
10001001 01100
10001010 01100
10001011 01100
10001100 01100
10001101 01100
10001110 01100
10001111 01100
10010000 01100
10010001 01100
10010010 01100
10010011 01100
10010100 01100
10010101 01100
10010110 01100
10010111 01100
10011000 01100
10011001 01100
10011010 01100
10011011 01100
10011100 01100
10011101 01101
10011110 01101
10011111 01101
10100000 01101
10100001 01101
10100010 01101
10100011 01101
10100100 01101
10100101 01101
10100110 01101
10100111 01101
10101000 01101
10101001 01101
10101010 01101
10101011 01101
10101100 01101
10101101 01101
10101110 01101
10101111 01101
10110000 01101
10110001 01101
10110010 01101
10110011 01101
10110100 01101
10110101 01101
10110110 01101
10110111 01110
10111000 01110
10111001 01110
10111010 01110
10111011 01110
10111100 01110
10111101 01110
10111110 01110
10111111 01110
11000000 01110
11000001 01110
11000010 01110
11000011 01110
11000100 01110
11000101 01110
11000110 01110
11000111 01110
11001000 01110
11001001 01110
11001010 01110
11001011 01110
11001100 01110
11001101 01110
11001110 01110
11001111 01110
11010000 01110
11010001 01110
11010010 01110
11010011 01111
11010100 01111
11010101 01111
11010110 01111
11010111 01111
11011000 01111
11011001 01111
11011010 01111
11011011 01111
11011100 01111
11011101 01111
11011110 01111
11011111 01111
11100000 01111
11100001 01111
11100010 01111
11100011 01111
11100100 01111
11100101 01111
11100110 01111
11100111 01111
11101000 01111
11101001 01111
11101010 01111
11101011 01111
11101100 01111
11101101 01111
11101110 01111
11101111 01111
11110000 01111
11110001 10000
11110010 10000
11110011 10000
11110100 10000
11110101 10000
11110110 10000
11110111 10000
11111000 10000
11111001 10000
11111010 10000
11111011 10000
11111100 10000
11111101 10000
11111110 10000
11111111 10000
```

Lähteülesanne 3

```
.i 8
.o 8
00000000 00000001
00000001 00000110
00000010 00001011
00000011 00010000
00000100 00010101
00000101 00011010
00000110 00011111
00000111 00100100
00001000 00101001
00001001 00101110
00001010 00110011
00001011 00111000
00001100 00111101
00001101 01000010
00001110 01000111
00001111 01001100
00010000 01010001
00010001 01010110
00010010 01011011
00010011 01100000
00010100 01100101
00010101 01101010
00010110 01101111
00010111 01110100
00011000 01111001
00011001 01111110
00011010 10000011
00011011 10001000
00011100 10001101
00011101 10010010
00011110 10010111
00011111 10011100
00100000 10100001
00100001 10100110
00100010 10101011
00100011 10110000
00100100 10110101
00100101 10111010
00100110 10111111
00100111 11000100
00101000 11001001
00101001 11001110
00101010 11010011
00101011 11011000
00101100 11011101
00101101 11100010
00101110 11100111
00101111 11101100
00110000 11110001
00110001 11110110
00110010 11111011
00110011 00000000
00110100 00000101
00110101 00001010
00110110 00001111
00110111 00010100
00111000 00011001
00111001 00011110
00111010 00100011
00111011 00101000
00111100 00101101
00111101 00110010
00111110 00110111
00111111 00111100
01000000 01000001
01000001 01000110
01000010 01001011
01000011 01010000
01000100 01010101
01000101 01011010
01000110 01011111
01000111 01100100
01001000 01101001
01001001 01101110
01001010 01110011
01001011 01111000
01001100 01111101
01001101 10000010
01001110 10000111
01001111 10001100
01010000 10010001
01010001 10010110
01010010 10011011
01010011 10100000
01010100 10100101
01010101 10101010
01010110 10101111
01010111 10110100
01011000 10111001
01011001 10111110
01011010 11000011
01011011 11001000
01011100 11001101
01011101 11010010
01011110 11010111
01011111 11011100
01100000 11100001
01100001 11100110
01100010 11101011
01100011 11110000
01100100 11110101
01100101 11111010
01100110 11111111
01100111 00000100
01101000 00001001
01101001 00001110
01101010 00010011
01101011 00011000
01101100 00011101
01101101 00100010
01101110 00100111
01101111 00101100
01110000 00110001
01110001 00110110
01110010 00111011
01110011 01000000
01110100 01000101
01110101 01001010
01110110 01001111
01110111 01010100
01111000 01011001
01111001 01011110
01111010 01100011
01111011 01101000
01111100 01101101
01111101 01110010
01111110 01110111
01111111 01111100
10000000 10000001
```

