TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Norman Pirk

Loogikavalemite teisendusredaktor

Bakalaureusetöö (9 EAP)

Juhendaja Reimo Palm

Tartu 2022

Loogikavalemite teisendusredaktor

Lühikokkuvõte:

Lõputöö eesmärk on loogikavalemite teisendamiseks luua veebirakendus, milles teisendussammude tegemine sarnaneb võimalikult palju käsitsi teisendamisega. Sellist tööriista on vaja uuel matemaatilise loogika kursusel ja muudel kursustel, kus olulisel kohal on lause- ja predikaatarvutuse valemite teisendamine. Töö alguses analüüsiti olemasolevaid rakendusi ja leiti nende põhilised puudujäägid. Tulemusena loodi uus veebirakendus, milles on kõrvaldatud või leevendatud nimetatud puudused. Selleks, et rakendus oleks võimalikult kasutajasõbralik, viidi läbi kasutajatestimine, mille tulemusi arvestati rakenduse lõplikul viimistlemisel.

Võtmesõnad:

Lausearvutus, predikaatarvutus, põhisamaväärsus, teisendamine, veebirakendus, Vue.js

CERCS: P175 Informaatika, süsteemiteooria; P110 Matemaatiline loogika, hulgateooria, kombinatoorika

Application for Transforming Mathematical Logic Expressions

Abstract:

The aim of this bachelor thesis is to create a web application that allows its users to perform transformations on mathematical logic expressions similarly to doing the transformations manually. The application is needed for a new mathematical logic course and other courses where transforming the expressions of propositional and predicate logic is one of the main activities. An analysis of existing software for transforming logic expressions was conducted which brought out the main disadvantages of the tools. Then, a new web application was created in which the negative aspects of the existing tools were either removed or mitigated. To further enhance the usability of the new application, user testing was performed. Its results were considered when completing the development of the application.

Keywords:

Propositional logic, predicate logic, logical equivalence, transformation, web application, Vue.js

CERCS: P175 Informatics, systems theory; P110 Mathematical logic, set theory, combinatories

Sisukord

1. Sissejuhatus

Tartu Ülikooli arvutiteaduse instituudis töötatakse välja uut matemaatilise loogika kursust, mille sisu kasutab olulisel määral aines „Diskreetne matemaatika I LTMS.00.019“ õpetatud lause- ja predikaatarvutust[1]. Kursusel on tähtsal kohal loogikavalemite teisendamine. Kuna teisendussammude tegemine käsitsi võtab kaua aega, on selleks mõistlik kasutada tööriista, mis aitab kasutajal pääseda kirjutamisvaevast ja sellega seotud ajakulust.

Loogikavalemite teisendamiseks on olemas palju rakendusi, kuid neist enamiku puhul ei sarnane teisenduste tegemine käsitsi teisendamisega, sest kogu teisenduskäik tehakse korraga ning kasutaja saab kätte vaid lõpptulemuse. Sel puhul on aga kasutajal võimatu seada teisendamiseks kohandatud eesmärk, näiteks teisendada ainult väiksemat osa valemist.

Selle bakalaureusetöö eesmärk on luua tööriist, millega saab lause- ja predikaatarvutuse valemeid teisendada võimalikult sarnaselt käsitsi teisendamisega. Oluline on, et kasutaja saaks ise määrata teisendamise eesmärgi. Loodav tööriist peab aitama kasutajatel teisendamisele kuluvat aega kokku hoida.

Töö teises peatükis tuuakse välja olulisemad lause- ja predikaatarvutusega seotud mõisted. Kolmandas peatükis analüüsitakse kuut olemasolevat tööriista, millega on võimalik loogikavalemeid lihtsustada või teisendada. Selle analüüsi põhjal tuuakse neljandas peatükis välja loodava tööriista funktsionaalsed ja mittefunktsionaalsed nõuded. Viiendas peatükis kirjeldatakse lühidalt loodud rakenduse arhitektuuri ning põhjalikumalt kasutatud tehnoloogiaid. Kuuendas peatükis kirjeldatakse automaat- ja kasutajatestimist ning tuuakse välja kasutajatestimise tulemused. Seitsmendas peatükis kirjeldatakse valminud rakendust. Näitena tuuakse üks teisendus, mille erinevate sammude juures selgitatakse, kuidas rakenduses vastav samm tehakse. Lisaks selgitatakse, millised muudatused tehti kasutajatestimise tulemuste analüüsi põhjal. Peatükk lõpetatakse loodud rakenduse eeliste ja edasiarendusvõimaluste väljatoomisega. Töö viimases peatükis võetakse tehtu lühidalt kokku.

Lisadena on tööle kaasa pandud kasutajatestimise küsimustik ning vastused, rakenduse lähtekood, rakenduses võimaldatud loogikavalemite teisendusreeglid, rakenduse kasutusjuhend, pikemad koodinäited ja näited allalaaditavate failide sisust.

[1] Informatsioon uue kursuse loomisest on saadud töö juhendajalt.

2. Lause- ja predikaatarvutuse mõisted

Selles peatükis tuuakse välja olulisemad lause- ja predikaatarvutuse mõisted. Rasvase kirjaga rõhutatakse mõisteid, mida töös hiljem kasutatakse. Need mõisted, mida ei ole rasvase kirjaga välja toodud, on vajalikud hiljem kasutatavate mõistete defineerimiseks.

2.1 Lausearvutuse mõisted

Lausearvutuse valemid on parajasti need, mida saab koostada järgmiste reeglite abil:

1) iga lausemuutuja on lausearvutuse valem;
2) kui F on predikaatarvutuse valem, siis $\neg F$ on lausearvutuse valem;
3) kui F ja G on lausearvutuse valemid, siis $(F \wedge G)$, $(F \vee G)$, $(F \Rightarrow G)$ ja $(F \Leftrightarrow G)$ on lausearvutuse valemid [1:9].

Valemeid F ja G nimetatakse **samaväärseteks**, kui nende tõeväärtused on võrdsed igal neis valemeis esinevate muutujate väärtustusel [1:21].

Lausemuutujaid või nende eitusi nimetatakse literaalideks [1:24].

Elementaarkonjunktsiooniks nimetatakse literaalide konjunktsiooni, kusjuures elementaarkonjunktsiooni nimetatakse täielikuks, kui selles esinevad valemi kõik muutujad [1:24,28].

