TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Anzelika Muravskaja 193414IADB

5-muutuja loogikafunktsiooni minimaalseid normaalkujusid ja loogikaavaldiste erikujusid leidva veebirakenduse programmeerimine Java keeles

Bakalaureusetöö

Juhendaja: Harri Lensen
Magistrikraad

Tallinn 2023

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: Anzelika Muravskaja

24.04.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on arendada veebipõhist tarkvaravahendit, mis genereerib erinevaid loogikaavaldisi 5-muutuja loogikafunktsioonide jaoks tudengi maatriklinumbri põhjal. Tööriist annab näiteid 5-muutujaliste loogiliste vormide ja nende kujude kohta, et aidata IT-süsteemide arenduse tudengitel loogilisi funktsioone ja nende vorme paremini mõista. Pakutud lahendus aitab lahendada probleemi, et 5-muutujaliste loogikafunktsioonide jaoks on vähe näiteid ning pakub laiemat valikut loogilisi avaldisi kui praegu saadaval. Projekti käigus analüüsitakse ka teistes Eesti ülikoolides kasutusel olevaid lahendusi ja veebipõhiseid tööriistu, mida tudengid loogiliste funktsioonidega töötamisel kasutada saavad.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 54 leheküljel, 5 peatükki, 36 joonist, 3 tabelit.

Abstract

Web Application Calculating Normal Form Expressions and Expansions for a 5-place Boolean Function

The aim of this bachelor's thesis is to develop a web-based software tool that generates various logical expressions for 5-variable logical functions based on a student's matrix number. The tool will provide examples of 5-variable logical forms and their shapes to assist IT-system development students in understanding logical functions and their forms. The proposed solution addresses the shortage of examples for 5-variable logical functions and provides a more comprehensive range of logical expressions than currently available. The project also includes an analysis of existing solutions used in other Estonian universities and web-based tools available to students working with logical functions.

The thesis is in Estonian and contains 54 pages of text, 5 chapters, 36 figures, 3 tables.

Lühendite ja mõistete sõnastik

Argumentvektor	Loogikamuutujate “väärtustekomplekt”, mis esitab funktsiooni igale üksikule muutujale omistavat väärtust “0” või “1”
C#	Programmeerimiskeel, mis on loodud mitmesuguste .NET Frameworkis töötavate rakenduste loomiseks
div	HTML-element, mida kasutatakse veebilehe sisu rühmitamiseks ja stiiliks
GET päring	HTTP-päringu tüüp, mida kasutatakse teabe toomiseks veebiserverist
HEX	*Hexadecimal* – 16 numbrisüsteem
HTML	*Hyper Text Markup Language* – märgistuskeel, mida kasutatakse veebilehtede ja rakenduste loomiseks
HTTP päringuid	Teabevahetus veebikliendi ja veebiserveri vahel *Hypertext Transfer Protocol* protokolli abil
Implikant	Iga loogikafunktsiooni ühtede või nullide piirkonna intervall
Integer	Numbriline andme tüüp Java keeles
Java	Erinevat tüüpi tarkvararakenduste arendamiseks kasutatav programmeerimiskeel
JavaScript	Skriptikeel, mida kasutatakse veebirakenduste loomiseks
Kahendsüsteem	Kahendarvusüsteem, mis kasutab ainult kahte numbrit (1 ja 0)
Klient-server mudel	Arvutustehnikas kasutatav mudel, mis jagab töötlemise kliendi- ja serveripoolsete komponentide vahel
Kontuur	Kindlate mõõtmetega ruutude gruppe Karnaugh’ kaardil
Liides	Objekti kujude kirjeldus
List	Andmestruktuur, mis sisaldab sama tüüpi elemente
MVC kontroller	*Model-View-Controller* – tarkvara kujundamisel kasutatav arhitektuurimuster
Magic string	Programmeerimisel kasutatav kõvakoodiga stringi väärtus, mida on sageli raske muuta või siluda
Map	Andmestruktuur, mis salvestab võtme-väärtuse paare, kus iga võti on kordumatu
Massiiv	Andmestruktuur, mis sisaldab elementide jada
Objektorienteeritud	Programmeerimisparadigma, mis rõhutab objektide ja klasside kasutamist koodi ja andmete struktureerimiseks
Otsepunkt	Ainulaadne võrguaadress või URL, mis identifitseerib sideseansi või süsteemi spetsiifilise funktsiooni asukoha või sihtkoha

PHP	Serveripoolne skriptikeel, mida tavaliselt kasutatakse veebiarenduseks
Pseudo-element	CSS-i valija, mis sihib HTML-i elemendi kindlat osa
REST API kontroller	*Representational State Transfer* – veebirakenduse komponent, mis käsitleb RESTful API päringuid ja vastuseid
ReadMe-fail	Fail, mis pakub teavet ja juhiseid koodi või rakenduse kasutamiseks
Regulaaravaldis	Programmeerimiskeeltes teksti sobitamiseks või sellega manipuleerimiseks kasutatav muster või märgijada
Skaleeritavus	Süsteemi või rakenduse võime tulla toime suurenenud töökoormuse või kasutajatega ilma jõudluse halvenemiseta
span	HTML-element, mida kasutatakse stiilide rakendamiseks elemendi konkreetsetele tekstiosadele
Spring	Raamistik Java-põhiste veebirakenduste loomiseks
String	Tähemärkide jada, mida kasutatakse teksti esitamiseks programmeerimiskeeltes
Teenus	Rakenduse komponent, mis täidab konkreetset ülesannet
Typescript	Javascripti superkomplekt, mis lisab keelele staatilise tippimise ja muid funktsioone
UML diagramm	*Unified Modeling Language diagram* – süsteemi või tarkvararakenduse visuaalne esitus
URL	Aadress, mida kasutatakse veebilehe või ressursi leidmiseks Internetis
JSON	*JavaScript Object Notation* – lihtsustatud andmevahetusvorming, mida kasutatakse andmete edastamiseks ja salvestamiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Iga TalTechi teaduskonna IT-süsteemide arendaja tudeng läbib diskreetse matemaatika ainekursuse. Enamik õpilasi õpib seda esimest korda, ainest õpitakse tundma loogilisi funktsioone ja nende kujusid. Iga õpilane saab ülesande, kus ta töötab 4 muutujaga funktsioonidega. Sisendandmed arvutatakse tudengi matriklinumbri põhjal.

Ülesannete lahendamisel võib õpilane vajada näiteid samadest lahendatud ülesannetest. Enamikes allikates on palju näiteid 3 ja 4 muutujaga funktsioonidest, kuid 5 muutujaga näiteid on vähe.

Probleemi pakutud lahenduseks on rakendada veebipõhine tarkvara, mis leiaks erinevaid loogikaavaldisi 5-muutuja loogikafunktsioonidele tudengi matriklinumbri põhjal. See annab õpilastele näiteid 5 muutujaga loogilistest vormidest ja vormide kujudest, mis lisaks funktsioonide ja nende vormi saamise näidetele aitab ka tudengil põhikursusest kaugemale jõuda.

Käesolevas lõputöös analüüsitakse ka üliõpilastele kättesaadavaid veebipõhiseid loogikafunktsioonidega töötamise viise ning teistes Eesti ülikoolides kasutatavaid lahendusi.

2 Metoodika

See lõputöö tehti autori poolt üksi, seega valiti koskmudel metoodikat, kus iga etapp tuleb lõpetada enne järgmise juurde liikumist.

Esimene samm oli analüüs, mis hõlmas olemasolevate lahenduste uurimist, mis on üldsusele kättesaadavad, ning suuremate Eesti ülikoolide diskreetse matemaatika õppejõudude küsitluse läbiviimist.

Järgmine samm oli projekti välimuse ja struktuuri planeerimine, tehnoloogiate ja tööriistade valimine veebirakenduse arendamiseks.

Seejärel arendas autor projekti teenuse poolt ja kliendi poolt. Töös kirjeldatakse põhjalikult loodud lahenduse kliendi ja serveri poole struktuuri ning põhjendatakse kasutatud tehnoloogiate valikut.

Viimane samm oli paigaldamine, et projekt oleks avalik ja kättesaadav.

3 Ülevaade probleemist

Selles peatükis analüüsitakse olemasolevaid lahendusi. Saadud teabe põhjal kirjeldatakse uue lahenduse skoop.

3.1 Eksisteerivad lahendused

Selles peatükis kirjeldatakse olemasolevaid lahendusi, mida võiks kasutada. Toovad välja põhjused, miks nende kasutamine on üliõpilaste jaoks keeruline. Samuti esitatakse Eesti suuremate ülikoolide diskreetse matemaatika õppejõudude küsitluse tulemused.

3.1.1 Veebipõhised kalkulaatorid

Internetis on saadaval palju kalkulaatoreid, mis leiavad 3, 4 ja 5 muutujaga loogikafunktsiooni minimaalseid normaalkujusid ja loogikaavaldiste erikujusid. Sisendandmed koosnevad enamasti funktsioonidest, mille alusel saadakse nende vormid.

Kuigi veebikalkulaatoreid on mugav ja lihtne kasutada, võib õpilasel tekkida mitmeid probleeme:

- Pole olemas üht kalkulaatorit, mis annaks kõik diskreetse matemaatika aines vajalikud funktsioonide vormid. Õpilane peab kasutama palju erinevaid kalkulaatoreid.
- Palju aega kulub sobiva kalkulaatori otsimisele.
- Erinevatel kalkulaatoritel on erinevat tüüpi sisendandmed. Funktsioonides saab muutujaid tähistada erinevalt (A, B, C… või X1, X2, X3…). Nulle ja ühtesid saab tähistada tõese ja valega. Loogilisi tehteid saab tähistada sümbolite (“&”, “|”, “⊕” …) või sõnadega (“ja”, “või”, “xor” …).

3.1.2 Avatud lähtekood

Rohkemate näidete jaoks on võimalik kasutada koode, mis asuvad avatud hoidlates ja on üsna kergesti kättesaadavad.

On olemas palju avatud hoidlaid, mille koodid lahendavad tekkinud probleeme, kuid neil on palju puudusi:

- Puudub kood, mis annaks funktsiooni kõik vajalikud vormid. Peab kasutama palju erinevaid.
- Õige koodi otsimisele ja selle käivitamisele kulub palju aega, kuna kõikides repositooriumides pole ReadMe-faili.
- Paljud algoritmid on piiratud (mõni ei saa töötada pikkade funktsioonidega, mõni lahendab vaid osa probleemist).

3.1.3 Teised organisatsioonid

Korraldati küsitlus kahe Eesti suurema ülikooli seas, kus õpetatakse diskreetset matemaatikat, et uurida, kas väljatöötatud lahendus oleks ka teistes haridusasutustes kasulik. Küsiti Tartu ja Tallinna ülikoolide õppejõududelt järgmisi küsimusi:

- Kui palju muutujatega funktsioone on teie õppematerjalides saadaval ja millised neist esinevad kõige sagedamini?
- Kui võtate ülesandeid, siis milliseid näiteid kasutate 5 muutujaga?
- Kas annate õpilastele veebikalkulaatorite linke, et nad saaksid täiendavaid näiteid probleemide lahendamiseks?
- Kas igale õpilasele antakse individuaalseid ülesandeid?

Analüüsides erinevaid vastuseid, võib järeldada, et lähenemine muutujatega funktsioonidele võib õppeasutuste vahel erineda.

Tallinna Ülikooli õppematerjalides leidub tavaliselt funktsioone, mis sisaldavad 2-3 muutujat. Nelja muutujaga funktsioonide kasutamine piirdub Karnaugh'i kaartide abil minimaalsete kanoniseeritud vormide leidmisega. Kahjuks ei ole kättesaadavad funktsioonide näited, mis sisaldavad 5 muutujat. Veebikalkulaatoreid ei kasutata, kuna need pole praktilised 2-3 muutujaga funktsioonide jaoks ning õpilased saavad neid probleeme ise tõhusamalt lahendada. Loogikaprobleemide korral ei anta tavaliselt iga õpilase jaoks individuaalseid ülesandeid.

Tartu Ülikooli matemaatika ja statistika instituudi poolt õpetatavates ainetes määratletakse küll loogikafunktsioone, kuid neid ei käsitleta rohkem.

Kokkuvõttes võib öelda, et teistes ülikoolides ei pakuta tudengitele lahendusi, mis laiendaksid nende teadmisi ning programm, mis sisaldab 5 muutujaga ülesannete näiteid, oleks rakendatav ka teistes õppeasutustes. Loogikafunktsioonidega huvitatud tudengid saaksid rakendusega rohkem näiteid.