```
10000001 10000110
10000010 10001011
10000011 10010000
10000100 10010101
10000101 10011010
10000110 10011111
10000111 10100100
10001000 10101001
10001001 10101110
10001010 10110011
10001011 10111000
10001100 10111101
10001101 11000010
10001110 11000111
10001111 11001100
10010000 11010001
10010001 11010110
10010010 11011011
10010011 11100000
10010100 11100101
10010101 11101010
10010110 11101111
10010111 11110100
10011000 11111001
10011001 11111110
10011010 00000011
10011011 00001000
10011100 00001101
10011101 00010010
10011110 00010111
10011111 00011100
10100000 00100001
10100001 00100110
10100010 00101011
10100011 00110000
10100100 00110101
10100101 00111010
10100110 00111111
10100111 01000100
10101000 01001001
10101001 01001110
10101010 01010011
10101011 01011000
10101100 01011101
10101101 01100010
10101110 01100111
10101111 01101100
10110000 01110001
10110001 01110110
10110010 01111011
10110011 10000000
10110100 10000101
10110101 10001010
10110110 10001111
10110111 10010100
10111000 10011001
10111001 10011110
10111010 10100011
10111011 10101000
10111100 10101101
10111101 10110010
10111110 10110111
10111111 10111100
11000000 11000001
11000001 11000110
11000010 11001011
11000011 11010000
11000100 11010101
11000101 11011010
11000110 11011111
11000111 11100100
11001000 11101001
11001001 11101110
11001010 11110011
11001011 11111000
11001100 11111101
11001101 00000010
11001110 00000111
11001111 00001100
11010000 00010001
11010001 00010110
11010010 00011011
11010011 00100000
11010100 00100101
11010101 00101010
11010110 00101111
11010111 00110100
11011000 00111001
11011001 00111110
11011010 01000011
11011011 01001000
11011100 01001101
11011101 01010010
11011110 01010111
11011111 01011100
11100000 01100001
11100001 01100110
11100010 01101011
11100011 01110000
11100100 01110101
11100101 01111010
11100110 01111111
11100111 10000100
11101000 10001001
11101001 10001110
11101010 10010011
11101011 10011000
11101100 10011101
11101101 10100010
11101110 10100111
11101111 10101100
11110000 10110001
11110001 10110110
11110010 10111011
11110011 11000000
11110100 11000101
11110101 11001010
11110110 11001111
11110111 11010100
11111000 11011001
11111001 11011110
11111010 11100011
11111011 11101000
11111100 11101101
11111101 11110010
11111110 11110111
11111111 11111100
```

Lisa 2 – Minimeerimis tulemused.

Lähteülesanne 1.

Tulemus 1 Lähteülesanne 1 minu lahendus

```
0101 1011011 T
100- 1111011 X
-000 1111110 V
0100 0110011 K
0110 1011111 U
0010 1101101 P
00-1 0110000 C
0011 1111001 S
0-11 1110000 N
```

Tulemus 2 Lähteülesanne 1 QMMinPro

```
0101 1011011 T
100- 1111011 W
-000 1111110 V
0100 0110011 M
0110 1011111 U
0010 1101101 R
00-1 0110000 E
0011 1111001 S
0-11 1110000 N
```

Tulemus 3 Lähteülesanne 1 Espresso