Elementaardisjunktsiooniks nimetatakse literaalide disjunktsiooni, kusjuures elementaardisjunktsiooni nimetatakse täielikuks, kui selles esinevad valemi kõik muutujad [1:24–25,28].

Lausearvutuse valemi F **täielikuks disjunktiivseks normaalkujuks (TDNK)** nimetatakse valemiga F samaväärset valemit, mis kujutab endast erinevate täielike elementaarkonjunktsioonide disjunktsiooni [1:24].

Lausearvutuse valemi F **täielikuks konjunktiivseks normaalkujuks (TKNK)** nimetatakse valemiga F samaväärset valemit, mis kujutab endast erinevate täielike elementaardisjunktsioonide konjunktsiooni [1:25].

Lausearvutuse valemi F **disjunktiivseks normaalkujuks (DNK)** nimetatakse valemiga F samaväärset valemit, mis kujutab endast erinevate elementaarkonjunktsioonide disjunktsiooni [1:28].

Lausearvutuse valemi F **konjunktiivseks normaalkujuks (KNK)** nimetatakse valemiga F samaväärset valemit, mis kujutab endast erinevate elementaardisjunktsioonide konjunktsiooni [1:28].

Valemi F **minimaalseks disjunktiivseks normaalkujuks (MDNK)** nimetatakse valemi F disjunktiivset normaalkuju, mis sisaldab kõige vähem literaale [1:31].

2.2 Predikaatarvutuse mõisted

Hulgal M määratud n-kohaliseks **predikaadiks** nimetatakse kujutust $P : M^n \rightarrow \{1,0\}$ [1:45].

Kui $P(x_1, x_2, ..., x_n)$ on hulgal M määratud n-kohaline predikaat, siis kirjutis $\forall x_i(x_1, ..., x_i, ..., x_n)$ tähistab $(n–1)$-kohalist predikaati, mis on tõene parajasti siis, kui argumentide $x_1, ..., x_{i-1}, x_{i+1}, ..., x_n$ väärtused on sellised, et hulga M iga elemendi m korral $P(x_1, ..., m, ..., x_n)$ on tõene; operaatorit $\forall$ nimetatakse **üldisuskvantoriks** [1:46].

Kui $P(x_1, x_2, ..., x_n)$ on hulgal M määratud n-kohaline predikaat, siis kirjutis $\exists x_i(x_1, ..., x_i, ..., x_n)$ tähistab $(n–1)$-kohalist predikaati, mis on tõene parajasti siis, kui argumentide $x_1, ..., x_{i-1}, x_{i+1}, ..., x_n$ väärtused on sellised, et hulgas M leidub element m, mille korral $P(x_1, ..., m, ..., x_n)$ on tõene; operaatorit $\exists$ nimetatakse **olemasolukvantoriks** [1:46–47].

Termid on parajasti need, mida saab koostada järgmiste reeglite abil:

1) iga indiviidmuutuja on term;
2) iga konstantsümbol on term;
3) kui f on n-kohaline funktsionaalsümbol ja $t_1, t_2, ..., t_n$ on termid, siis $f(t_1, t_2, ..., t_n)$ on term [1:49].

Predikaatarvutuse valemid on parajasti need, mida saab koostada järgmiste reeglite abil:

1) kui P on n-kohaline predikaatsümbol ja $t_1, t_2, ..., t_n$ on termid, siis $P(t_1, t_2, ..., t_n)$ on predikaatarvutuse valem;
2) kui F on predikaatarvutuse valem, siis $\neg F$ on predikaatarvutuse valem;
3) kui F ja G on predikaatarvutuse valemid, siis $(F \wedge G)$, $(F \vee G)$, $(F \Rightarrow G)$ ja $(F \Leftrightarrow G)$ on predikaatarvutuse valemid;
4) kui x on indiviidmuutuja ja F on predikaatarvutuse valem, siis $\forall xF$ ja $\exists xF$ on predikaatarvutuse valemid [1:50].

Kui indiviidmuutuja x esineb valemis F mingi kvantori mõjupiirkonnas, ehk osavalemit $\forall xG$ või $\exists xG$ moodustavas valemis G, siis nimetatakse teda **seotud muutujaks**, vastasel juhul **vabaks muutujaks** [1:50].

Signatuuriks nimetatakse kolmikut $\sigma = \langle C;F;P \rangle$, kus C on konstantsümbolite, F funktsionaalsümbolite ja P predikaatsümbolite hulk [1:52].

Predikaatarvutuse valemit, milles esinevad konstantsümbolid kuuluvad kõik hulka C, funktsionaalsümbolid hulka F ning predikaatsümbolid hulka P, nimetatakse valemiks signatuuris σ [1:52].

Interpretatsioon on paar $\alpha = (M_\alpha, I_\alpha)$, kus M_α on mingi mittetühi hulk, mida nimetatakse interpretatsiooni kandjaks, ja I_α on interpreteeriv kujutus, mis teisendab

1) iga konstantsümboli hulga M_α mingiks elemendiks;
2) iga n-kohalise funktsionaalsümboli mingiks (kõikjal määratud) n-kohaliseks funktsiooniks hulgal M_α;
3) iga n-kohalise predikaatsümboli mingiks n-kohaliseks predikaadiks hulgal M_α [1:53].

Termi t väärtus t^α interpretatsioonis α vabade muutujate fikseeritud väärtustel leitakse järgmiste reeglite abil:

1) kui $t = x$, kus x on indiviidmuutuja, siis t^α on muutuja x väärtus;
2) kui $t = c$, kus c on konstantsümbol, siis $t^\alpha = c^\alpha$;
3) kui $t = f(t_1, t_2, ..., t_n)$, kus f on n-kohaline funktsionaalsümbol ja $t_1, t_2, ..., t_n$ on termid, siis $t^\alpha = f^\alpha(t_1^a, t_2^a, \ldots, t_n^a)$ [1:54].