3.2 Uue lahenduse skoop

Uue lahenduse eesmärk on pakkuda veebipõhist tarkvaravahendit, mis võimaldab genereerida minimaalseid normaalkujusid ja loogikaavaldiste erikujusid 5-muutujaliste loogiliste funktsioonide jaoks, lähtudes üliõpilase matriklinumbrist. Selle tööriistaga saavad TalTechi diskreetse matemaatika esimest kursust läbivad IT-süsteemide arenduse tudengid paremini mõista loogilisi funktsioone ja nende vorme. Tarkvara näitab 5-muutujaliste loogiliste funktsioonide näidete ja nende kujude abil, kuidas tudengid saavad edasi liikuda põhikursusest. Uus lahendus eemaldab olemasolevate lahenduste puuduseid, nagu vajadust kasutada mitut kalkulaatorit või otsida sobivat koodi avatud lähtekoodiga hoidlatest. See pakub ka ulatuslikumat valikut loogilisi avaldisi kui praegu saadaval, parandades seeläbi TalTechi IT-süsteemide arenduse tudengite hariduse kvaliteeti. Lahendust võib potentsiaalselt kasutada ka teistes õppeasutustes.

4 Lahenduse analüüs

Selles peatükis kirjeldatakse nõudeid, mis on esitatud arendatava tarkvara suhtes, ning valitud tööriistu nende nõuete täitmiseks iga osa rakenduse jaoks.

4.1 Nõuete määramine

Lõputöö nõudeks oleks disainida ja arendada veebipõhine tarkvara, mis genereerib 5-muutuja loogikafunktsiooni minimaalseid normaalkujusid ja loogikaavaldiste erikujusid. Tarkvara peab olema võimeline võtma tudengi matriklinumbri sisendseadmeks ning pakkuma näiteid 5-muutuja loogiliste avaldiste ja nende vormide kohta.

Tarkvara peab lahendama tudengite probleeme, mis tekivad olemasolevate lahenduste kasutamisel, nagu vajadus kasutada mitut kalkulaatorit kõigi vajalike vormide näidete saamiseks, kulutades aega õige kalkulaatori ja näidete allikate leidmisele.

Tarkvara peab olema lihtne kasutada ja pakkuma tudengitele kõiki funktsioone, mida nad vajavad oma kursusetööde lõpetamiseks. Lisaks peaks tarkvara suutma töödelda pikki funktsioone ja omama kasutajasõbralikku liidest.

4.1.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded:

- Rakenduses peab olema aken matriklinumbri sisestamiseks.
- Matriklinumbri sisestamise aken peab olema punane, kui sisestatud andmed ei sobi.
- Avalehel peaks olema nupp, mis saadab päringu koos matriklinumbriga.
- Kui kasutaja vajutab “arvuta” nuppu ja väli on tühi, siis kuvatakse veateade “Tudenginumber on nõutav”.
- Kui kasutaja vajutab “arvuta” nuppu ja viis viimast numbrit on nullid, siis kuvatakse veateade “Tudenginumbri 5 viimast numbrit ei saa olla nullid”.
- Kui kasutaja vajutab “arvuta” nuppu ja matriklinumber on lühem, kui viis sümbolit, siis kuvatakse veateade “Tudenginumber on liiga lühike”.

- Kui kasutaja sisestas korrektse matriklinumbri, ja vajutas “enter” klahvi või “arvuta” nuppu, siis saadetakse päring.
- Andmete laadimisel kuvatakse laadimise animatsioon.
- Veebirakendus peab pakkuma loogikafunktsiooni normaalkujude ja loogikaavaldiste erikujude loendit tabelite ja valemite kujul, lähtudes tudengi matriklinumbrist.
- Iga ülesanne peab algama pealkirjaga.
- Igal tabelil peab olema pealkiri.
- Iga jääkfunktsioon peab olema esitatud tõeväärtustabelina.
- Argumentvektorid kleepimistabelis peavad olema esitatud kahendkujul.
- Valitud implikandid kleepimistabelis peavad olema märgistatud “A” tähega koos järjekorranumbriga.
- Katmistabelis valitud implikandid ja nende positsioonid peavad olema märgistatud punase värviga.
- Karnaugh’ kaart peab olema koostatud iga funktsiooni normaalkujule.
- Karnaugh’ kaardi all peab olema legend.
- Kontuurid Karnaugh’ kaardil, mis asuvad ainult ühel tasandil peavad olema punased ja need, mis asuvad mõlemal tasandil, peavad olema sinised.
- Iga jääkfunktsioon Shannoni arenduses peab alustama uuest joonest.
- Iga jääkfunktsiooni ülesande jääkfunktsioon peab olema kuvatud tõeväärtustabelina.

4.1.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded:

- Veebirakendus peaks olema kujundatud kasutajate jaoks lihtsasti kasutatavaks ja intuitiivseks.
- Veebirakendus peaks olema sobiv kasutamiseks töölaual.
- Veebirakendus peaks olema paigaldatud ja juurdepääsetav.
- Veebirakendus peaks kiiresti laadima ja reageerima kasutaja toimingutele ilma märkimisväärse viivituseta.
- Veebirakendus peaks olema lokaliseeritud eesti keele kasutajate jaoks.
- Kui ühe matriklinumbri asemel pannakse teine ja vajutatakse “arvuta” nuppu või “enter” klahvi, siis andmed peavad uuendama.

4.2 Tehnoloogia valik

Rakendus koosneb kliendi ja teenuse poolest. Andmebaasi kasutamine pole vajalik, seega võiks piirduda ainult kliendi poolega, kuid autor leidis, et on vajalik teha arvutusi teenuse poolel, sest arvutuste tegemine kliendi poolel võib kasutaja seadmele tarbetult palju ressursse kulutada ja aeglustada rakenduse tööd. Seetõttu valis autor erinevad tehnoloogiad kasutamiseks rakenduse esimese ja teise poole jaoks.

4.2.1 Programmeerimiskeele valik teenusepool

Autor omab kogemusi PHP, Java ja C# keelega töötamisega ja valis nende vahel.

Programmeerimiskeele valimisel lähtus autor järgnevatest kriteeriumidest:

- Kas keel on objektorienteeritud.
- Kas see sobib teenusepoole kirjutamiseks.
- Kui hästi autor keelt valdab.

Serveripoolse keele kiirust ei kaalutud, sest keerulisi päringuid suurtele andmebaasidele ei ole vaja teha, vaid toimuvad ainult arvutused, mis ei sisalda pikki tsükleid.

Kõige paremini sobivad Java ja C#, kuna need on objektorienteeritud ja neis on tugev tüübitus, mida PHP-s ei ole. Tugev tüübitus on oluline selle ülesande täitmiseks, kuna see võimaldab vähem vigu koodi kirjutamisel.

Ülesande täitmiseks valiti Java, kuna autoril on selles keeles rohkem kogemusi ja ta plaanib oma Java-teadmisi laiendada.

4.2.2 Programmeerimiskeele valik kliendipool

Programmeerimiskeele valik kliendi poolel oli oluliselt kitsam. Autor oli tuttav ainult JavaScript ja TypeScript -iga. Nendest valiti TypeScript tänu staatilise tüpiseerimise toetusele, mis muudab koodi arusaadavamaks ja vähendab vigade hulka.

4.2.3 Tööriistade valik teenusepool

Raamistiku valik lähtus valitud programmeerimiskeelest. Java raamistikudest oli autorile tuttav ainult Spring, seetõttu valiti see.

Projekti sõltuvuste haldamiseks valiti tööriist, mida saab kasutada mistahes Java-põhiste projektide koostamiseks ja haldamiseks [1] – Maven, kuna see on lihtsam ja selles

väikeses projektis ei ole vaja võimsaid ja paindlikke funktsioone, mis annab Gradle – ehitustööriist, mis on piisavalt paindlik, et ehitada peaaegu igat tüüpi tarkvara [2].

4.2.4 Tööristade valik kliendipool

TypeScript-i toetavad ja autorile tuttavad raamistikud on järgmised:

- Vue: JavaScript-i raamistik kasutajaliideste loomiseks. [1]
- Aurelia: kogumik kaasaegseid JavaScripti mooduleid, mis koos kasutades toimivad võimsa platvormina brauseri, töölaua- ja mobiilirakenduste loomiseks. [2]
- React: raamatukogu veebi- ja mobiilirakenduste kasutajaliideste loomiseks. [3]
- Angular: platvorm ja raamistik üheleheküljeliste klientrakenduste loomiseks HTML-i ja TypeScripti kasutades. [4]

Arenduseks valiti Angular, kuna autoril on suurem kogemus selle raamistikuga ja ta on tuttav Angular Material-iga, mis võimaldab luua ilusama rakenduse.

Samuti kasutati kliendi poole väljatöötamiseks Bootstrap-i. See on kliendipoolse arendamise raamistik veebilehe loomisele [5]. Selle kasutamine lihtsustas ja kiirendas arendamist.

4.2.5 Paigaldamine

Paigaldamisele oli valitud Microsoft Azure Web Service, sest see võimaldab luua, hallata ja arendada oma veebirakendusi.. Lisaks, autor oli kokku puutunud sellega ja Azure on väga lihtsalt ja mugavalt kasutatav ning projekti osasid saab kiiresti muuta ja ümberpaigaldada.

5 Lahenduse teenusepoolne arendus

Selles peatükis kirjeldatakse diskreetsete matemaatiliste probleemide lahenduse rakendamist, objektorienteeritud arhitektuuri serveripoolset osa ja selle arendamise etappe. Samuti esitatakse erinevate komponentide tüübid koos nende kirjeldustega.

5.1 Tulemus

Klassi CalculatorResponse objekt sisaldab arvutuste tulemusi, mis edastatakse kliendi poolele.

```
public class CalculatorResponse {

    private TruthTableResponse        truthTableResponse;

    private McCluskeyFunctionResponse
                disjunctiveMcCluskeyFunctionResponse;
    private McCluskeyFunctionResponse
                conjunctiveMcCluskeyFunctionResponse;

    private ShannonResponse           shannonResponseX1X3X5;
    private ShannonResponse           shannonResponseX2X4;
    private ReducedFunctionResponse   reducedFunctionResponseX2Zero;
    private ReducedFunctionResponse   reducedFunctionResponseX4One;
    private ReducedFunctionResponse   reducedFunctionResponseX1One;
    private ReducedFunctionResponse   reducedFunctionResponseX3Zero;

}
```

Joonis 1 CalculatorResponse klass.

5.1.1 TruthTableResponse klass

TruthTableResponse hoiab teavet nullide ja ühtede piirkondade kohta. Klass võtab sisendiks TruthTable klassi objekti (Tõeväärtus tabel), millest initsialiseeritakse TruthTableResponse objekti vastavad positsioonid.

```
public class TruthTableResponse {
    private List<Integer> zeroPositions;
    private List<Integer> onePositions;

    public TruthTableResponse(TruthTable truthTable) {
        this.zeroPositions = truthTable.getZeroArea().getPositions();
        this.onePositions = truthTable.getOneArea().getPositions();
    }}
```

Joonis 2 TruthTableResponse klass.

5.1.2 McCluskeyFunctionResponse klass

McCluskeyFunctionResponse on nii nimetatud sellepärast, et klass hoiab infot McCluskey' minimiseerimis meetodi (kleepimistabel ja katmistabel) ning funktsioonide (minimaalne, täielik, taandatud normaal kujud) kohta.

Sisendandmed on McCluskey' klassi objekt (McCluskey' meetod), millest võetakse vajalikud andmed, vajadusel need genereeritakse abimeetodiga.

```
public class McCluskeyFunctionResponse {
    private Map<Integer, Map<Integer, List<String>>>
customTabulationData;

    private Map<String, List<Integer>> primaryImplicantPositionsByBinaryValues;
    private FunctionFormData fullNormalFormData;
    private FunctionFormData reducedNormalFormData;
    private FunctionFormData minimalNormalFormData;

    public McCluskeyFunctionResponse(McCluskey mcCluskey) {
        Map<Integer, Map<Integer, List<Implicant>>> tabulationData =
mcCluskey.getTabulationData();
        this.customTabulationData = getCustomTabulationData(tabulationData);

        CoverageTable coverageTable = mcCluskey.getCoverageTable();
        this.PrimaryImplicantPositionsByBinaryValues =
getPrimaryImplicantsPositions(coverageTable.getPrimaryImplicants());

        fullNormalFormData = mcCluskey.getFullNormalFormData();
        reducedNormalFormData = mcCluskey.getReducedNormalFormData();
        minimalNormalFormData = mcCluskey.getMinimalNormalFormData();
    }
…
}
```

Joonis 3 McCluskeyFunctionResponse klass.