```
0-00 0100000
0-11 1110000
0-10 1001100
01-0 0010011
-00- 0110000
001- 0101001
100- 1001011
-000 1001110
0101 1011011
```

Lähteülesanne 2.

Tulemus 4 Lähteülesanne 2 Minu lahendus

```
00-001-- 00010 s
000-0111 00001 I
1-10---- 01101 Bu
10000-00 01011 n
0010---- 00110 BH
000010-- 00011 o
-0000001 00001 B
-0000010 00001 D
-0000011 00010 F
00-01100 00011 P
-00-11-1 00100 AQ
-00-111- 00100 AS
010011-- 01001 AN
100000-- 01011 Ao
000110-- 00101 AC
00101-11 00111 AW
001011-- 00111 BA
110----- 01110 Bv
01------ 01000 Bd
10--1--- 01100 Bl
10---1-1 01100 BV
10---11- 01100 BY
10-1---- 01100 Bo
-0111--1 01000 Aa
-0111-1- 01000 Ac
-01111-- 01000 Ae
01-111-- 01010 BD
-110---- 01010 Bk
0-011111 00010 K
01-11-11 01010 Af
0111---- 01011 Br
00110--- 00111 BP
-001---- 00100 BJ
111-0000 01111 Ay
-01100-- 00101 At
-0110-0- 00101 Au
-0110--0 00101 Av
10111--- 01110 Bb
0011-000 00111 AB
1111---1 10000 Ah
1111--1- 10000 Ai
1111-1-- 10000 Aj
11111--- 10000 Ak
011-1111 01011 Al
1011-111 01110 An
01001--1 01001 AL
010-1010 01001 g
0001-101 00101 c
00011--0 00101 AE
0--1011- 00001 AV
1-0111-1 01101 BF
1-01111- 01101 BG
-101100- 01001 AO
01-10--- 01001 BU
1101-1-- 01111 Bs
1101--11 01111 Bc
11011--- 01111 Bt
```

Tulemus 5 Lähteülesanne 2 QMMinPro

```
00-001-- 00010 r
000-0111 00001 H
1-10---- 01101 Bs
10000-00 01011 z
0010---- 00110 BM
000010-- 00011 u
-0000001 00001 A
-0000010 00001 B
-0000011 00010 F
00-01100 00011 Z
-00-11-1 00100 AJ
-00-111- 00100 AK
010011-- 01001 AU
100000-- 01011 Ap
000110-- 00101 AE
00101-11 00111 Ad
001011-- 00111 Az
110----- 01110 Bu
01------ 01000 Bd
10--1--- 01100 Bm
10---1-1 01100 BX
10---11- 01100 BY
10-1---- 01100 Bn
-0111--1 01000 AW
-0111-1- 01000 AX
-01111-- 01000 AY
01-111-- 01010 BC
-110---- 01010 Bh
0-011111 00010 J
01-11-11 01010 Af
0111---- 01011 Bp
00110--- 00111 BO
-001---- 00100 BG
111-0000 01111 BA
-01100-- 00101 As
-0110-0- 00101 Ar
-0110--0 00101 Aq
10111--- 01110 BZ
0011-000 00111 AF
1111---1 10000 Ag
1111--1- 10000 Ah
1111-1-- 10000 Ai
11111--- 10000 Aj
011-1111 01011 Al
1011-111 01110 Am
01001-1- 01001 AT
010-1001 01001 f
0--101-1 00001 AO
1-01111- 01101 BE
0--1011- 00001 AP
00011--0 00101 AC
-0011101 00101 h
11011--- 01111 Br
01-1-0-0 01001 At
01-10--- 01001 BP
-10101-- 01001 Ay
-1010-11 01001 AZ
```

Tulemus 6 Lähteülesanne 2 Espresso