Predikaatarvutuse valemi F tõeväärtus F^α interpretatsioonis α vabade muutujate fikseeritud väärtustel leitakse järgmiste reeglite abil:

1) kui $F = P(t_1, t_2, ..., t_n)$, kus P on n-kohaline predikaatsümbol ja $t_1, t_2, ..., t_n$ on termid, siis $F^\alpha = 1$ parajasti siis, kui $P^\alpha(t_1^a, t_2^a, \ldots, t_n^a) = 1$;
2) kui $F = \neg G$, siis $F^\alpha = 1$ parajasti siis, kui $G^\alpha = 0$;
3) kui $F = G \wedge H$, siis $F^\alpha = 1$ parajasti siis, kui $G^\alpha = 1$ ja $H^\alpha = 1$;
4) kui $F = G \vee H$, siis $F^\alpha = 1$ parajasti siis, kui $G^\alpha = 1$ või $H^\alpha = 1$;
5) kui $F = G \Rightarrow H$, siis $F^\alpha = 1$ parajasti siis, kui $G^\alpha = 0$ või $H^\alpha = 1$;
6) kui $F = G \Leftrightarrow H$, siis $F^\alpha = 1$ parajasti siis, kui $G^\alpha = 1$ ja $H^\alpha = 1$ või $G^\alpha = 0$ ja $H^\alpha = 0$;

7) kui $F = \forall xG$ siis $F^{\alpha} = 1$ parajasti siis, kui interpretatsiooni kandja M_{α} iga elemendi m korral $G^{a}_{[x/m]} = 1$, kus *[x/m]* tähendab, et muutuja *x* väärtuseks loetakse element *m*;
8) kui $F = \exists xG$ siis $F^{\alpha} = 1$ parajasti siis, kui interpretatsiooni kandjas M_{α} leidub selline element *m*, et $G^{a}_{[x/m]} = 1$, kus *[x/m]* tähendab, et muutuja *x* väärtuseks loetakse element *m* [1:55–56].

Valemeid *F* ja *G* nimetatakse **samaväärseteks**, kui nende tõeväärtused on võrdsed igas interpretatsioonis valemite vabade muutujate kõikidel väärtustel [1:64].

Öeldakse, et valem *F* on **prefikskujul**, kui $F = Q_1x_1Q_2x_2...Q_nx_nF'$, kus Q_1, Q_2, ..., Q_n on kvantorid, x_1, x_2, ..., x_n indiviidmuutujad ja *F'* ilma kvantoriteta valem [1:70].

Selles peatükis toodi välja lause- ja predikaatarvutuse mõisted, mis on olulised valemite teisendamise juures. Järgmises peatükis analüüsitakse erinevaid olemasolevaid tööriistu, millega saab loogikavalemeid teisendada.

3. Olemasolevad rakendused loogikavalemite teisendamiseks

Lause- ja predikaatarvutuse valemite töötlemiseks on olemas erinevaid tööriistu. Selles peatükis analüüsitakse neist kuut, millega on võimalik loogikavalemeid teisendada esialgse valemiga samaväärsele kujule. Eelkõige keskendutakse valemi sisestamise mugavusele, võrreldakse teisendussammude tegemist sellega, kuidas loogikavalemeid käsitsi teisendatakse, ning tuuakse välja teisenduste võimalikud tulemused. Peatüki lõpus tuuakse analüüsi põhjal välja nende tööriistade kasutamise peamised puudused võrreldes loogikavalemite käsitsi teisendamisega.

3.1 Logic Calculator

Rakendus Logic Calculator[2] sisaldab erinevaid funktsioone, mida saab lausearvutuse valemitele rakendada. Muuhulgas on ka võimalik sisestatud valem teisendada erinevatele esialgse valemiga samaväärsetele kujudele. Valemi sisestamiseks on nupud nii loogikasümbolite kui ka lausemuutujate jaoks. Tabelis 1 on välja toodud loogikatehete sisestamise võimalikud variandid klaviatuurilt. Kuna predikaatarvutuse valemeid teisendada ei saa, on kvantorid tabelist välja jäetud.

Tabel 1. Loogikatehete sisestamine rakenduses Logic Calculator.

Tehe[3]	¬	∧	∨	⇒	⇔
Sisestusviisid	~, not	&, and	v, or	->	<->

Klaviatuurilt sisestamine on eituse, konjunktsiooni ja disjunktsiooni korral kiire, sest minimaalselt on vaja sisestada üks sümbol. Implikatsiooni ja ekvivalentsi sisestamine on võimalik vastavalt kahe ja kolme sümboliga. Sisendlahtris ei teisendata loogikatehete tähistusi ühtlaseks. Näiteks võivad samas valemis erinevate osavalemite ees eitust tähistada nii sümbol „~“ kui ka sõna „not“.

Sisestatud valemile on võimalik rakendada ühte järgmistest teisendustest: implikatsioonide ja ekvivalentside eemaldamine, eituste viimine muutujate ette ning viimine kujudele TDNK,

[2] https://www.erpelstolz.at/gateway/formular-uk-zentral.html (11.04.2022)

[3] Tabelites 1–6 tähistatakse tehteid ja kvantoreid sümbolitega, mida kasutati teises peatükis predikaatarvutuse valemi definitsioonis.

DNK, TKNK, KNK ja MDNK. Pärast valitud teisenduse rakendamist avatakse uus lehekülg, kus kõigepealt tuuakse välja valitud teisendus ja esialgne valem ning allpool kuvatakse teisenduse tulemus. Tulemuse kuvamisel on loogikasümbolite tähised ühtlustatud. Tulemuse juures ei ole eraldi välja toodud, millised sammud on teisenduse käigus tehtud. Kui valem on sisestatud, ei ole sellest teisendamiseks võimalik välja eraldada osavalemit. Seetõttu on teisenduste tegemine selles rakenduses erinev käsitsi teisendamisest.

3.2 Logictools

Tammeti [2] loodud Logictools[4] on rakendus, mille eesmärk on aidata kasutajatel õppida matemaatilist loogikat. Rakendus sisaldab tööriistu predikaat- ja lausearvutuse valemite töötlemiseks. Autor on välja toonud, et rakenduse kasutamisel on rõhk sellel, et seda oleks mugav ja lihtne kasutada.

Valemite sisestamine toimub klaviatuurilt, lausemuutujate tähistamiseks on kasutusel ladina tähestiku väike- ja suurtähed ning loogikatehete sisestamiseks eraldi nuppe pole. Tabelis 2 on toodud võimalused loogikatehete sisestamiseks. Kvantoreid ei ole välja toodud, kuna predikaatarvutuse valemeid rakendus teisendada ei võimalda.

Tabel 2. Loogikatehete sisestamine rakenduses Logictools.