5.1.3 ShannonResponse klass

Klass ShannonResponse hoiab Shannoni arenduse infot. Klassi konstruktori sisendandmed on kaks funktsiooni: startFunction (funktsioon, millele tehakse Shannoni arendus) ja endFunction (Shannoni arenduse tulemus). Infot Function objektidest salvestatakse klassi väljadele.

```
public class ShannonResponse {
    String startFunctionString;
    String endFunctionString;
    Function startFunction;
    Function endFunction;

    public ShannonResponse(Function startFunction, Function
endFunction) {
        this.startFunctionString = startFunction.toString();
        this.endFunctionString = endFunction.toString();
        this.startFunction = startFunction;
        this.endFunction = endFunction;
    }
}
```

Joonis 4 ShannonResponse klass.

5.1.4 ReducedFunctionResponse klass

ReducedFunctionResponse objekt hoiab jääkfunktsiooni infot.

Nagu klassi ShannonResponse, ReducedFunctionResponse sisendandmed on kaks funktsiooni: startFunction (funktsioon, millele tehakse Shannoni arendus) ja endFunction (Shannoni arenduse tulemus). Lisaks, konstruktorisse tuleb veel tõeväärtustabel, mis oli genereeritud tulemuse funktsiooni põhjal. Infot Function objektidest salvestatakse klassi väljadele.

```
public class ReducedFunctionResponse {
    TruthTableResponse truthTableResponse;

    String startFunctionString;
    String endFunctionString;

    Function startFunction;
    FunctionPart endFunction;

    public ReducedFunctionResponse(TruthTable truthTable, Function
startFunction, FunctionPart endFunction) {

        this.truthTableResponse = new TruthTableResponse(truthTable);
        this.startFunctionString = startFunction.toString();
        this.endFunctionString = endFunction.toString();
        this.startFunction = startFunction;
        this.endFunction = endFunction;
    }
}
```

Joonis 5 ReducedFunctionResponse klass.

5.2 CalculatorController klass

See on Java klass, mis esindab kalkulaatori rakenduse REST API kontrollerit. See klass on Spring MVC kontroller, mis vastutab sissetulevate HTTP päringute töötlemise ja HTTP vastuste tagastamise eest ning on kättesaadav teistest domeenidest.

See sisaldab enda sees otspunkti "/calculator/calculate/{studentNumber}", kus "studentNumber" on teepunkti muutuja, mis saab dünaamiliselt URL-ist väärtuse. Antud teepunkt tagastab CalculatorResponse klassi objekti ja kasutab CalculatorService teenust, et saada antud objekt.

5.3 Teenuseid

Koodi mugavaks ja arusaadavaks korraldamiseks kasutati teenuseid. Igat tüüpi vastuse objektide genereerimiseks, mida on kirjeldatud eelmises peatükis, kasutatakse eraldi teenust.

5.3.1 CalculatorService klass

CalculatorService sisaldab endas teisi teenuseid (TruthTableService, ReducedFunctionService, McCluskeyFunctionService, ShannonFunctionService), mis on injekteeritud klassi sisse (Lisa 2).

Teenusel on ainult üks meetod nimega "getCalculations", mis võtab õpilase koodi String kujul ja tagastab CalculatorResponse-i. Meetodis toimub järgnev:

1. Tudengi matriklinumbri kontroll.
2. TruthTableResponse klassi genereerimine.
3. Disjunktiivse ja konjunktiivse McCluskeyFunctionResponse objekti genereerimine.
4. Kahe ShannonResponse klassi genereerimine muutujate X1, X3, X5 ja muutujate X2, X4 järgi.
5. Neli ReducedFunctionResponse klassi genereerimine, kus muutuja X2 on 0, X4 on 1, X1 on 1, X3 on 0.
6. Saavutatud andmete põhjal CalculatorResponse klassi objekti koostamine ja tagastamine.

5.3.2 TruthTableService klass

TruthTableService klassi sees on 3 privaatset välja:

- workingNumber – tudengi matriklinumber long kujul.
- onePositions – ühtede piirkonna positsioonid List kujul, kus iga positsioon on Integer objekt.
- zeroPositions – nullide piirkonna positsioonid List kujul, kus iga positsioon on Integer objekt.

On veel konstruktor, mis võtab tudengi numbri String kujul ja tagastab TruthTableResponse klassi objekti (TruthTableResponse klass). Konstruktor kasutab privaatseid abimeetode.

```
public class TruthTableService {
    private long workingNumber;
    private List<Integer> onePositions = new ArrayList<>();
    private List<Integer> zeroPositions = new ArrayList<>();

    public TruthTableResponse calculateTruthTable(String studentNumber) {
        Map<AreaName, Area> areas = getAreas(studentNumber);
        TruthTable truthTable =
new TruthTable(areas.get(AreaName.ZERO), areas.get(AreaName.ONE));

        return new TruthTableResponse(truthTable);
    }
…
}
```

Joonis 6 TruthTableService klass.

5.3.3 McCluskeyFunctionService klass

McCluskeyFunctionService klassis on ainult üks meetod calculateMcCluskeyFunction. Selle meetodi sisendandmed on piirkond (Piirkonnad) ja väljundandmed on McCluskeyFunctionResponse objekt (McCluskeyFunctionResponse klass).

```
public class McCluskeyFunctionService {

    public McCluskeyFunctionResponse calculateMcCluskeyFunction(Area area) {
        McCluskey mcCluskey= new McCluskey(area);

        return new McCluskeyFunctionResponse(mcCluskey);
    }}
```

Joonis 7 McCluskeyFunctionService klass.

5.3.4 ShannonFunctionService klass

ShannonFunctionService klassi sees on üks meetod – calculateDisjunctiveShannon (Lisa 3). See meetod võtab funktsiooni, millega tehakse Shannoni disjunktiivne arendus (see funktsioon on minimaalne disjunktiivne normaalkuju) ja muutujate nimed List kujul ValueName objektiga (Väärtus), mille järgi tehakse arendus. Meetodi tulemus on ShannonResponse objekt (ShannonResponse klass).

5.3.5 ReducedFunctionService klass

ReducedFunctionService klassi sees on meetod calculateReducedFunction ja abimeetod calculateEndFunction. Meetodi sisendandmed on funktsioon, mille jääkfunktsioon leitakse (see funktsioon on minimaalne disjunktiivne normaal kuju), muutuja nimi ja konstant (1 või 0), mille korral leiakse jääkfunktsioon. Meetod tagastab ReducedFunctionResponse klassi objekti (ReducedFunctionResponse klass).

```
public class ReducedFunctionService {

    public ReducedFunctionResponse calculateReducedFunction(Function function,
ValueName valueName, int constant) {
        Function startFunction = function.clone();
        startFunction.setNumberToValue(constant, valueName);
        FunctionPart endFunction = calculateEndFunction(startFunction);
        TruthTable truthTable = new TruthTable(endFunction);
        return new ReducedFunctionResponse(truthTable, startFunction,
endFunction);
    }

    private FunctionPart calculateEndFunction(Function input) {
        FunctionPart endFunction = calculateWithConstants(input);

        if (endFunction.isFunction()) {
            List<FunctionPart> newElements = new ArrayList<>();
            ((Function) endFunction).getElements().forEach(e ->
newElements.add(simplifyFunction(e)));
            ((Function) endFunction).setElements(newElements);
            endFunction = simplifyFunction(endFunction);
        }
        return endFunction;
    }
}
```

Joonis 8 ReducedFunctionService.

5.4 Abiklassid

5.4.1 Piirkonnad

Esiteks arvutatakse sisendandmetest (Tudengi matriklinumber – String kujul) tõeväärtuspiirkonnad. Lõpuprojektis kasutatakse 2 tüüpi piirkondi: nullide piirkond ja ühtede piirkond.

ga piirkonda esindab Area klass (Joonis 9 Area klass.), mis hoiab positsioonide loendi ja piirkonna nime AreaName kujul.

```
public class Area {
    private AreaName name;
    private List<Integer> positions;

    public void addPosition(int position){
        positions.add(position);
    }
}
```

Joonis 9 Area klass.

AreaName (Joonis 10 AreaName Enum) esindab piirkonna nime. Mugavuse huvides saab AreaName'ist saada arvväärtuse stringi kujul (“1” või “0”).

```
public enum AreaName {
 ONE("1"), ZERO("0");

 public final String Value;

 AreaName(String value) {
 this.Value = value;
 }
}
```

Joonis 10 AreaName Enum.

AreaName loodi selleks, et vältida “magic string” kasutamist ja vähendada koodiridu kohtades, kus on vaja saada väärtust AreaName objektist.

5.4.2 Piirkondade kalkuleerimine

Arvutused teeb AreasCalculator. See võtab matriklinumbri String kujul ja tagastab kaks piirkonda Map-na, kus võti on AreaName objekt ja väärtus on Area objekt.

Piirkondade saamise algoritm on järgmine:

1. Tudengikoodi 5 viimast numbrit teisendada HEX ja korrutada 7-ga kuni HEX-arv kasvab 7-kohaliseks arvuks.
2. Need HEX-numbrid on esimene grupp, mis läheb ühtede piirkonda.
3. Edasi jätkata 7-ga korrutamist kuni HEX-arv kasvab 8-järguliseks - ja igale HEXnumbrile (8 tk) 0.....F ehk 0....15 liita juurde 8 (misjärel nad kasvavad vahemikku 8....23)
4. Kõik need 8 liitmisel saadud kümnendarvud (8.....23) (8 tk) lisanduvad ühtede piirkonnale. Korduvaid võib ignoreerida.
5. Edasi jätkata 7-ga korrutamist kuni HEX-arv kasvab üheksajärguliseks - ja igale HEX numbrile (9 tk) 0.....F ehk 0....15 liita juurde 16 (misjärel nad kasvavad vahemikku 16....31)
6. Kõik need 16 liitmisel saadud kümnendarvud (16.....31) (9 tk) lisanduvad ühtede piirkonnale. Korduvaid ignoreeritakse.
7. Nüüd on ühtede piirkond selgunud - koosneb arvudest vahemikus 0....31.
8. Kõik need kümnendarvud vahemikus 0....31 mis pole 1-de piirkond - see on nullide piirkond.

Kuna töö tehakse 5 muutujaga, on piirkondade arvude (implikandi) koguarv 32 (0-st 31-ni kaasa arvatud). Implikandi arv arvutatakse kahendsüsteemi järgukaalude põhjal.

5.4.3 Implikant

Loogikafunktsiooni implikandiks nimetatakse igat tema 1-de või 0-de piirkonna intervalli. [8]

Implikant sisaldab järgmisi väljasid:

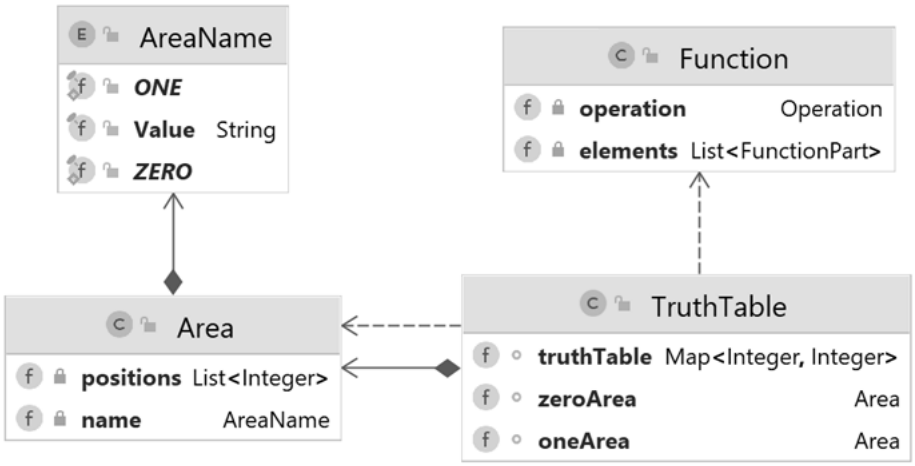
- binaryValue – numbriline element teisendatud 2-dal kujul 5 sümbolite pikkusega (n. “01000”).
- Peale kleepimist, binaryValue omandab mitte ainult nullid ja ühed, aga ka kriipsud mõnedel positsioonidel (n. “01--0”).
- coverage – implikandina kaetud liikmed List kujul.
- wasGlued – hoiab infot, kas implikant oli kleebitud, või mitte.
- isPrimary – hoiab infot, kas see implikant on lihtimplikant.

Implikant võib olla kahendarvu kujul, kuid võib saada listi kümnendarvudega, mis on implikandi kattuvuse positsioonid.

Lisaks, implikanti ei kasutata piirkondades või tõeväärtuste tabelis, sest siin ei toimu mingeid tegevusi implikantidega. Aga seda kasutatakse laialdaselt allpool kirjeldatud McCluskey' meetodis.

5.4.4 Tõeväärtus tabel

Tõeväärtustabel (Joonis 11) koostatakse leitud piirkondade põhjal ja hoiab need sees. TruthTable-i saab luua mitte ainult piirkondadest, vaid ka funktsiooni põhjal (Funktsioon).



Joonis 11 TruthTable UML diagramm.