```
0-011111 00010
-0000011 00010
-0110111 00010
0000-001 00001
0000-010 00001
00-01011 00001
111-0000 01111
10000-00 00011
000-0111 00001
0001-101 00001
0001-110 00001
00001-00 00011
0-0-1001 00001
0011-000 00111
--101111 00001
-1011-11 00010
-1010-11 00001
00-010-- 00010
01001-1- 00001
010011-- 00001
1111---1 10000
1111--1- 10000
1-0111-1 00001
1-01111- 00001
1111-1-- 10000
-01011-- 00001
100000-- 00011
11111--- 10000
00011--0 00001
01-1-0-0 00001
00-001-- 00010
-0111--1 01000
-0111-1- 01000
-01111-- 01000
-00-11-1 00100
-00-111- 00100
-10111-- 00010
-0110-0- 00001
-10101-- 00001
11011--- 00001
000-10-- 00001
-0110--0 00101
-01100-- 00101
10---1-1 00100
10---11- 00100
10111--- 00110
01-10--- 00001
0010---- 00110
00110--- 00111
-110---- 01010
0111---- 00011
1-0-1--- 00100
1-10---- 00101
110----- 01110
-001---- 00100
01------ 01000
10------ 01000
```

Lähteülesanne 3.

Tulemus 7 Lähteülesanne 3 Minu lahendus

```
-------0 00000001 Jn
-----001 00000110 Ir
----0-10 00001011 Jo
---0-011 00010000 DJ
---001-0 00010101 Ip
---00101 00011010 Gs
--0-0111 00100100 EJ
-----1-0 00000101 Jq
----0101 00001010 HB
-1000--- 01000000 Bx
---01010 00010011 Gt
--0010-- 00100000 Br
---01100 00011101 JA
--001-00 00101001 GN
------10 00000011 Jp
-0-0111- 01000100 EL
-0-01101 01000010 k
0-011-1- 10000000 AF
---1111- 00010100 IG
010-111- 10000100 Ao
010-1101 10000010 C
1000---- 10000000 CC
----1-00 00001001 Is
0-0111-- 10000000 AG
---11-01 00010010 Hh
--1111-- 00100000 FB
----10-1 00001000 HD
---1-00- 00010000 GY
-10010-- 01100000 DB
-10-1100 01001101 He
0101---- 10000000 DN
---100-0 00010001 GS
--0101-- 00100000 DL
-0010--- 01000000 Bs
-001100- 01111000 Gh
--010-11 00100000 AE
--01100- 00111000 Gv
10010--- 11000000 DG
10-1100- 10011000 EG
-0101--- 01000000 DM
0010---- 10000000 Bt
----1111 00001100 IE
---1-111 00010100 IF
-010-111 01000100 Ak
-1-11-1- 01000000 FC
--1-000- 00100000 Bw
0011000- 11110000 CK
-1111--- 01000000 FF
--1000-- 00100000 Bu
001100-0 11110001 Gb
011000-- 11100000 Ca
--11-010 00100011 Hf
-011000- 01110000 CU
-01100-0 01110001 Gj
1-11-1-- 10000000 FG
1-000--- 10000000 CE
11-010-- 10000000 AK
11-01-00 10001001 EI
1111---- 10000000 FJ
--111-1- 00100000 FA
-1-111-- 01000000 FD
-----11- 00000100 JK
1-11--11 10000000 BR
1-111-1- 10100000 Fp
------01 00000010 It
111--111 10000100 BW
-1100-0- 01100000 DE
0-100-0- 10100000 Bh
-111-1-- 01000000 FE
-11-0011 01000000 I
-1100--0 01100001 HA
0-100--0 10100001 GQ
--1-11-1 00100000 Ey
--10-110 00100111 Ja
--100-0- 00100000 Bv
111-1--- 10000000 FI
```

Tulemus 8 Lähteülesanne 3 QMMinPro