Tehe	¬	∧	∨	⇒	⇔
Sisestusviisid	-, ~	&, and	|, v, V, or	->, =>	<->, <=>

Sarnaselt eelmisele rakendusega on eituse, konjunktsiooni ja disjunktsiooni sisestamine klaviatuurilt teostatav ühe ja implikatsiooni ning ekvivalentsi korral vastavalt kahe ja kolme sümboliga. Disjunktsiooni puhul kasutusel ka tähed „v“ ja „V“, mida on võimalik samal ajal kasutada ka lausemuutujate tähistamiseks. Sisendlahtris ei teisendata tehteid tähistavaid sümboleid ühtlaseks. Valemi sisestamisel peab arvestama, et tehetele ei ole määratud prioriteete, mis tähendab, et kasutaja peab tehete järjekorra sulgude abil määrama.

Sisestatud lausearvutuse valemile on võimalik rakendada ühte teisendust, mille tulemus on KNK. Muudele kujudele selles rakenduses valemit teisendada ei saa. Tulemus kuvatakse nii, et tehteid tähistavad sümbolid on viidud ühtlasele kujule. Valemi sisestamise järel ei ole

[4] https://logictools.org/prop.html (11.04.2022)

võimalik valida teisenduse tegemiseks väiksemat osavalemit. Sarnaselt eelmise rakendusega on teisenduste tegemine selles rakenduses erinev võrreldes teisenduste tegemisega käsitsi.

3.3 WolframAlpha

Rakendus WolframAlpha[5] lubab lausearvutuse valemeid teisendada erinevatele kujudele, nende hulgas DNK ja KNK. Teisenduse tegemiseks tuleb sisendisse esimesena anda vastava teisenduse käsk (*DNF*[6] või *CNF*[7] vastavalt DNK või KNK korral) ja seejärel lisada lausearvutuse valem. Lausemuutujatena on kasutusel ladina tähed ning loogikatehete sisestamise võimalikud variandid on välja toodud tabelis 3.

Tabel 3. Loogikatehete sisestamine rakenduses WolframAlpha.

Tehe	¬	∧	∨	⇒	⇔
Sisestusviisid	~, !, not	&, &&, and	||, or	=>	<->, <=>

Sarnaselt eelmise kahe rakendusega on ka rakenduses WolframAlpha loogikatehete sisestamine kiire, erinevus on selles, et disjunktsiooni puhul peab minimaalselt kasutama kahte sümbolit. Samamoodi jäetakse sisendlahtris sümbolite ühtlustamine tegemata.

Pärast käsu ja valemi sisestamist ning reavahetusklahvi vajutamist kuvatakse kõigepealt sisend ja selle all rakendatud funktsiooni tulemus. Lisaks kuvatakse tabel kõikide kujudega, millele rakendus lubab sisendit viia. Tulemuse kuvamisel on loogikasümbolid ühtlustatud. Kui tulemusse jääb sisse konjunktsioon (ühtlustatuna on selle tähis ∧) ja see kopeerida ning kleepida sisendlahtrisse, siis ei tunne rakendus enam sisendit ära. Teisendamiseks ei saa sisestatud valemist valida väiksemat osavalemit. Ka selles rakenduses on lausearvutuse valemite teisendamine erinev võrreldes käsitsi teisendamisega.

[5] https://www.wolframalpha.com/ (11.04.2022)

[6] ingl *Disjunctive Normal Form*

[7] ingl *Conjunctive Normal Form*

3.4 eMathHelp

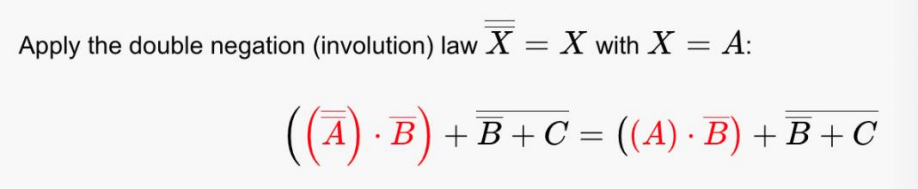
Rakenduses eMathHelp[8] on lausearvutuse valemeid võimalik teisendada kalkulaatoriga Boolean algebra calculator[9]. Valemite sisestamisel on lausemuutujate tähistamiseks ladina tähestik ning loogikatehteid saab sisestada nuppude abil ja klaviatuurilt. Tabelis 4 on toodud võimalikud variandid loogikatehete sisestamiseks. Kuna predikaatarvutuse valemeid rakendus teisendada ei võimalda, on kvantorid tabelist välja jäetud.

Tabel 4. Loogikatehete sisestamine rakenduses eMathHelp.

Tehe	¬	∧	∨	⇒	⇔
Sisestusviisid	~, not	and	or	=>, →	=, ↔, xnor

Selle rakenduse puhul on võimalik eitust ja ekvivalentsi klaviatuurilt sisestada minimaalselt ühe, disjunktsiooni ja implikatsiooni kahe ning konjunktsiooni kolme sümboli abil. Sisendlahtris sümboleid ei ühtlustata, kuid tulemuse kuvamisel on esimesena välja toodud sisend, mille puhul on sümbolid juba ühtlustatud.

Sisestatud valemit proovitakse lihtsustada. Kui on märgitud ka kujudele teisendamise valik (ingl *Calculate the forms*) kuvatakse ka DNK ja KNK ning kuju, millest on eemaldatud implikatsioonid ja ekvivalentsid. Rakenduses kuvatakse ka lihtsustamise käigus tehtud sammud, mille puhul tuuakse välja rakendatud reegel koos reegli tähise ja valemis reeglile vastava osavalemiga. Vastavad osavalemid on esile toodud punase värviga. Katsetamise käigus selgus, et need osavalemid ei ole alati korrektselt märgitud, sest kohati tuuakse lisaks osavalemile punasena välja ka sellele järgnev osa (joonis 1).



Joonis 1. Lihtsustamise käigus kasutatud reegli kuvamine keskkonnas eMathHelp. Ülemises reas on näha, et teisendust on rakendatud osavalemile A, kuid alumises reas on punasega välja toodud ka sellele järgnev osa.