Tabel saadakse funktsiooni väärtuste kõiki võimalikke tõeväärtuste kombinatsioone arvesse võttes. Iga kombinatsioon kuvatakse tabeli ühel real [9].

5.4.5 McCluskey' meetod

Järgmisena edastatakse tõeväärtustabeli andmed McCluskey klassi McCluskey' meetodite rakendamiseks ja erinevate funktsiooni vormide leidmiseks.

Funktsioonide leidmine on võimalik ka Karnaugh' kaardi abil. See meetod polnud valitud sellepärast, et Karno kaartide rakendamine rohkem kui nelja muutujate funktsiooni vormide leidmiseks on ebamugav nii leidmiseks, kui ka lugemiseks. Lisaks, Karno kaardid on visuaalne tööriist ja selle eeldused saavad nähtavaks juhul, kui

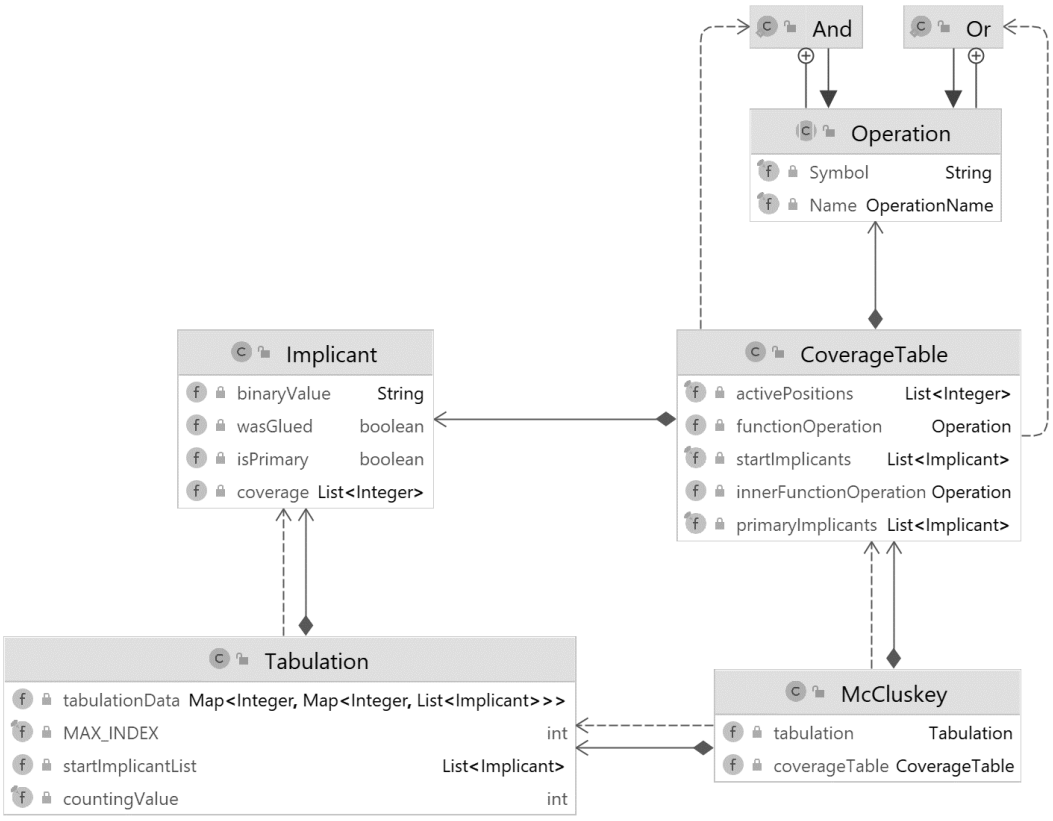
rakendatakse inimesega. Selle projekti jaoks oli vajalik automatiseeritav implikantide lihtsustamise meetod, ja selle kriteeriumi järgi McCluskey' meetod sobib kõige paremini, kuna seda võib kasutada piiramatult muutujate arvuga funktsiooni kujude leidmiseks [10].

Kuna minimeerimise toiming Karnaugh' kaardil põhineb sobivaimate kontuuride visuaalsel intuitiivsel valikul. Selline heuristiline valik on halvasti algoritmiseeritav ja seega ka halvasti arvutiprogrammina realiseeritav. Erinevalt Karnaugh' kaardiga minimeerimisest kujutab McCluskey' meetod endast minimeerimise algoritmi [8].

McCluskey' meetodi rakendamine tuleb välja ülalttoodud põhjuste alt.

5.4.6 McCluskey' meetodi rakendamine

McCluskey' klassi konstruktorisse edastatakse TruthTable klassi objekt (Tõeväärtus tabel), kus TruthTable klassis asuvalt väärtuste piirkondadest koostatakse disjunktsiooni või konjunktsiooni kleepimis- ja katmistabelid, mille jaoks oli ka loodud Tabulation ja CoverageTable klassid (Joonis 12).



Joonis 12 McCluskey UML diagramm.

Esimese sammuna grupeeritakse ühtede piirkond disjunktiivse kleepimistabel jaoks ja nullide piirkond konjunktiivse kleepimistabeli jaoks.

Tabel 1 McCluskey' meetodi disjunktiivse kleppimistabeli koostamise esimese sammu näidis 5 muutujaga.

index	1de pk
0	
1	00001 00100 10000
2	01100 00110 01001 10001 10010
3	01101 01110 01011 10101 11010 11001 10110
4	11101
5	11111

Kleepimistabeli andmed hoitakse Tabulation klassi objektis. Tabulation klassi objektid luuakse ühest väärtuste piirkondadest: konjunktiivne nulli piirkonnast, disjunktiivne ühtede piirkonnast. Väli countingValue hoiab andmeid tulenevalt sellest, kumb piirkondadest on töös. Selle välja põhjal määratakse indeks ja teostatakse alltoodud kleepimise protseduur.

Iga piirkonna numbrilistest elementidest luuakse Implicant klassi objekt (Implikant).

Teine samm on kleepimine. Implikandid, mis erinevad ainult ühe sümboliga, võivad olla kleebitud. Seega kaks implikandid, mida saab kleepida, erinevad täpselt ühe bittiga

bitti kujundamisel. Kui kaks implikanti kombineeritakse üheks, siis see sisaldab kahte bitilise kujundamist. Kahe bitti kujundamise korrutisel kasutatakse kriipsu, et märkida sümbolit, mis ei kordu [10].

Tabel 2 McCluskey' meetodi disjunktiivse kleepimistabeli näidis.

index	0de pk	M	2-sed interv	M	4-sed interv	M
0			0-001	K	--001	A9
1	00001	K	-0001	K	0-1-0	A10
	00100	K	0-100	K	-1-01	A11
	10000	K	001-0	K	1--01	A12
2	01100	K	1000-	A1		
	00110	K	100-0	A2		
	01001	K	0110-	A3		
	10001	K	011-0	K		
	10010	K	0-110	K		
3	01101	K	-0110	A4		
	01110	K	01-01	K		
	01011	K	010-1	A5		
	10101	K	-1001	K		
	11010	K	10-01	K		
	11001	K	1-001	K		
	10110	K	1-010	A6		
4	11101	K	10-10	A7		
5	11111	K				
			-1101	K		
			1-101	K		
			11-01	K		
			111-1	A8		

Kleepimise andmed salvestatakse TabulationData klassis, mis kujutab ennast Map (Map<Integer, Map<Integer, List<Implicant>>>), kus esimese võtmeks on intervall ja väärtuseks on Map, kus võtmeks on indeks ja väärtuseks on list selle indeksi implikantidega.

Järgmine etapp on katmistabeli koostamine. Selle jaoks valitakse kõik lihtimplikandid. Lihtimplikantideks nimetatakse täpselt intervalli ehk siis intervalle, mis ei sisaldu üheski veelgi suuremas selle funktsiooni intervallis [8].

McCluskey’ meetodi kleepimistabelis osutuvad lihtimplikantideks:

- Kõik viimase intervallide veeru intervallid (suurimad kleepimistabelis);
- Eelmiste veergude need intervallid, mis ei sisaldu selles kleepimistabelis kuskil suurema intervalli koosseisus. (näteks: mida ei kleebitud “edasi”) [8].

Katmistabeli esitamiseks on loodud klass CoverageTable, mis võtab sisse lihtimplikandid, implikandid, mis kujutavad endas piirkonna liikmeid (Tabel 3) ja konkreetset piirkonda (Piirkonnad).

Tabel 3 McCluskey’ meetodi disjunktiivse katmistabeli näidis.

A\piirkond	1	4	6	9	11	12	13	14	16	17	18	21	22	25	26	29	31
A1									X	X							
A2									X		X						
A3						X	X										
A4						X					X						
A5				X	X												
A6											X				X		
A7											X		X				
A8																X	X
A9	X			X						X				X			
A10		X	X			X		X									
A11				X			X							X		X	
A12										X		X		X		X	

Implikantidest ja nende kattuvusest saadakse funktsioone erinevate kujudega.

5.4.7 Operatsioonid

Loogilisi tehteid esindab abstraktne klass Operation. Selle klassi sees on And ja Or siseklassid, mis pärivad Operation klassi. And esindab konjunktsiooni ja Or esindab disjunktsiooni.

Igal toimingul on nimi ja sümbol, mille jaoks on loodud eraldi Enum.

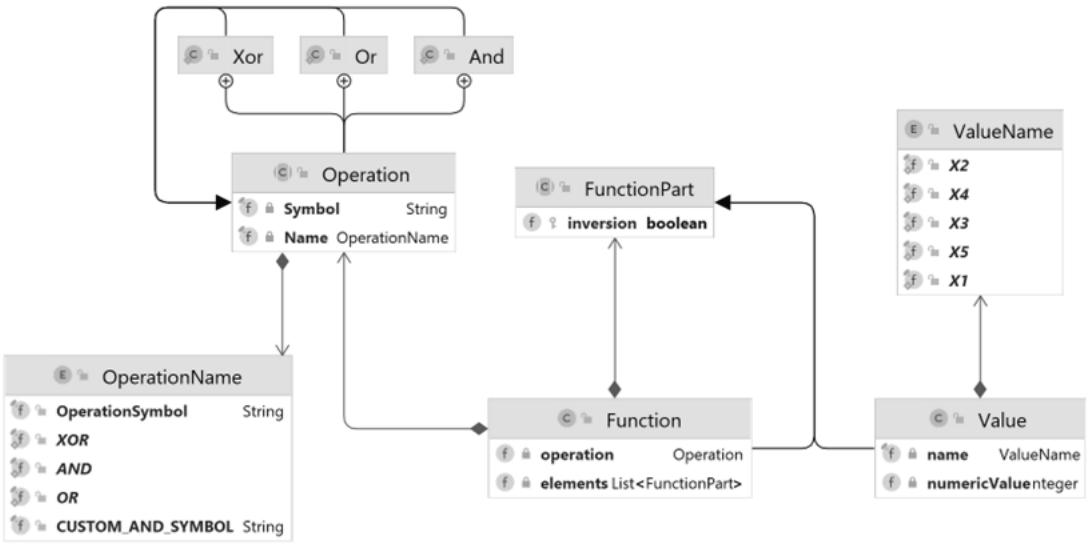
Mugavuse huvides on igal nimel operatsioonisümbol, mida kasutatakse funktsioonide visualiseerimiseks. Samuti, nagu näete (Joonis 13), on olemas CUSTOM_AND_SYMBOL, mida kasutatakse Function klassis toString meetodis siis, kui konjunktiivses funktsioonis on konstandid. See on visuaalse komponendi jaoks oluline, sest konstantide vahel ei saa olla tühja literaali ega tühikut.

```
public enum OperationName {
    AND(""), OR(" v "), XOR(" ⊕ ");

    public final String CUSTOM_AND_SYMBOL = " & ";

    public final String OperationSymbol;

    private OperationName(String symbol){
        this.OperationSymbol = symbol;
    }}
```

Joonis 13 OperationName enum.

5.4.8 Funktsioon

Funktsiooni kuvamiseks oli loodud klassid: Function, FunctionPart ja Value. Function ja Value on FunctionPart päritud klassid (Joonis 14). Seda pärandit kasutatakse seetõttu, et Function klassi elemendi listi sees võib sisalduda Value ning Function klassi objektid.