```
-------0 00000001 Jd
 -----001 00000110 Ih
 ----0-10 00001011 Jg
 ---0-011 00010000 CD
 ---001-0 00010101 Ik
 ---00101 00011010 GU
 --0-0111 00100100 DL
 -----1-0 00000101 Jf
 ----0101 00001010 GS
 -1000--- 01000000 Bo
 ---01010 00010011 GV
 --0010-- 00100000 BZ
 ---01100 00011101 Io
 --001-00 00101001 GM
 ------10 00000011 Je
 -0-0111- 01000100 DQ
 -0-01101 01000010 d
 0-011-1- 10000000 n
 ---1111- 00010100 Hd
 010-111- 10000100 Aa
 010-1101 10000010 D
 1000---- 10000000 CB
 ----1-00 00001001 Ii
 0-0111-- 10000000 o
 ---11-01 00010010 Gx
 --1111-- 00100000 EO
 ----10-1 00001000 GT
 ---1-00- 00010000 GH
 -10010-- 01100000 CV
 -10-1100 01001101 HF
 0101---- 10000000 Cl
 ---100-0 00010001 GJ
 --0101-- 00100000 CH
 -0010--- 01000000 Bh
 -001100- 01111000 Gi
 --010-11 00100000 a
 --01100- 00111000 Ga
 10010--- 11000000 DC
 10-1100- 10011000 Dz
 -0101--- 01000000 CM
 0010---- 10000000 Bs
 ----1111 00001100 Hb
 ---1-111 00010100 Hc
 -010-111 01000100 AK
 -1-11-1- 01000000 ES
 --1-000- 00100000 Bb
 0011000- 11110000 Ck
 -1111--- 01000000 EX
 --1000-- 00100000 Bd
 001100-0 11110001 Gp
 011000-- 11100000 Co
 --11-010 00100011 Gz
 -011000- 01110000 CP
 -01100-0 01110001 Gk
 1-11-1-- 10000000 Ek
 1-000--- 10000000 Bx
 11-010-- 10000000 AF
 11-01-00 10001001 ED
1111---- 10000000 FB
--111-1- 00100000 EN
-1-111-- 01000000 ET
-----11- 00000100 Il
1-11--11 10000000 BF
1-111-1- 10100000 FU
------01 00000010 Ig
111--111 10000100 BQ
-111--11 01000000 Aw
-11000-- 01100000 CZ
111-1--- 10000000 Ey
0-100-0- 10100000 Bp
0-100--0 10100001 GR
-1100-0- 01100000 CY
-111-1-- 01000000 EW
-1100--0 01100001 Gn
--1-11-1 00100000 EL
--10-110 00100111 JE
--100-0- 00100000 Bc
```

Tulemus 9 Lähteülesanne 3 Espresso

```
11-00111 10000000
001-00-0 10000000
010-11-1 10000000
10-10-11 10000000
001-000- 10000000
010-111- 10000000
10-1100- 10000000
11-01-00 10000000
-010-111 01000000
-11-0011 01000000
-001-00- 01000000
-100--00 01000000
--11-010 00100000
--0-0111 00100000
-0-100-0 01000000
--010-11 00100000
-10-101- 01000000
--001-00 00100000
--10-110 00100000
0-011-1- 10000000
-0-1000- 01000000
-0-011-1 01000000
0-100--0 10000000
0-0111-- 10000000
11-010-- 10000000
0-100-0- 10000000
-0-0111- 01000000
0-1000-- 10000000
----0101 00001000
---1-111 00010000
100-0--- 10000000
0101---- 10000000
1000---- 10000000
---0-100 00010000
---0-011 00010000
---11-01 00010000
0010---- 10000000
1111---- 10000000
---100-0 00010000
111-1--- 10000000
--0-100- 00100000
--1-11-1 00100000
-0010--- 01000000
---001-0 00010000
1-11-1-- 10000000
-100-0-- 01000000
--1-000- 00100000
---1111- 00010000
-111-1-- 01000000
-1-00--0 01000000
---0010- 00010000
---0101- 00010000
--0101-- 00100000
1-000--- 10000000
1-111--- 10000000
--111-1- 00100000
-1-00-0- 01000000
-1-111-- 01000000
-1000--- 01000000
-0101--- 01000000
-1111--- 01000000
--0010-- 00100000
--100-0- 00100000
--1111-- 00100000
--1000-- 00100000
-----001 00000100
----1-00 00001000
----0-10 00001000
----1-11 00001000
----10-1 00001000
---1-00- 00010000
------01 00000010
------10 00000010
-----1-0 00000100
-----11- 00000100
-------0 00000001
```

Lisa 3 – Programmi kood.