[8] https://www.emathhelp.net/en/ (11.04.2022)

[9] https://www.emathhelp.net/en/calculators/discrete-mathematics/boolean-algebra-calculator/ (11.04.2022)

Selle rakenduse puhul on sisendis ja väljundis tehete tähistamiseks kasutatud erinevaid mooduseid. Sisendi puhul kasutatakse sümboleid ja sümbolikombinatsioone, mis on välja toodud tabelis 4, väljundi puhul kasutatakse eituse, konjunktsiooni ja disjunktsiooni jaoks vastavalt osavalemi kohal olevat joont, korrutamismärki ja liitmismärki. Teisendamiseks ei ole võimalik valida sisestatud valemist väiksemat osavalemit. Kuna tulemuste kuvamisel näidatakse lihtsustamisel kasutatud samme, on sarnasus käsitsi teisendamisega suurem kui eelmistes alapeatükkides analüüsitud rakenduste puhul. Sellegipoolest on rakenduses ette määratud, milliseid teisendusi ja millises järjekorras tehakse. Sellepärast on ka selles rakenduses teisenduste tegemine erinev käsitsi teisendamisest.

3.5 First order logic tool

Rakendusega First order logic tool[10] analüüsitakse lause- ja predikaatarvutuse valemeid. Rakendus võtab sisendiks valemi ning näitab valemi iga osa korral, millise valemi liikmega on tegu: lausemuutuja, indiviidmuutuja, konstandi, funktsiooni või predikaadiga. Lisaks sellele teisendatakse sisestatud valem automaatselt kolmele erinevale prefikskujule sõltuvalt prefiksile järgnevast valemi kujust: tavaline kuju, DNK ja KNK. Tavalise kuju puhul on kvantorid toodud prefiksina ette, aga kvantoritele järgnev osa ei ole viidud normaalkujule. Valemite sisestamisel on abiks loogikasümbolitega nupud. Lisaks sellele on võimalik sümboleid sisestada ka klaviatuurilt või kopeerides ja kleepides. Vastavad sümbolid on välja toodud tabelis 5.

Tabel 5. Loogikatehete sisestamine rakenduses First order logic tool.

Tehe/kvantor	¬	∧	∨	⇒	⇔	∀	∃
Sisestusviisid	!	&	|	→, ->	↔, <->	\A	\E

Loogikatehete ja kvantorite sisestamine klaviatuurilt on kiire. Eituse, konjunktsiooni ja disjunktsiooni puhul on minimaalselt vaja sisestada üks sümbol, implikatsiooni ja kvantorite korral kaks ning ekvivalentsi korral kolm sümbolit. Sisendlahtris teisendatakse nii tehete kui ka kvantorite tähised ühtlasele kujule. Samas tuleb valemite sisestamisel arvestada sellega, et teatud juhtudel peab sisestatud sümbolile järgnema tühik, et vastavat sümbolit ei arvataks kokku sellele järgneva osaga. Näiteks sisendeid $\forall xA(x)$ ja $\forall x\ A(x)$ tõlgendatakse

[10] https://mamo.dev/first-order-logic-tool (11.04.2022)

erinevalt. Esimesel juhul loetakse *xA* muutujaks ja viimane *x* lausemuutujaks. Teisel juhul loetakse esimene *x* muutujaks ja *A* ühekohaliseks predikaadiks. Sisendi tuvastamine ei tööta alati korrektselt, näiteks sisendi $A \rightarrow B \leftrightarrow \neg C$ korral näidatakse kasutajale teadet oodatava sisendi lõpu kohta (ingl *Expected end of input*). Kui valem on sisestatud ja korrektselt tuvastatud, kuvatakse kasutajale nimetatud normaalkujud. Eraldi ei tooda välja tulemuse saamiseks kasutatud teisendussamme. Sisestatud valemist vaid väiksema osavalemi valimine teisendamiseks ei ole võimalik. Võrreldes eelmistes alapeatükkides analüüsitud tööriistadega oli rakenduses First order logic tool sisendi puhul parem see, et loogikatehete tähistused ühtlustati juba sisendi kirjutamise ajal. Kuna tehtud teisendussamme ei kuvata ja teisendada pole võimalik väiksemat osavalemit, ei sarnane rakenduses teisenduste tegemine siiski käsitsi teisendamisega.

3.6 Lausearvutuse ja predikaatloogika valemite teisendamise õpiprogramm

Lausearvutuse ja predikaatloogika valemite teisendamise õpiprogrammi on koostanud Vaiksaar [3] ning seda on hiljem täiendanud Stroom [4]. See programm võimaldab kasutajal luua ja lahendada teisendusülesandeid. Ülesannete koostamiseks ja lahendamiseks on kaks erinevat keskkonda: vastavalt *koostaja.jar* ja *teisendaja.jar*. Ülesannete loomisel on kasutajal võimalik valida erinevate ülesandetüüpide vahel nii lause- kui ka predikaatarvutuse korral. Lausearvutuse valemite puhul on muu hulgas valikus ka DNK ja KNK ning predikaatarvutuse valemite korral prefikskuju. Mõlemal juhul on valikus ka vaba teisendamise variant. Teisendatav valem on võimalik ette anda nii ise kui ka lasta programmil teatud tingimustele vastav valem genereerida. Nimetatud tingimused on erinevate tehete, muutujate ja kvantorite arv. Lisaks tuleb ülesande koostamisel ette anda erinevate vigade lubatavad arvud. Kui kasutaja tahab valemi ise ette anda, siis saab ta loogikasümboleid sisestada nii klaviatuurilt kui ka nuppude abil. Tabelis 6 on toodud klahvid, millega loogikatehteid ja kvantoreid tähistavaid sümboleid sisestatakse.

Tabel 6. Loogikatehete sisestamine õpiprogrammis.

Tehe/kvantor	¬	∧	∨	⇒	⇔	∀	∃
Sisestusviisid	F1	F2	F3	F4	F5	F6	F7

Võrreldes eelmistes alapeatükkides analüüsitud rakendustega, on selle programmi puhul loogikasümbolite sisestamine kõige kiirem, sest need saab sisestada ühe klahvivajutusega.

Kuna loogikasümbolite tähiseid on iga tehte ja kvantori jaoks ainult üks, on valemite kuvamisel sümbolite kasutus ühtlane. Kui ülesanded on sisestatud, luuakse neile vastav ülesandekogu fail.