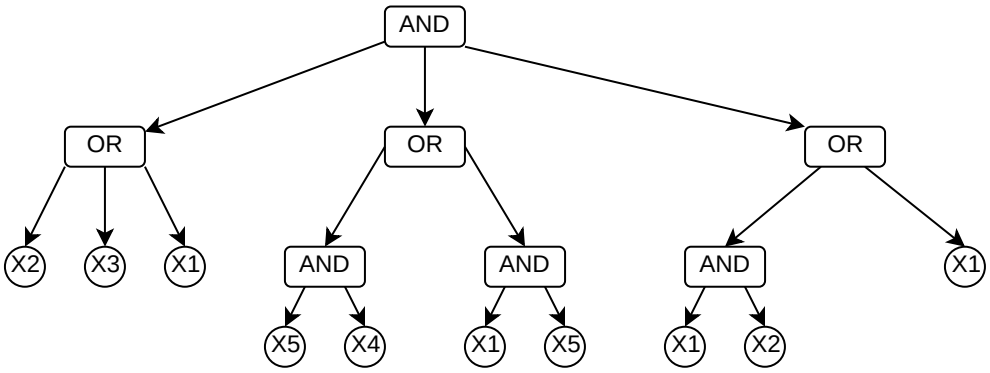
Joonis 14. Funktsiooni UML diagramm.

Funktsioon sisaldab välju:

- operation – selle funktsiooni loogikatehe (Operatsioonid), millest sõltub tehtemärk funktsiooni stringiks teisendamisel ja funktsiooni teisendus.
- elements – leht selles funktsioonis olevate elementidega.
- inversion – päritud FunctionPart klassist. Sisaldab teavet selle kohta, kas funktsioon on inverteeritud.

Funktsioon on puu, mille algus on funktsioon ja lõpp on väärtused.

Funktsioonipuu diagramm (Joonis 15) näitab, et esialgne AND-funktsioon sisaldab 3 alamfunktsiooni OR. Vasakpoolseim sisaldab 3 väärtust. OR-funktsioon sisaldab keskel teist AND-funktsiooni, mis sisaldab 2 väärtust. Parempoolseim funktsioon OR sisaldab ühte väärtust ja funktsiooni AND, mis sisaldab kahte väärtust.



Joonis 15 Funktsioonipuu diagramm.

Selline puu esindab funktsiooni: $(X_2 \text{ v } X_3 \text{ v } X_1)(X_5X_4 \text{ v } X_1X_5)(X_1X_2 \text{ v } X_1)$

5.4.9 Väärtus

Väärtust esindab Value klass. See sisaldab välju:

- name – väärtuse nimi. Kuna neid on piiratud arv, oli tehtud enum ValueName, mis on välja toodud allpool.
- numericValue – hoiab arv 1 või 0, kui väärtuse asemele pannakse konstant. Või hoiab null, kui mitte.
- inversion – päritud FunctionPart klassist. Sisaldab teavet selle kohta, kas väärtus on inverteeritud.

Kuna lõputöö on välja töötatud rangelt viie muutujaga, muutujate nimesid on viis: “X1”, “X2”, “X3”, “X4”, “X5”. Samuti sisaldab Enum meetodit getNameFromPosition, mis tagastab väärtuse nime sõltuvalt sisendnumbrist. See on vajalik implikandi muutmisel funktsiooni osaks.

```
public enum ValueName {
    X1, X2, X3, X4, X5;

    public static ValueName getNameFromPosition(int position) {
        return switch (position) {
            case 0 -> X1;
            case 1 -> X2;
            case 2 -> X3;
            case 3 -> X4;
            case 4 -> X5;
            default -> null;
        };
}}
```

Joonis 16 ValueName enum.

5.4.10 Seadused

Loodi Law klass, mis sisaldab staatilisi klassi: Absorption (Neeldumine) , Idempotency (Idempotentsus) ja Bonding (Kleepimine). Seadused kasutatakse Shannoni arenduse ja jääkfunktsiooni kalkuleerimisel.

Igas klassis on meetod nimega apply, mis võtab loogika funktsiooni ja tagastab modifitseeritud funktsiooni pärast seaduse rakendamist. Kui seadust pole võimalik rakendada, siis tagastatakse sisendfunktsioon.

5.5 Testimine

Kuna lõputöö põhieesmärgiks on arvutamine, siis on kõige olulisem testimine tagumise poole Unit testimine, mis tegeleb arvutustega. Seadused olid eriti hoolikalt testitud, kuna jääkfunktsioon ja Shannoni arendus sõltuvad neist täielikult ning seaduste rakendamist on raskem esitada koodi kujul, kui näiteks McCluskey’ minimiseerimis meetod või tõeväärtustabel.

Samuti on arenduse varajases staadiumis oluline testimise osa kontrollida, kas andmed on saadaval igale 5-kohalisele sisendväärtusele.

6 Kliendipoolne arendus

Selles peatükis kirjeldatakse kliendipoolse osa loomise protsessi ja esitatakse välja töötatud komponendid koos nende kirjeldustega.

6.1 Välimus

Veebirakendus koosneb ühest leheküljest, mille ülaosas asub sinine päis nimega “5-muutuja loogikafunktsiooni kujud”, mis on eraldatud halli joonega. Selle all on kirje “Sisesta oma matriklinumber”, sisestusväli matriklinumbri jaoks ja nupp “arvuta” (Joonis 17).

Joonis 17 Lehekülje välimus enne kalkuleerimist.

Kui matriklinumber on sisestatud, vajutab kasutaja “arvuta” nuppu ja pärast laadimise animatsiooni ilmuvad komponendid lahendusega (Joonis 18).

Joonis 18 Lehekülje välimus pärast kalkuleerimist.

6.2 Valideerimine

Väljale on võimalik sisestada ainult numbrilisi väärtusi. Matriklinumbri väljale sisestatud andmete kontrollimiseks tehakse järgmised kontrollid:

- Välja ei ole jäetud tühjaks.
- Matriklinumbri pikkus on vähemalt 5 tähemärki.
- Matriklinumbri viimased viis numbrit ei ole nullid.

Kui mõni kontrollidest ei lähe läbi, siis kuvatakse vastav teade: “Matriklinumber on nõutav”, “Matriklinumber on liiga lühike” või “Matriklinumbri 5 viimast numbrit ei saa olla nullid”. Seni kuni sisestatud number ei ole korrektne, ei saa päringut saata.

6.3 Enumid

Käesolevas töös kasutati Enumeid, sest need võimaldavad väärtuste jaoks kasutada kirjeldavaid nimesid, muutes koodi loetavamaks ja paremini mõistetavaks, selle asemel et väärtused koodi otse sisestada (mitte kasutada “magic string”).

6.3.1 Value enum

Enum Value oli kasutatud ühtede ja nullide esitamiseks. Lisaks kõigile enumi kasutamise eelistele, kirjeldatud üleval, see enum aitab aitab piirata väärtuste arvu nulli ja ühega.

```
export enum Value {
    One = 1,
    Zero = 0
}
```

Joonis 19 Value enum.

6.3.2 ImplicantType enum

ImplicantType enum sisaldab võimalikke implikantide tüüpe. Neid on kaks:

- detached – see, mis asub ainult ühel Karnaugh’ kaardi tasemel.
- shared – see, mis asub mõlemal Karnaugh’ kaardi tasandil.

```
export enum ImplicantType {
    detached,
    shared
}
```

Joonis 20 ImplicantType enum.

6.4 Liidesed

Liidesed esindavad vastust teenuse poolest. Valiti just nimelt liideseid, kuna on vaja ainult lugeda andmeid, mis tähendab, et objektide loomine ja meetodite kasutamine pole vajalik, on vaja ainult määrata objekti kuju.

6.4.1 CalculatorResponse liides

CalculatorResponse on teenuse poole vastuse kuju. See liides sisaldab mitmeid erinevaid omadusi, mis viitavad vastavatele andmestruktuuridele. Need omadused on järgmised:

- truthTableResponse – tõeväärtustabeli vastus TruthTableResponse kujul (TruthTableResponse liides).
- conjunctiveMcCluskeyFunctionResponse – konjunktivse McCluskey' minimiseerimis meetodi ja funktsiooni kujude vastus McCluskeyFunctionResponse kujul (McCluskeyFunctionResponse liides).
- disjunctiveMcCluskeyFunctionResponse – disjunktivse McCluskey' minimiseerimis meetodi ja funktsiooni kujude vastus McCluskeyFunctionResponse kujul (McCluskeyFunctionResponse liides).
- shannonResponseX1X3X5 – Shannoni funktsiooni X1, X3 ja X5 järgi vastus ShannonResponse kujul (ShannonResponse liides).
- shannonResponseX2X4 – Shannoni funktsiooni X2 ja X4 järgi vastus ShannonResponse kujul (ShannonResponse liides).
- reducedFunctionResponseX2Zero – vähendatud funktsiooni vastus, kus X2 väärtus on 0 ReducedFunctionResponse kujul (ReducedFunctionResponse liides).
- reducedFunctionResponseX4One – vähendatud funktsiooni vastus, kus X4 väärtus on 1 ReducedFunctionResponse kujul(ReducedFunctionResponse liides).
- reducedFunctionResponseX1One – vähendatud funktsiooni vastus, kus X1 väärtus on 1 ReducedFunctionResponse kujul (ReducedFunctionResponse liides).
- reducedFunctionResponseX3Zero – vähendatud funktsiooni vastus, kus X3 väärtus on 0 ReducedFunctionResponse kujul (ReducedFunctionResponse liides).

```
export interface CalculatorResponse {
    truthTableResponse: TruthTableResponse
    conjunctiveMcCluskeyFunctionResponse: McCluskeyFunctionResponse
    disjunctiveMcCluskeyFunctionResponse: McCluskeyFunctionResponse
    shannonResponseX1X3X5: ShannonResponse
    shannonResponseX2X4: ShannonResponse
    reducedFunctionResponseX2Zero: ReducedFunctionResponse
    reducedFunctionResponseX4One: ReducedFunctionResponse
    reducedFunctionResponseX1One: ReducedFunctionResponse
    reducedFunctionResponseX3Zero: ReducedFunctionResponse
}
```

Joonis 21 CalculatorResponse liides.

6.4.2 TruthTableResponse liides

Liidest nimega TruthTableResponse kasutatakse tõeväärtustabeli kujundamiseks ja sisaldab kahte omadust:

- onePositions – massiiv, mis sisaldab ühtede positsioone tõeväärtustabelis numbrite kujul.
- zeroPositions – massiiv, mis sisaldab nullide positsioone tõeväärtustabelis numbrite kujul.

```
export interface TruthTableResponse {
    zeroPositions: number[]
    onePositions: number[]
}
```

Joonis 22 TruthTableResponse liides.

6.4.3 FunctionFormData liides

Liides nimega FunctionFormData kasutatakse funktsiooni vormi andmete hoidmiseks ning sisaldab kolme omadust:

- positions – kahe mõõtmelise massiivi omadus, kus hoitakse implikante numbrite massiivi kujul. Iga massiivi number on implikandi positsioon.
- implicants – implikantide list stringi kujul.
- functionString – funktsioon stringi kujul.

```
export interface FunctionFormData {
    positions: number[][]
    implicants: string[]
    functionString: string
}
```

Joonis 23 FunctionFormData liides.

6.4.4 McCluskeyFunctionResponse liides

Liidest nimega McCluskeyFunctionResponse kasutatakse McCluskey' minimiseerimismeetodi kujundamiseks ja sisaldab viit omadust:

- customTabulationData – sisaldab andmeid kleepimistabeli kohta Map kujul, kus võti on intervall ja väärtus on Map, kus võti on index ja väärtus on implikantide massiiv.
- primaryImplicantPositionsByBinaryValues – implikantide asukohad Map kujul, kus võti on implikant stringi kujul ja väärtus on massiiv tema positsiooniga.
- fullNormalFormData – täielik normaalne funktsiooni kuju FunctionFormData vormis (FunctionFormData liides).
- reducedNormalFormData – taandatud normaalne funktsiooni kuju FunctionFormData vormis (FunctionFormData liides).
- minimalNormalFormData – minimaalne normaalne funktsiooni kuju FunctionFormData vormis (FunctionFormData liides).

```
export interface McCluskeyFunctionResponse {
    customTabulationData: Map<number, Map<number, string[]>>
    primaryImplicantPositionsByBinaryValues: Map<String, number[]>
    fullNormalFormData: FunctionFormData
    reducedNormalFormData: FunctionFormData
    minimalNormalFormData: FunctionFormData
}
```

Joonis 24 McCluskeyFunctionResponse liides.

6.4.5 ReducedFunctionResponse liides

TypeScripti liides nimega ReducedFunctionResponse kasutatakse jääkfunksiooni kujundamiseks ja sisaldab kolme omadust:

- truthTableResponse – omadus, mis viitab TruthTableResponse tüübile (TruthTableResponse liides). See sisaldab tulemuse tõeväärtustabeli andmeid.
- startFunctionString – string tüüpi omadus, mis sisaldab algset funktsiooni.
- endFunctionString – string tüüpi omadus, mis sisaldab jääkfunktsiooni vastust.

```
export interface ReducedFunctionResponse {
    truthTableResponse: TruthTableResponse
    startFunctionString: string;
    endFunctionString: string }
```

Joonis 25 ReducedFunctionResponse liides.