```
public static boolean isNumeric(String initial) {
		String[] parts = initial.split(" ");
	for (int i = 0; i < parts[0].length(); i++) {
		if ((parts[0].charAt(i) == '1') || (parts[0].charAt(i) == '0')){
			} else {
				return false;
			}
		}
	for (int i = 0; i < parts[1].length(); i++) {
		if ((parts[1].charAt(i) == '1') || (parts[1].charAt(i) == '0')
		|| (parts[1].charAt(i) == '-')){
			} else {
				return false;
			}
		}
		return true;
	}
```

Joonis 8. isNumeric() kood.

```
public static ArrayList<String> multiply(String initial) {
	ArrayList<String> seperated = new ArrayList<String>();
	String[] parts = initial.split(" ");
		for (int i = 0; i < parts[1].length(); i++) {
			if ((parts[1].charAt(i) == '1')) {
				String make = "";
				for (int j = 0; j < parts[1].length(); j++) {
					if (j == i) {
						make = make + "1";
					} else {
						make = make + "0";
					}
				}
				seperated.add("_" + " " + parts[0] + " " + make);
			}
			if ((parts[1].charAt(i) == '-')) {
				String make = "";
				for (int j = 0; j < parts[1].length(); j++) {
					if (j == i) {
						make = make + "1";
					} else {
						make = make + "0";
					}
				}
				seperated.add("*" + " " + parts[0] + " " + make);
			}
		}
		return seperated;
}
```

Joonis 9. multiply() kood

```
public void checkCoverage(JTextArea textArea, JTable table) {
	checkCoverageMk2(initialnumbers, implicants);
	textArea.append("\nProcessing prime Implicants: \n\n");
	primeProcess = primeProcess + "\nProcessing prime Implicants: \n\n";
	while (!initialnumbers.isEmpty()) {
		System.out.println();
		System.out.println("01 " + initialnumbers);
		String result = getPrimeMk2(implicants, initialnumbers,
		primeImplicants, table) + "\n";
		textArea.append(result);
		System.out.println(result);
		primeProcess = primeProcess + result;
	}
	System.out.println("01 " + primeImplicants);

}
```

Joonis 10. checkCoverage() kood

```
public HashSet<Todo> getCovers(String covered) {
	HashSet<Todo> coversLower = new HashSet<Todo>();
	HashSet<Todo> coversUpper = new HashSet<Todo>();
	HashSet<Todo> covers = new HashSet<Todo>();
	for (int i = 0; i < superGroup.size(); i++) {
		for (int j = 0; j < superGroup.get(i).size(); j++) {
			for (int k = 0; k < superGroup.get(i).get(j).size(); k++)
			{
				System.out.println("02 " +
				superGroup.get(i).get(j).get(k).toString()+"    "
				+ covered);
				if
				(superGroup.get(i).get(j).get(k).toString().equals(
				covered)) {
					coversLower.addAll(superGroup.get(i).get(j).
					get(k).getCoverage());
					coversLower.forEach((a) -> {
					coversLower.addAll(a.getCoverage());
					});
					coversUpper.addAll(superGroup.get(i).get(j).
					get(k).getCoveredBy());
					for (int n = 0; n < superGroup.size() - i;
					n++) {
					HashSet<Todo> coversUpperTemp = new
					HashSet<Todo>();
					coversUpper.forEach((a) -> {
					coversUpperTemp.addAll(a.getCoveredBy());
					});
					coversUpper.addAll(coversUpperTemp);
					}
					covers.add(superGroup.get(i).get(j).get(k));
					covers.addAll(coversLower);
					covers.addAll(coversUpper);
					return covers;
				}
			}
		}
	}
	return covers;
}
```

Joonis 11. getCovers() kood