Teisenduste tegemiseks tuleb lahendajal luua uus õpilasfail, mille loomisel peab sisestama rühma ning ees- ja perekonnanime ning lisaks valima ülesandefaili. Seejärel näidatakse kasutajale ülesandekogus sisalduvaid ülesandeid, ning kasutaja saab valida neist sobiva ning asuda seda lahendama. Ülesande lahendamine seisneb teisendussammude tegemises ja ülesanne loetakse sooritatuks, kui valem on teisendatud ülesandes nõutud kujule, kusjuures teisendamisel ei tohi teha rohkem vigu, kui ülesande koostaja ette määras. Kui lahendaja eesmärk ei ole teisendussammude tegemise käigus ülesannet sooritatud saada, on võimalik käsitsi teisendamist jäljendada kasutaja enda soovi kohaselt, kusjuures teisendusreegleid saab rakendada nii tervele valemile kui ka väiksemale osavalemile. Lahendamise ajal kuvatakse teisenduste käigus saadud valemid üksteise all musta värviga, kuid ei ole näha, milline osavalem valiti ning millist teisendusreeglit rakendati. Osavalem teisendamiseks valitakse kõige alumisena kuvatud valemist. Lahenduskäigu kõik sammud koos valitud osavalemite, rakendatud reeglite ja kokkuvõtva infoga teisenduskäigu kohta salvestatakse õpilasfaili. Salvestatut on hiljem võimalik teisenduskeskkonnas tulemuste alt vaadata. Selles programmis teisenduste tegemine sarnaneb võrreldes eelmiste rakendustega kõige rohkem käsitsi teisendamisega, kuid selleks, et valemit saaks hakata teisendama, peab võrreldes eelmiste rakendustega tegema palju eeltööd.

3.7 Analüüsi kokkuvõte

Selles peatükis analüüsiti kuut tööriista, millega saab lause- ja/või predikaatarvutuse valemeid teisendada. Kui võrrelda nende kasutusvõimalusi käsitsi teisendamisega, siis on selleks analüüsitutest parim valik viimasena uuritud lausearvutuse ja predikaatloogika valemite teisendamise õpiprogramm, mis võimaldab teisendada nii lause- kui ka predikaatarvutuse valemeid, valida teisendamiseks väiksemaid osavalemeid, rakendada valitud osavalemile teisendusreegleid ja tehtut salvestada. Selle rakenduse suurim puudus on see, et valemi sisestamine ja valemi teisendamine toimuvad erinevates keskkondades. Ülejäänud tööriistade puhul peeti valemite sisestamist mugavamaks seetõttu, et seda ei pidanud võrreldes teisendamisega tegema erinevas keskkonnas. Samas on neis käsitsi teisendamise jäljendamine võimatu, sest valemile pole võimalik rakendada teisendusreegleid, vaid need viiakse ette määratud kujule. Lisaks ei ole nende puhul võimalik teisendamiseks valida väiksemat

osavalemit ega saadud tulemust rakenduse abil salvestada. Nelja tööriista puhul oli võimalik teisendada ainult lausearvutuse valemeid. Järgmises peatükis tuuakse antud analüüsi põhjal välja loodava rakenduse nõuded.

4. Nõuded rakendusele

Eelmises peatükis analüüsiti kuut loogikavalemite teisendamisega seotud tööriista, lähtudes eelkõige valemite sisestamise mugavusest ja teisendussammude tegemise vabadusest. Analüüsis rõhutati peamiste puudustena teisendussammude tegemise ja võimalike tulemuste piiratust ning valemi sisestamise keerulisust. Et neid puudusi selle töö raames koostatava rakenduse puhul leevendada, on valmivale rakendusele seatud funktsionaalsed ja mittefunktsionaalsed nõuded, millega arenduse käigus arvestatakse.[11]

4.1 Funktsionaalsed nõuded

Rakenduse põhilised kasutajad on matemaatilist loogikat käsitlevate kursuste õppejõud ja tudengid, kuid seda võivad kasutada ka teised. Rakenduse kasutamisel ei ole oluline, kas kasutaja on tudeng või õppejõud, seega on rakendusel üks kasutajaroll, millele peavad olema võimaldatud järgnevad tegevused.

1) Sisestada on võimalik lause- või predikaatarvutuse valem, et sellele rakendada teisendusi.
2) Sisestatud valemist saab valida osavalemi, mis vastab tervele valemile või on väiksem osavalem, et see osavalem teisendada uuele kujule.
3) Valitud osavalemile saab vastavalt selle kujule rakendada lause- ja/või predikaatarvutuse põhisamaväärsuseid, et uus kuju oleks eelmisega samaväärne.
4) Lausearvutuse valemi saab viia kõikidele järgmistele kujudele: TDNK, TKNK, DNK ja KNK.
5) Predikaatarvutuse valemi saab viia prefikskujule.
6) Kustutada saab viimase teisendussammu, et vale teisendus eemaldada.
7) Kustutada saab kõik teisendussammud, et tööd uuesti alustada.
8) Vahetulemuse saab salvestada, et teisendustega hiljem jätkata.
9) Salvestatud vahetulemusega saab tööd jätkata, et ei peaks kõiki juba korra tehtud teisendussamme uuesti tegema.
10) Lõpptulemuse saab salvestada, et seda kaastudengite ja õppejõududega jagada.
11) Lahenduskäik on võimalik viia TeXi kujule, et seda olemasolevatesse .tex failidesse lisada.

[11] Lisaks peatükis 3 toodud analüüsile arutati rakenduse nõuded ka töö juhendajaga.

4.2 Mittefunktsionaalsed nõuded

Selleks, et rakendus oleks võimalikult kasutajasõbralik, peab see rahuldama järgmisi mittefunktsionaalseid nõudeid.

1) Rakendus peab olema kättesaadav veebi kaudu, et kasutaja ei peaks oma arvutisse eraldi tarkvara paigaldama hakkama.
2) Rakenduses peab olemas olema kasutusjuhend, et kasutaja saaks vajaliku info kiiresti kätte.
3) Tehtud teisendussammude puhul tuleb kuvada info valitud osavalemi, valitud teisendusreegli ja saadud tulemuse kohta.
4) Vea korral tuleb kasutajat sellest teavitada. Olulisimad võimalikud vead on järgmised:
    a) sisestatud osavalem ei ole korrektse struktuuriga;
    b) teisendamiseks on valitud osa valemist, mis ei ole korrektne osavalem;
    c) korrektselt valitud osavalemile rakendatakse reeglit, mis ei sobi kokku osavalemiga;
    d) reeglit rakendatakse siis, kui osavalem on valimata.
5) Rakendus peab andma informatsiooni kiirklahvide kasutamise kohta.