6.4.6 ShannonResponse liides

Liides nimega ShannonResponse kasutatakse Shannoni arenduse kujundamiseks ja sisaldab kahte omadust:

- startFunctionString – string tüüpi omadus, mis sisaldab funktsiooni algset kirjeldust.
- endFunctionString – string tüüpi omadus, mis sisaldab Shannoni arenduse vastust.

```
export interface ShannonResponse {
    startFunctionString: string
    endFunctionString: string
}
```

Joonis 26 ShannonResponse liides.

6.5 Komponentid

See rakendus on jaotatud komponentideks, et struktureerida ja korraldada koodi. Komponentideks jagamine aitab vältida koodi kordamist ning muudab selle redigeerimise lihtsamaks, mis omakorda kiirendab koodi kirjutamist.

6.5.1 CalculationResultComponent komponent

CalculationResultComponent sisaldab allolevaid CalculatorResponse objektist renderdatud komponente, töö päiseid ja valemeid, mis esindavad töö tulemust.

Komponent võtab vastu CalculatorResponse objekti ja jagab selle muutujateks, mida edastatakse alamkomponentidele. Lisaks töötleb see TruthTableResponse (TruthTableResponse liides) andmeid ja esitab need massiivina numbritega, kus indeks tähistab positsiooni ja elemendi indeks tähistab ühte või nulli.

6.5.2 TruthTableComponent komponent

TruthTableComponent kujundab tõeväärtustabelit. Komponent võtab sisse tõeväärtustabeli numbrite massiivi kujul, mis on moodustatud ülemkomponendis, ning väärtuste loendi stringide massiivi kujul, mille järgi koostatakse tabel.

Tõeväärtustabeli suurus sõltub väärtuste kogust.

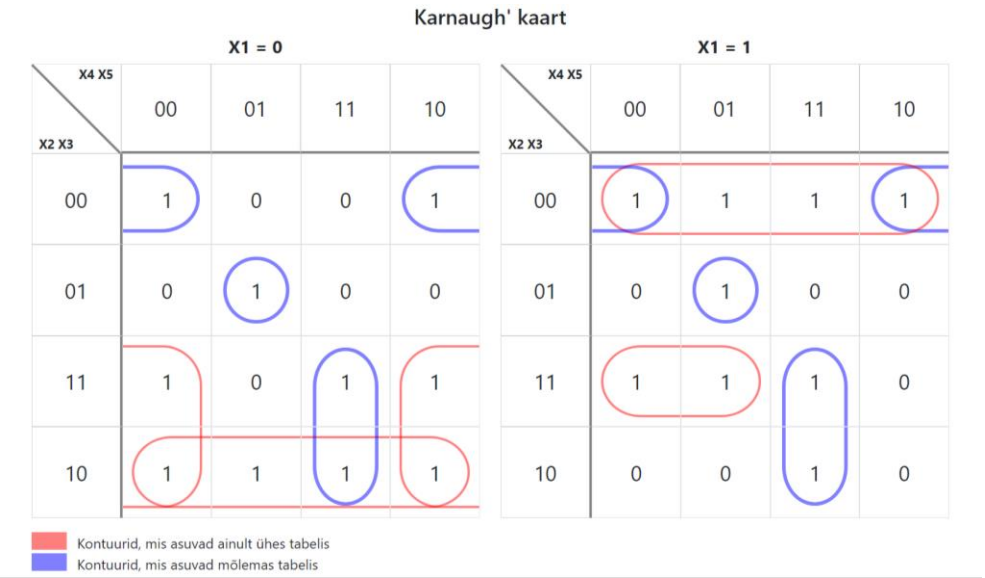
6.5.3 KarnaughMapComponent komponent

KarnaughMapComponent kujundab Karnaugh' kaarti (Lisa 4). 5-muutuja Karnaugh' kaart on kolmemõõtmeline, kaardil on 10 omavahel kattuvat piirkonda (ruutude gruppi), mis on määratud kaardi loogikamuutujate kõikvõimalike väärtustega nendes ruutudes [8].

Selles töös kasutajaliides näitab taandatud, täielikku ja minimaalset disjunktiivset ja konjunktiivset funktsioonide kujude leidmist.

Vasak pool on kaardi osa, kus X1 on 0 ja parem pool, kus X1 on 1. Iga kaardi osas ülemises vasakus nurgas on X2, X3, X4 ja X5 muutujad. Vertikaalses vasakus reas on X2 ja X3 väärtused ja horisontaalses ülemises veerus on X4 ja X5 väärtused. Tabeli teised ruudud on tõeväärtustabeli ümberpaigutus. Iga null ja üks ümberpaigutuse osas on esitatud Value kujul.

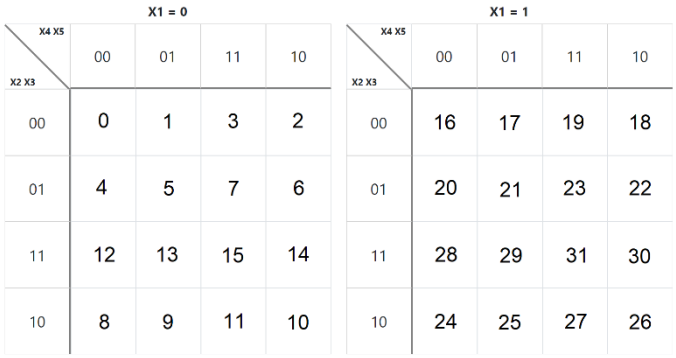
All vasakus nurgas on legend, kus on kirjeldatud kontuuride värvide tähendused. Punased on need, mis asuvad ainult ühes osas ja sinised on kontuurid, mis asuvad mõlemal poolel. See värvi erinevus aitab kasutajal paremini neid lugeda. Sellele aitab kaasa ka see, et sinistel kontuuridel on natuke väiksem raadius kui punastel, kuna tänu sellele need ei kattu üksteisega. Veel on omakorda sinisel kontuuril suurem piiri laius kui punastel.



Joonis 27 KarnauhgMap komponent.

Komponent võtab sisse:

- Kujundava funktsiooni FunctionFormData objekti kujul (FunctionFormData liides), kust võetakse implikandi infot.
- Kujundava funktsiooni nime stringi kujul.
- Tõeväärtustabel numbrilise massiivi kujul. Need numbrid pannakse oma positsioonile vastavalt argumentvektorile (Joonis 28).



Joonis 28 5-muutuja Karnaugh’ kaardi argumentvektorite paiknemine.

Komponendis on veel kaks KarnaughMapPartComponent komponenti (KarnaughMapPartComponent komponent), mida kirjeldatakse allpool. Komponendis määratakse muutujad alamkomponentidele:

- notInvertedPart - Karnaugh’ kaardi osa Value maatriksi kujul, kus X1 = 1.
- invertedPart - Karnaugh’ kaardi osa Value maatriksi kujul, kus X1 = 0.
- notInvertedImplicants - implikandid numbrilise maatriksi kujul, mis asuvad Karnaugh’ kaardi osas, kus kus X1 = 1 (nende hulgas ka need, mis asuvad mõlemal poolel).
- invertedImplicants - implikandid numbrilise maatriksi kujul, mis asuvad Karnaugh’ kaardi osas, kus X1 = 0 (nende hulgas ka need, mis asuvad mõlemal poolel).

6.5.4 KarnaughMapPartComponent komponent

KarnaughMapPart komponent kujundab kaardi vasakut või paremat poolt. Sisendandmed on:

- Kaardi osa numbriliste väärtuste maatriks, millele oli tehtud eraldi enumi Value (Value enum), kirjeldatud üleval.

- Value enumi objekt, mis määrab X1 väärtust.
- Implikandid, mis asuvad selles osas numbrite maatriksi kujul.
- Funktsiooni nimi, mida kasutatakse unikaalsete klasside nimede genereerimisel.

6.5.5 Kontuuri visualiseerimine Karnaugh' kaardil

Implikantide kuvamine sõltub elemendi klassist, mis esindab ühte ruutu kaardil. Ruut esitatakse div elemendina, mis sisaldab 13 span elementi.

```
<div id="{{functionName}}-cell{{matrix[y][x]}}"
            class="col border cell d-flex align-items-center justify-
content-center" *ngFor="let el of row; index as x">
            {{ el }}
            <span id="{{functionName}}-l-100-cell{{matrix[y][x]}}"
class="line l-line-100"></span>
            <span id="{{functionName}}-r-100-cell{{matrix[y][x]}}"
class="line r-line-100"></span>
            <span id="{{functionName}}-t-100-cell{{matrix[y][x]}}"
class="line t-line-100"></span>
            <span id="{{functionName}}-b-100-cell{{matrix[y][x]}}"
class="line b-line-100"></span>
            <span id="{{functionName}}-l-r-t-b-cell{{matrix[y][x]}}"
class="line l-r-t-b-line"></span>
            <span id="{{functionName}}-l-t-cell{{matrix[y][x]}}"
class="line l-t-corner"></span>
            <span id="{{functionName}}-r-t-cell{{matrix[y][x]}}"
class="line r-t-corner"></span>
            <span id="{{functionName}}-l-b-cell{{matrix[y][x]}}"
class="line l-b-corner"></span>
            <span id="{{functionName}}-r-b-cell{{matrix[y][x]}}"
class="line r-b-corner"></span>
            <span id="{{functionName}}-l-t-r-cell{{matrix[y][x]}}"
class="line l-t-r-cup"></span>
            <span id="{{functionName}}-l-b-r-cell{{matrix[y][x]}}"
class="line l-b-r-cup"></span>
            <span id="{{functionName}}-t-l-b-cell{{matrix[y][x]}}"
class="line t-l-b-cup"></span>
            <span id="{{functionName}}-t-r-b-cell{{matrix[y][x]}}"
class="line t-r-b-cup"></span>
        </div>
```

Joonis 29 Üks ruut Karnaugh' kaardil.

Kui kontuur puudutab lahtrit, omandab vastav element div sobiva klassi, olenevalt kontuuri asukohast:

- l-100 – vasak joon terve ruudu pikkusega.
- r-100 – parem joon terve ruudu pikkusega.
- t-100 – ülemjoon terve ruudu pikkusega.
- b-100 – alamjoon terve ruudu pikkusega.
- l-r-t-b – joon, mis moodustab ringi ühes ruudus.
- l-t – joon, mis moodustab vasaku ülemnurga.
- r-t – joon, mis moodustab parema ülemnurga.
- l-b – joon, mis moodustab vasaku alamnurga.
- r-b – joon, mis moodustab parema alamnurga.
- l-t-r – joon, mis kulgeb mööda vasakut, ülemist ja paremat külge.
- l-b-r – joon , mis kulgeb mööda vasakut, alumist ja paremat külge.
- t-l-b – joon , mis kulgeb mööda ülemist, vasakut ja alumist külge.
- t-r-b – joon , mis kulgeb mööda ülemist, paremat ja alumist külge.

Klasside arv võib olla mitu, kui ühte lahtrit mõjutab mitu kontuuri erinevatel viisidel.

Iga span element vastab võimalikule implikandi joone asukohale, mida eristab klassi nimi:

- line l-line-100 – vasak joon terve ruudu pikkusega.
- line r-line-100 – parem joon terve ruudu pikkusega.
- line t-line-100 – ülemjoon terve ruudu pikkusega.
- line b-line-100 – alamjoon terve ruudu pikkusega.
- line l-r-t-b-line – joon, mis moodustab ringi ühes ruudus.
- line l-t-corner – joon, mis moodustab vasaku ülemnurga.
- line r-t-corner – joon, mis moodustab parema ülemnurga.
- line l-b-corner – joon, mis moodustab vasaku alamnurga.
- line r-b-corner – joon, mis moodustab parema alamnurga.
- line l-t-r-cup – joon, mis kulgeb mööda vasakut, ülemist ja paremat külge.
- line l-b-r-cup – joon , mis kulgeb mööda vasakut, alumist ja paremat külge.
- line t-l-b-cup – joon , mis kulgeb mööda ülemist, vasakut ja alumist külge.

- line t-r-b-cup – joon , mis kulgeb mööda ülemist, paremat ja alumist külge.

Kui span elemendile vastab osa implikandist, mis asub kahel tasandil, siis lisatakse klassi nimele “shared”, kui aga ühel tasandil, siis “detached”. Tänu “detached” klassile saavad span elemendid punase piirjoone. Klassiga “shared” span elementidele luuakse pseudo-element (before), millele määratakse vastava kujuga sinine piirjoon. Kuna pseudo-element asub elemendi sees, siis kui samal ruudul on nii sinine kui ka punane implikant, siis jooned ei lähe väga palju üksteise peale.

Joonis 30 Näide kattuvatest implikantidest.