```
public static ArrayList<ArrayList<Todo>> check(ArrayList<ArrayList<Todo>>
groups) {
	ArrayList<ArrayList<Todo>> newGroup =new ArrayList<ArrayList<Todo>>();
	for (int i = 0; i < groups.size() - 1; i++) {
		ArrayList<Todo> outputGroup =
		Checker.outputCheck(groups.get(i));
		ArrayList<Todo> functionGroup =
		Checker.functionCheck(groups.get(i), groups.get(i + 1));
		ArrayList<Todo> mergedGroup = new ArrayList<Todo>();
		mergedGroup.addAll(outputGroup);
		for (int j = 0; j < functionGroup.size(); j++) {
			boolean check = false;
			for (int k = 0; k < mergedGroup.size(); k++) {
				if
				((mergedGroup.get(k).toString().equals(functionGrou
				p.get(j).toString()))) {
					check = true;
				}
			}
			if (!check) mergedGroup.add(functionGroup.get(j));
		}
		newGroup.add(mergedGroup);
	}
	ArrayList<Todo> mergerGroup = new ArrayList<Todo>();
	if (groups.size() > 1) {
		mergerGroup = Checker.outputCheck(groups.get(groups.size() -
		1));
	}
	if (!(mergerGroup.isEmpty())) {
		newGroup.add(mergerGroup);
	}
	return newGroup;
}
```

Joonis 12. check() kood

```
public static String getPrimeMk2(ArrayList<Todo> implicants, ArrayList<Todo>
initialgroup, ArrayList<Todo> primeImplicants, JTable table) {
	String response = "None to be removed";
	ArrayList<Todo> toBeRemoved = new ArrayList<Todo>();
	if (!check2) {
		addCoversForAll(implicants, initialgroup);
	}
	check2 = true;
	for (int i = 0; i < implicants.size(); i++) {
		for (int j = 0; j < implicants.size(); j++) {
			if (!implicants.get(i).equals(implicants.get(j))) {
				if
				(implicants.get(i).getCoverageForAll().containsAll(
				implicants.get(j).getCoverageForAll())) {
					toBeRemoved.add(implicants.get(j));
				}
			}
		}
		if (!toBeRemoved.isEmpty()) {
			response = implicants.get(i).getIdentifier() +" covers ";

			for (int k = 0; k < toBeRemoved.size(); k++) {
```

```
                    response = response + " " +
                    toBeRemoved.get(k).getIdentifier();
                    removeColumn(table,
                    toBeRemoved.get(k).getIdentifier());
                    mplicants.remove(toBeRemoved.get(k));
             }return response;
      }
}
for (int k = 0; k < initialgroup.size(); k++) {
      int singularityCheck = 0;
      int checkPos = 0;
      for (int i = 0; i < implicants.size(); i++) {
             if
             (implicants.get(i).getCoverage().
             contains(initialgroup.get(k))) {
                    singularityCheck++;
                    checkPos = i;
             }
      }
      if (singularityCheck == 1) {
             response = implicants.get(checkPos).getIdentifier() + "
             is a prime Implicant";
             for (int j = 0; j < implicants.get(checkPos).
             getCoverage().size(); j++) {
                    removeRow(table,
                    implicants.get(checkPos).getCoverage().get(j).toStr
                    ing());}
             initialgroup.removeAll(implicants.
             get(checkPos).getCoverage());
             primeImplicants.add(implicants.get(checkPos));
             removeColumn(table,
             implicants.get(checkPos).getIdentifier());
             implicants.remove(checkPos);
             check2 = false;
             return response;
      }
}
Todo uncertainty;
if (implicants.size() < 3) {
      uncertainty = implicants.get(0);
} else {
      System.out.println("OG implicant group size:
      "+implicants.size());
      uncertainty = uncertaintySolver(implicants, implicants,
      initialgroup, primeImplicants);
}
response = uncertainty.getIdentifier() + " was chosen by the
uncertainty solver.";
for (int j = 0; j < uncertainty.getCoverage().size(); j++) {
      removeRow(table, uncertainty.getCoverage().get(j).toString());
}
initialgroup.removeAll(uncertainty.getCoverage());
primeImplicants.add(uncertainty);
removeColumn(table, uncertainty.getIdentifier());
implicants.remove(uncertainty);
check2 = false;
      return response;
}
```

Joonis 13. getPrime() kood