Selles peatükis toodi välja rakenduse nõuded, millega piiritleti rakenduse funktsionaalsus ja seati eesmärgiks saavutada suurem kasutajamugavus võrreldes kolmandas peatükis analüüsitud tööriistadega. Järgmises peatükis kirjeldatakse rakenduse arhitektuuri ja arendamisel kasutatud tehnoloogiaid.

5. Rakenduse arhitektuur ja kasutatud tehnoloogiad

Loodud veebirakendusel puuduvad andmebaas ja tagasüsteem (ingl *back end*), seega on arhitektuurilt tegu eessüsteemirakendusega (ingl *front end application*). Rakenduse loomisel lähtuti sellest, et kõik vajalik loogikavalemite teisendamiseks ja tulemuste salvestamiseks mahuks ühele lehele. Ühelehelise rakenduse (ingl *Single-page application*) puhul laaditakse kasutaja veebilehitsejasse ainult üks lehekülg, mille sisu kasutamise ajal uuendatakse [5]. Selles peatükis põhjendatakse rakenduse arendamiseks kasutatud raamistiku (ingl *framework*) valikut. Lühidalt kirjeldatakse, milliste valitud raamistikuga seotud tehnoloogiate abil on võimaldatud sümbolite sisestamine, teisendussammude kustutamine, töö allalaadimine ja kuvatava teksti hoidmine. Veidi põhjalikumalt kirjeldatakse tehnoloogiat, mida on kasutatud nii kasutajasisendi kontrollimiseks kui ka teisenduste tegemiseks.

5.1 Eessüsteemiraamistik

Eessüsteemiraamistiku valik langetati kolme variandi vahel: React.js, Angular ja Vue.js. Kõik nimetatud raamistikud kuuluvad 2021. aasta seisuga viie enimkasutatu hulka [6]. Samad raamistikud on viie parima üheleheliste rakenduste arendamiseks kasutatava raamistiku seas välja toonud ka Patel [7]. Wohlgethan [8] on oma bakalaureusetöös teinud ülevaate samuti neist kolmest raamistikust. Ühe olulise aspektina on raamistiku Vue.js puhul nimetatud kahesuunalist andmete sidumise direktiivi *v-model*, mis võimaldab vajalike muutujate väärtusi muuta nii skriptisiseselt kui ka kasutajal sisestusvälja kaudu. Lisaks on autor välja toonud, et tegu on kergelt õpitava raamistikuga. Raamistiku Vue.js samu tugevusi rõhutas oma bakalaureusetöös ka Breedis [9]. Ta uuris mainitud kolme raamistikku vähekogenud arendaja seisukohast ja arvas kokkuvõttes algajaile parimaks variandiks raamistiku Vue.js. Selle töö juures on direktiiv *v-model* kasulik selleks, et kasutajasisend siduda muutujaga, milles hoitakse teisendatava loogikavalemi väärtust. Kasutaja saab seda muuta loogikavalemi sisestamisel ning skriptiga muudetakse väärtust teisenduste tegemise ajal. Teise aspektina nimetatud kerge õpitavus aitas samuti kaasa otsusele valida eessüsteemiraamistikuks Vue.js (versioon 3.2.31). Rakenduse loomise ajal tugineti valitud raamistiku dokumentatsioonile [10].

5.2 Sümbolite sisestamine

Rakenduses on kasutajale võimaldatud sümbolite sisestamine kolmel erineval moel: TeXi koodiga, numbrite abil ja sümbolinuppudega. Et nuppude abil sümbolite sisestamine oleks skriptis lihtsamalt teostatav, on kasutatud paketihalduri npm (Node Package Manager) paketti insert-text-at-cursor[12] (versioon 0.3.0), mille abil saab tekstiväljadesse andmeid sisestada kursori positsioonile. Kui tekstiväljas on märgistatud mingi osa tekstist, siis asendatakse uute andmetega kogu märgistatud osa.

5.3 Teisendussammude kustutamine

Selleks, et kasutajatel oleks võimalik teisendussamme tagasi võtta, peab rakenduses hoidma tehtud teisenduste ajalugu. Järgnev lõik põhineb teegi Vuex dokumentatsioonil [11].

Vuex on teek, mille abil hoitakse ja muudetakse andmeid raamistikku Vue.js kasutavates rakendustes. Sellised rakendused võivad koosneda paljudest komponentidest, mille vahel andmete edastamine ühelt komponendilt teisele kohandatud tunnuste (ingl *prop*) abil võib muutuda tülikaks. Teek Vuex aitab andmed, mida kasutatakse mitmes komponendis, ja nende muutmisega seotud funktsioonid koondada tsentraalsesse lattu (ingl *store*), mis tagab rakenduse oleku (ingl *state*) korrektsuse. Välja on toodud ka see, et kuigi selle teegi kasutamine toob lisaks raamistiku Vue.js enda põhimõtetele juurde uusi, aitab see pikas perspektiivis projekte produktiivsemalt hallata.

Kuna loodud Vue.js rakendus koosneb mitmest komponendist, mis töötlevad samu andmeid, on teegi Vuex (versioon 4.0.2) kasutamine vajalik. Rakenduse arendamisel lähtuti teegi Vuex dokumentatsioonist [11].

5.4 Vahe- ja lõpptulemuste salvestamine

Vahe- ja lõpptulemuste salvestamiseks kasutatakse rakenduses paketihalduri npm pakette file-saver[13] (versioon 2.0.5) ja pdfmake[14] (versioon 0.2.5). Esimene võimaldab keele JavaScript abil koostada ja alla laadida erinevates vormingutes faile, muu hulgas ka vormingutes .json ja .tex. Selleks, et tulemuse saaks alla laadida ka. pdf vormingus, on kasutatud teisena

[12] https://www.npmjs.com/package/insert-text-at-cursor (22.04.2022)

[13] https://www.npmjs.com/package/file-saver (22.04.2022)

[14] https://www.npmjs.com/package/pdfmake (22.04.2022)

nimetatud paketti. Kuna rakenduse abil teisendatavad valemid sisaldavad sümboleid, mida paketiga pdfmake kaasasolevad šriftid ei võimaldanud tulemusfailis kuvada, lisati sellele paketile eraldi šrift Noto Sans Math[15]. Paketi pdfmake kasutamisel rakenduses lähtuti pdfmake dokumentatsioonist [12].