Kindlaks teha iga lahtri klassi nime aitab ImplicantData klass. Lahtrite klassi nimed täiendatakse dünaamiliselt.

```
for (const implicant of this.implicants) {
    const implicantData = new ImplicantData(implicant, this.matrix)
    for (const cell of implicant) {
        const className = implicantData.getCellClassName(cell)

        if (className && className.length > 0) {
            const cellElement = document.getElementById(
                        `${this.functionName}-cell${cell}`);

            let names = className.split(' ')

            for (const name of names) {
                this.renderer.addClass(cellElement, name);

                const lineElement = document.getElementById(
                          `${this.functionName}-${name}-cell${cell}`)
                if (implicantData.implicantType === ImplicantType.shared) {
                    this.renderer.addClass(lineElement, 'shared');
                } else if (implicantData.implicantType ===
                                ImplicantType.detached) {
                    this.renderer.addClass(lineElement, 'detached');
}}}}}
```

Joonis 31 Dünaamiline klassi nimede määratlus.

6.5.6 ImplicantData klass

ImplicantData on klass, mis aitab määrata Karnaugh' kaardi lahtri jaoks klassi.

Klassi konstruktor võtab vastu konkreetse taseme kaardimatriksi ja implikandi, mis asub sel tasemel või puudutab seda. Implikant saabub massiivina, millel on tema positsioonide numbrid.

Klass sisaldab selliseid välju nagu:

- matrix – maatriks, mis esindab Karnaugh' kaardi taset, millel implikant asub.
- implicant – implikant, mis tuleb massiivina koos positsioonidega. Kui implikant paikneb kahe taseme peal, siis loendis on ainult need positsioonid, mis asuvad maatriksil.
- implicantType – implikantide tüüp ImplicantType enum väärtuse kujul (ImplicantType enum).
- cellClassNames – hoiab Map, kus võti on ruudu positsiooni number ja väärtus on klassi nimi string kujul.

Lahtri klassi saamiseks rakendatakse meetodit getCellClassName, mis võtab ruudu numbri ja tagastab klassi nime div elemendi jaoks, kui ImplicantData klassis on olemas vastav teave. Vastasel juhul tagastatakse undefined.

6.5.7 McCluskeyComponent komponent

Komponenti McCluskey' kujundab McCluskey' minimeerimismeetod. See meetod koosneb kleepimistabelist ja katmistabelist.

6.5.8 Kleepimistabeli visualiseerimine McCluskeyComponent komponendis

Kleepimistabel näitab valitud piirkonna kleepimisprotseduuri McCluskey' meetod kasutab mõistet "arvu indeks", mis on ühtede arv disjunktiivse kujude leidmisel või nullide arv konjunktiivse kujude leidmisel selle arvu kahendkujus [8], see on index veerg tabelis. "valitud pk" veerus on kujutatud valitud piirkonna kahendvektorid, grupeeritud vastavalt nende indeksitele.

Sellise vektori põhjal toimub kleepimine. Iga naabersektsiooni kleebitud argumentvektor moodustab lähisvektoreid, mis asuvad "2-sed interv" veerus. Järgmise naabersektsioonide kleepimise tulemus asub "4-sed interv" ja "8-sed interv" veerus.

“M” veerus märgistatakse kas implikant oli kleebitud, või see on lihtimplikant. “K” täht näitab, et vektor oli kleebitud. “A” täht märkis lihtimplikandid. Number pannakse sellele, et aru saada, mis lihtimplikant kleepimistabelist on näidatud katmistabelis.

Kleepimistabel

index	valitud pk	M	2-sed interv	M	4-sed interv	M
0						
1	00001 00100 10000	K K K	0-001 -0001 0-100 001-0 1000- 100-0	K K K K A3 A4	--001 0-1-0	A16 A17
2	01100 00110 01001 10001 10010	K K K K K	0110- 011-0 0-110 -0110 01-01 010-1 -1001 10-01 1-001 1-010 10-10	A5 A6 A7 A8 K A9 K K K A10 A11	-1-01 1--01	A18 A19
3	01101 01110 01011 10101 11010 11001 10110	K A1 A2 K K K K	-1101 1-101 11-01	A12 A13 A14		
4	11101	K	111-1	A15		
5	11111	K				

Joonis 32 Kleepimistabel McCluskey’ komponendis.

6.5.9 Katmistabeli visualiseerimine McCluskeyComponent komponendis

Katmistabel on lihtimplikantide tabel, mis näitab valitud piirkonna katmist lihtimplikantide poolt [8].

“A” veerus on kirjutatud lihtimplikantide tähistused kleepimistabelist. Ülemises reas on valitud piirkonna vektorid. “X” tähega on märgistatud lihtimplikandiga kaetud vektorid.

Punase värviga on märgistatud need implikandid, mis olid valitud konkreetse funktsiooni kuju koostamiseks.

Katmistabel

A	0	1	3	5	10	12	13	22	26	27	28	29	30	31
A1	X	X												
A2		X	X											
A3		X		X										
A4				X			X							
A5					X				X					
A6								X					X	
A7										X				X
A8													X	X
A9						X	X				X	X		
A10									X	X			X	X
A11											X	X	X	X

Joonis 33 Katmistabel McCluskey' komponendis.

6.5.10 ReducedFunctionComponent komponent

ReducedFunction komponent visualiseerib jääkfunktsiooni. Komponendi sisendandmed on "reducedFunctionData" mille tüüp on ReducedFunctionResponse (ReducedFunctionResponse liides). Selle andmete põhjal genereeritakse väärtused:

- startFunctionString – funktsioon, millest arvutatakse jääkfunktsioon stringi kujul.
- endFunctionString – tulemus funktsiooni string kujul.
- truthTable – massiiv, mis sisaldab Value väärtuseid tõeväärtustabeli koostamiseks.
- endValues – jääkfunktsiooni muutujad.

Komponendi üleval pool asub algne funktsioon, järgmisel joonel asub selle funktsiooni jääkfunktsioon. Allpool asub TruthTable komponent, mis visualiseerib jääkfunktsiooni tõeväärtustabeli kujul.

MDNK: (X̄2X3X̄4X5) v (X1X2X3X̄4) v (X̄2X̄3X̄5) v (X̄1X2X̄3) v (X̄1X2X̄5) v (X1X̄2X̄3) v (X2X4X5)

(X̄2 & X3 & 0 & X5) v (X1 & X2 & X3 & 0) v (X̄2X̄3X̄5) v (X̄1X2X̄3) v (X̄1X2X̄5) v (X1X̄2X̄3) v (X2 & 1 & X5) =

= (X2X5) v (X̄2X̄3X̄5) v (X̄1X2X̄3) v (X̄1X2X̄5) v (X1X̄2X̄3)

jääkfunktsiooni f(x1 x2 x3 1 x5) tõeväärtustabel

X1	X2	X3	X5	f
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**0**
0	1	0	0	**1**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**1**
1	0	1	0	**0**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	**1**

Joonis 34 ReducedFunction komponent.

6.5.11 ShannonComponent komponent

Shannon komponent kujundab minimaalse normaalkuju Shannoni disjunktiivse arenduse.

Komponent võtab shannonData ShannonResponse kujul (ShannonResponse klass) ja määrab muutujad:

- startFunctionString – funktsioon, millest leitakse Shannoni arendus stringi kujul.
- endFunctionString – arenduse tulemus funktsioon stringi kujul.

Komponendi üleval pool on kujundatud iga algtermi jääkfunktsioonid. Iga jääkfunktsioon alustab uuest joonest, et suur funktsioon oleks arusaadavam. All on kujundatud disjunktiivse Shannoni arenduse vastus.

f = (X̄1X̄3X̄5((X̄2 & 1 & X4 & 1) v (X̄2 & 0 & X̄4 & 1) v (1 & X2 & X4 & 0) v (X2 & 1 & X̄4) v (0 & 1 & X̄4) v (1 & 0 & X4) v (0 & X̄2 & 0))) v
(X̄1X̄3X5((X̄2 & 1 & X4 & 0) v (X̄2 & 0 & X̄4 & 0) v (1 & X2 & X4 & 1) v (X2 & 1 & X̄4) v (0 & 1 & X̄4) v (1 & 0 & X4) v (0 & X̄2 & 1))) v
(X̄1X3X̄5((X̄2 & 0 & X4 & 1) v (X̄2 & 1 & X̄4 & 1) v (1 & X2 & X4 & 0) v (X2 & 0 & X̄4) v (0 & 0 & X̄4) v (1 & 1 & X4) v (0 & X̄2 & 0))) v
(X̄1X3X5((X̄2 & 0 & X4 & 0) v (X̄2 & 1 & X̄4 & 0) v (1 & X2 & X4 & 1) v (X2 & 0 & X̄4) v (0 & 0 & X̄4) v (1 & 1 & X4) v (0 & X̄2 & 1))) v
(X1X̄3X̄5((X̄2 & 1 & X4 & 1) v (X̄2 & 0 & X̄4 & 1) v (0 & X2 & X4 & 0) v (X2 & 1 & X̄4) v (1 & 1 & X̄4) v (0 & 0 & X4) v (1 & X̄2 & 0))) v
(X1X̄3X5((X̄2 & 1 & X4 & 0) v (X̄2 & 0 & X̄4 & 0) v (0 & X2 & X4 & 1) v (X2 & 1 & X̄4) v (1 & 1 & X̄4) v (0 & 0 & X4) v (1 & X̄2 & 1))) v
(X1X3X̄5((X̄2 & 0 & X4 & 1) v (X̄2 & 1 & X̄4 & 1) v (0 & X2 & X4 & 0) v (X2 & 0 & X̄4) v (1 & 0 & X̄4) v (0 & 1 & X4) v (1 & X̄2 & 0))) v
(X1X3X5((X̄2 & 0 & X4 & 0) v (X̄2 & 1 & X̄4 & 0) v (0 & X2 & X4 & 1) v (X2 & 0 & X̄4) v (1 & 0 & X̄4) v (0 & 1 & X4) v (1 & X̄2 & 1))) =

= (X̄1X̄3X̄5((X̄2X4) v (X2X̄4))) v (X̄1X̄3X5((X2X4) v (X2X̄4))) v (X̄1X3X̄5(X4 v (X̄2))) v (X̄1X3X5(X4)) v (X1X̄3X̄5((X̄4 v (X̄2)) v (X2X̄4))) v (X1X̄3X5(X̄4 v X̄2)) v (X1X3X̄5(X̄2X̄4)) v
(X1X3X5(X̄2))

Joonis 35 ShannonComponent komponent.

6.6 Suhtlus veebiteenusega

Suhtlus veebiteenusega toimub ainsa teenuse kaudu. Teenuse nimetus on CalculationService, siin on meetod getCalculatorResponse, mis võtab sisse tudengi matriklinumbri 5 viimast arvu ja tagastab CalculatorResponse objekti (CalculatorResponse liides).

```
export class CalculationService {
    private baseUrl = environment.baseUrl
    private apiUrl = 'calculator/calculate';  // URL to web api

    constructor(private http: HttpClient) {}

    getCalculatorResponse(studentNumber: string):
Observable<CalculatorResponse> {
        const url = `${this.baseUrl}/${this.apiUrl}/${studentNumber}`;
        return this.http.get<CalculatorResponse>(url);
    }}
```

Joonis 36 CalculationService klass.

Päring koosneb kolmast osast:

- baseUrl - link hostile. Sõltuvalt sellest, kus rakendus töötab, võib see olla erinev. Kui see töötab lokaalselt, siis on baseUrl “http://localhost:8080”, kuid kui see on toodangus, siis on see “https://app-diskmatexpressioncalculator-230324182727.azurewebsites.net”.
- apiUrl - URL päring “calculator/calculate”.
- studentNumber – viis viimast matriklinumbri arvu string kujul.

7 Veebirakenduse arhitektuur

Kirjeldatud veebirakendus on kalkulaator, mis kasutab nii teenusepoolseid kui ka kliendipoolseid komponente. Kliendipoolne komponent sisaldab välja tudengi matriklinumbri sisestamiseks ja nuppu, mis saadab GET päringu teenusepoolsele komponendile arvutuse sooritamiseks. Andmebaasi ei kasutata, kuna teenusepoolne komponent tegeleb ainult arvutuste tegemisega.

Kliendipoolne osa on teostatud kasutades Angulari ja TypeScripti, samal ajal kui teenusepoolne on teostatud kasutades Springi ja Java -t. Teenusepoolne komponent pakub ainult ühte otsepunkti arvutuse tulemuste saamiseks vastusena GET päringule, mis sisaldab string parameetrit tudengi matriklinumbri viis viimast numbrit.

Kuigi süsteem ei sisalda autentimist, teostab teenusepoolne komponent sisendi valideerimist kasutades regulaaravaldisi. Kui tudengi sisend ei vasta oodatud formaadile või sisaldab ohtlikke märke, visatakse erand.

Üldiselt järgib selle veebirakenduse arhitektuur klient-server mudelit, kus Angular toimib kliendina ja Spring Boot serverina. Kliendi pool ja teenuse pool suhtlevad kasutades HTTP päringuid ja JSON andmeid. Teenusepoolne komponent on disainitud olema moodulipõhine ja sõltumatu, mis võimaldab tulevikus hõlpsamat hooldust ja skaleeritavust, ehkki seda pole selle rakenduse jaoks vaja.

See rakendus on välja töötatud lõputöö raames ega hõlma skaleeritavust, kõrget koormust ega ründeid.

8 Kokkuvõte

Antud lõputöö eesmärk on rakendada veebipõhine tarkvara, mis leiaks erinevaid loogikaavaldisi 5-muutuja loogikafunktsioonidele tudengi matriklinumbri põhjal, et anda õpilastele näiteid 5-muutujaga loogilistest vormidest ja vormide kujudest, mis lisaks funktsioonide ja nende vormi saamise näidetele aitab ka tudengil põhikursusest kaugemale jõuda.

Anti ülevaade olemasolevatest lahendustest, mis andis teada, et 5-muutujaga kalkulaator oleks kasulik üliõpilastele ja seda saab rakendada mitte ainult TalTechis, vaid ka teistes haridusasutustes.

Analüüsis on ülevaade antud rakenduse arendusprotsessile ja koodi struktuurile nii kliendi poolel kui ka teenuse poolel. On kirjeldatud kasutatud tehnoloogiad ja sõnastatud nende kasutamine.

Projekt on edukalt arendatud ja kättesaadav avaliku domeeni kaudu. Kõik määratud nõuded olid täidetud, rakendus on avalik ja töötab nii, nagu oli planeeritud. Tulevikus saab laiendada kalkulaatori funktsionaalsust, näiteks laiendada leidvate funktsioonide erikujude arvu ja lisada erinevaid sisendandmete tüüpe, et kasutaja saaks valida endale sobiva. Veel saab rakendatud kalkulaatorit kasutada teistes projektides, mis on seotud erikujude kalkuleerimisega.

Kirjanduse loetelu

[1] Apache Maven Project, "Introduction," Apache, 11 Detsember 2022. [Online]. Available: https://maven.apache.org/what-is-maven.html. [Accessed 7 Aprill 2023].

[2] Gradle, "What is Gradle?," Gradle, [Online]. Available: https://docs.gradle.org/current/userguide/what_is_gradle.html. [Accessed 7 Aprill 2023].

[3] Vue, "Introduction," Vue, [Online]. Available: https://vuejs.org/guide/introduction.html. [Accessed 7 Aprill 2023].

[4] Aurelia, "What is Aurelia," Aurelia, [Online]. Available: https://aurelia.io/docs/overview/what-is-aurelia/. [Accessed 7 Aprill 2023].

[5] React, "React," React, [Online]. Available: https://react.dev/. [Accessed 7 Aprill 2023].

[6] Angular , "Introduction to Angular concepts," Angular , [Online]. Available: https://angular.io/guide/architecture. [Accessed 7 Aprill 2023].

[7] A. Zola, "Bootstrap," techtarget, August 2022. [Online]. Available: https://www.techtarget.com/whatis/definition/bootstrap. [Accessed 7 Aprill 2023].

[8] M. K. Harri Lensen, Diskreetne matemaatika, Tallinn: Infotrükk, 2006.

[9] S. S. Epp, Discrete Mathematics with Applications, Boston: Cengage Learning, 2018.

[10] K. H. Rosen, Discrete Mathematics and Its Applications, New York: McGraw-Hill Education, 2019.

[11] S. L. Ben Lutkevich, "Waterfall Model," TechTarget, September 2022. [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model. [Accessed 7 Oktoober 2022].

[12] Angular, "Input," Angular, 2 Juuni 2022. [Online]. Available: https://v14.material.angular.io/components/input/overview. [Accessed 3 Veebruar 2023].

[13] Oracle, "JDK 17 Documentation," Oracle, 25 Jaanuar 2022. [Online]. Available: https://docs.oracle.com/en/java/javase/17/. [Accessed 13 Oktoober 2022].

[14] VMware, "Building an Application with Spring Boot," VMware, [Online]. Available: https://spring.io/guides/gs/spring-boot/. [Accessed 3 Oktoober 2022].

[15] Angular, "Getting started with Angular," Angular, [Online]. Available: https://angular.io/start. [Accessed 12 Veebruar 2023].

[16] learntek, "What is Maven?," Learntek, 4 Juuni 2019. [Online]. Available: https://www.learntek.org/blog/what-is-maven/. [Accessed 7 Oktoober 2022].

[17] Amazon, "What Is An API (Application Programming Interface)?," Amazon, [Online]. Available: https://aws.amazon.com/what-is/api/. [Accessed 2 September 2022].

[18] Intellipaat, "What is Client Server Architecture?," Intellipaat, [Online]. Available: https://intellipaat.com/blog/what-is-client-server-architecture/?US. [Accessed 20 August 2022].

[19] M. Roomi, "5 Advantages and Disadvantages of HTTP | Drawbacks & Benefits of HTTP," Hitechwhizz, 14 August 2020. [Online]. Available:

https://www.hitechwhizz.com/2020/08/5-advantages-and-disadvantages-drawbacks-benefits-of-http.html. [Accessed 17 September 2022].

[20] SmartBear, "SOAP vs REST. What's the Difference?," SmartBear, 2 Jaanuar 2020. [Online]. Available: https://smartbear.com/blog/soap-vs-rest-whats-the-difference/#:~:text=While%20SOAP%20and%20REST%20share,and%20is%20naturally%20more%20flexible.. [Accessed 10 August 2022].

Lisa1 – Lihtlitsens

Mina, Anzelika Muravskaja

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose “5-muutuja loogikafunktsiooni minimaalseid normaalkujusid ja loogikaavaldiste erikujusid leidva veebirakenduse programmeerimine JAVA keeles, mille juhendaja on Harri Lensen.

   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

Lisa 2 – CalculatorService klass

```
public class CalculatorService {

    @Autowired
    private TruthTableService truthTableService;
    @Autowired
    private ReducedFunctionService reducedFunctionService;
    @Autowired
    private McCluskeyFunctionService mcCluskeyFunctionService;
    @Autowired
    private ShannonFunctionService shannonFunctionService;

    public CalculatorResponse getCalculations(String studentNumber) {
        checkStudentNumber(studentNumber);

        //Truth table
        TruthTableResponse truthTableResponse =
truthTableService.calculateTruthTable(studentNumber);

        //McCluskey + all function responses
        McCluskeyFunctionResponse conjunctiveMcCluskeyFunctionResponse =
mcCluskeyFunctionService.calculateMcCluskeyFunction(truthTableResponse.getZer
oPositions());
        McCluskeyFunctionResponse disjunctiveMcCluskeyFunctionResponse =
mcCluskeyFunctionService.calculateMcCluskeyFunction(truthTableResponse.getOne
Positions());

        //Shannon
        Function functionForShannonExpansion =
disjunctiveMcCluskeyFunctionResponse.getMinimalNormalFormData().getFunction()
;
        ShannonResponse shannonResponseX1X3X5 =
shannonFunctionService.calculateDisjunctiveShannon(functionForShannonExpansio
n, List.of(ValueName.X1, ValueName.X3, ValueName.X5));
        ShannonResponse shannonResponseX2X4 =
shannonFunctionService.calculateDisjunctiveShannon(functionForShannonExpansio
n, List.of(ValueName.X2, ValueName.X4));

        //Expansion
        Function functionForExpansion =
disjunctiveMcCluskeyFunctionResponse.getMinimalNormalFormData().getFunction()
;
        ReducedFunctionResponse reducedFunctionResponseX2Zero =
reducedFunctionService.calculateReducedFunction(functionForExpansion,
ValueName.X2, 0);
        ReducedFunctionResponse reducedFunctionResponseX4One =
reducedFunctionService.calculateReducedFunction(functionForExpansion,
ValueName.X4, 1);
        ReducedFunctionResponse reducedFunctionResponseX1One =
reducedFunctionService.calculateReducedFunction(functionForExpansion,
ValueName.X1, 1);
        ReducedFunctionResponse reducedFunctionResponseX3Zero =
reducedFunctionService.calculateReducedFunction(functionForExpansion,
ValueName.X3, 0);
```

```
        //Set response values
        CalculatorResponse calculatorResponse = new CalculatorResponse();

        calculatorResponse.setTruthTableResponse(truthTableResponse);

calculatorResponse.setConjunctiveMcCluskeyFunctionResponse(conjunctiveMcClusk
eyFunctionResponse);

calculatorResponse.setDisjunctiveMcCluskeyFunctionResponse(disjunctiveMcClusk
eyFunctionResponse);

        calculatorResponse.setShannonResponseX1X3X5(shannonResponseX1X3X5);
        calculatorResponse.setShannonResponseX2X4(shannonResponseX2X4);


calculatorResponse.setReducedFunctionResponseX2Zero(reducedFunctionResponseX2
Zero);

calculatorResponse.setReducedFunctionResponseX3Zero(reducedFunctionResponseX3
Zero);

calculatorResponse.setReducedFunctionResponseX1One(reducedFunctionResponseX1O
ne);

calculatorResponse.setReducedFunctionResponseX4One(reducedFunctionResponseX4O
ne);

        return calculatorResponse;
    }
```

Lisa 3 – ShannonFunctionService klass

```
public class ShannonFunctionService {

    public ShannonResponse calculateDisjunctiveShannon(Function function,
List<ValueName> expansionValues) {
        Function startFunction = new Function(new Operation.Or());
        Function endFunction = new Function(new Operation.Or());

        int amountOfValues = expansionValues.size();
        int settingNumber = (int) Math.pow(2, amountOfValues);

        for (int i = 0; i < settingNumber; i++) {
            char[] settingNumbers = decToBinaryWithLength(i,
amountOfValues).toCharArray();

            Function shannonStartPart = new Function(new Operation.And());
            Function shannonEndPart = new Function(new Operation.And());

            Function functionClone = function.clone();

            for (int bin = 0; bin < settingNumbers.length; bin++) {
                ValueName valueName = expansionValues.get(bin);
                Value value = new Value(valueName);

                Integer number =
Integer.parseInt(String.valueOf(settingNumbers[bin]));

                if (number.equals(0)) {
                    value.invert();
                }
                shannonStartPart.addElement(value);
                shannonEndPart.addElement(value);

                functionClone.setNumberToValue(number, valueName);

            }

            shannonStartPart.addElement(functionClone);

            FunctionPart endElement = simplifyFunction(functionClone);
            if (endElement.isValue()) {
                endElement = new Function(new Operation.And(),
List.of(endElement));
            }

            shannonEndPart.addElement(endElement);

            startFunction.addElement(shannonStartPart);
            endFunction.addElement(shannonEndPart);
        }

        return new ShannonResponse(startFunction, endFunction);

    }
}
```

Lisa 4 – Tõeväärtustabeli komponent

Tõeväärtustabel

X1	X2	X3	X4	X5	f
0	0	0	0	0	**1**
0	0	0	0	1	**1**
0	0	0	1	0	**1**
0	0	0	1	1	**1**
0	0	1	0	0	**0**
0	0	1	0	1	**1**
0	0	1	1	0	**0**
0	0	1	1	1	**0**
0	1	0	0	0	**0**
0	1	0	0	1	**0**
0	1	0	1	0	**0**
0	1	0	1	1	**1**
0	1	1	0	0	**1**
0	1	1	0	1	**1**
0	1	1	1	0	**0**
0	1	1	1	1	**1**
1	0	0	0	0	**0**
1	0	0	0	1	**1**
1	0	0	1	0	**1**
1	0	0	1	1	**1**
1	0	1	0	0	**1**
1	0	1	0	1	**1**
1	0	1	1	0	**0**
1	0	1	1	1	**1**
1	1	0	0	0	**1**
1	1	0	0	1	**1**
1	1	0	1	0	**0**
1	1	0	1	1	**1**
1	1	1	0	0	**0**
1	1	1	0	1	**0**
1	1	1	1	0	**0**
1	1	1	1	1	**0**

Lisa 5 – Linkid proekti lähtekoodile ja paigaldatud rekendusele

Teenusepoolne lähtekood:
https://bitbucket.org/anzelikamuravskaja/diskmatcalculatorback/src/master/

Kliendipoolne lähtekood:
https://bitbucket.org/anzelikamuravskaja/diskmatcalculatorfront/src/master/

Link avaliku kliendipoole:
https://expressioncalculator.azurewebsites.net/

Link avaliku teenusepoole:
https://app-diskmatexpressioncalculator-230324182727.azurewebsites.net
(/calculator/calculate/12345)