5.5 Kuvatud teksti hoidmine ja tõlkevõimalus

Rakendus on esialgu koostatud vaid eestikeelsena. Selleks, et vältida eestikeelse teksti hoidmist HTML (HyperText Markup Language) märgendite vahel, ning võimaldada rakenduse tõlkimine teistesse keeltesse lihtsamalt, on kogu kuvatav tekst salvestatud eraldi .json vormingus faili. Selle faili sisu lisatakse rakendusse paketihalduri npm paketi vue3-i18n[16] (versioon 1.1.4) abil.

5.6 Sisendi töötlemine

Selleks, et kasutajasisendina saadud loogikavalemit töödelda, on see vaja teisendada sobilikule kujule. Parr [13] on loonud tööriista ANTLR v4 (ANother Tool for Language Recognition version 4), mis grammatika ehk formaalse keele kirjelduse põhjal koostab lekseri ehk leksilise analüsaatori ja parseri ehk süntaksianalüsaatori. Lekseriga analüüsitakse sisendit ja tagastatakse lekseemide (ingl *token*) jada, mille põhjal parseriga koostakse nimetatud grammatikale vastavaid süntaksipuid. Selleks, et saadud süntaksipuid läbida kohandatud viisil, on võimaldatud koos lekseri ja parseriga luua ka mehhanism *visitor*, millega saab muuta süntaksipuu läbimise järjekorda ja kutsuda välja iga grammatika reegli jaoks eraldi funktsiooni. Lisaks on autor kirjeldanud, kuidas kohandada veateateid, mida lekseri ja parseriga väljastatakse. Sama tööriista on predikaatarvutuse valemite samaväärsust kontrolliva teegi valmistamiseks kasutanud Saar [14] ning selle varasemat versiooni staatilise koodianalüsaatori loomiseks Jaggo [15]. Lisaks sellele on vahendiga ANTLR v4 juba loodud palju grammatikaid, mille eeskuju oli loogikavalemite teisendusredaktori koostamisel võimalik kasutada. Nimetatud grammatikate hulgas on ka spetsiaalselt predikaatarvutuse valemite äratundmiseks loodud grammatika [16].

Tööriista ANTLR v4 kirjeldatud omadused olid valiku langetamisel olulised sellepärast, et see võimaldab luua loogikavalemite teisendamiseks sobiliku mehhanismi *visitor* ja

[15] https://fonts.google.com/noto/specimen/Noto+Sans+Math?query=math (22.04.2022)

[16] https://www.npmjs.com/package/vue3-i18n (22.04.2022)

kohandatud veateadete käsitlemisega anda kasutajale vajalikku informatsiooni nii loogikavalemi sisestamise kui ka teisenduste tegemise ajal. Neil põhjustel on selle lõputöö raames kasutajasisendi töötlemiseks kasutatud vahendit ANTLR v4. Rakenduse arendamise käigus tugineti juhenditele Parri [13] raamatus ja spetsiaalselt keelele JavaScript mõeldud ANTLR v4 dokumentatsioonile [17].

Selles peatükis kirjeldati loodud rakenduse arhitektuuri ning toodi välja kasutatud tehnoloogiad. Tegu on ühelehelise Vue.js rakendusega, kus olekut hoitakse teegi Vuex abil loodud tsentraalses laos ja kasutajasisendina saadud loogikavalemi töötlemiseks kasutatakse tööriista ANTLR v4. Lisaks on kasutusel väiksemad paketihalduri npm paketid, mis hõlbustavad sümbolite sisestamist, failide loomist ja salvestamist ning eestikeelse teksti lisamist rakendusse. Järgmises peatükis kirjeldatakse lühidalt, kuidas rakendust arendamise käigus testiti.

6. Testimine

Selleks, et rakenduse lõplik versioon selle töö raames saaks võimalikult korrektne ja kasutajasõbralik, testiti rakendust nii automaatselt kui ka paluti abi potentsiaalsetelt kasutajatelt. Selles peatükis kirjeldatakse automaattestimist ja kasutajatestimist ning selle tulemusi.

6.1 Automaattestimine

Automaattestimiseks lisati rakenduse loomisel kaasa JavaScripti testraamistikud Jest[17] ja Cypress[18]. Neist esimest kasutati üksuste testimiseks (ingl *unit testing*) ja integratsioonitestimiseks (ingl *integration testing*) ning teist läbivtestimiseks (ingl *end-to-end testing*). Automaattestide ulatuse mõõtmisel jälgiti lausete katvust (ingl *statement coverage*) ja harude katvust (ingl *branch coverage*).

Üksuste ja integratsioonitestimisel oli põhiline eesmärk kontrollida teisenduste tegemise ja sulgude lisamise korrektsust. Sellest osast jäeti välja need funktsioonid, mille korral oli rakendusel vaja kasutada elemente rakenduse veebilehelt. Joonisel 2 on toodud ülevaade lausete ja harude katvusest üksuste ja integratsioonitestimisel.

File		Statements		Branches	
ANTLR/leftAssocGrammar		78.52%	424/540	46.15%	66/143
ANTLR/leftAssocGrammar/visitors/coloringParens		100%	31/31	100%	2/2
ANTLR/leftAssocGrammar/visitors/freeVariables		100%	47/47	100%	6/6
ANTLR/leftAssocGrammar/visitors/predicate		100%	351/351	100%	198/198
ANTLR/leftAssocGrammar/visitors/propositional		100%	614/614	100%	288/288
ANTLR/leftAssocGrammar/visitors/validatingSymbols		100%	56/56	100%	20/20
ANTLR/rightAssocGrammar		65.78%	348/529	31.91%	45/141
js		99.42%	343/345	98.92%	275/278
store		100%	55/55	100%	18/18

Joonis 2. Lausete ja harude katvus üksuste ja integratsioonitestimisel.

Kaustades *ANTLR/leftAssocGrammar* ja *ANTLR/rightAssocGrammar* on tööriistaga ANTLR v4 loodud failid, mille sisu läheb rakenduse kasutamise ajal vaja ainult osaliselt. Mittevajaminevaid osasid eraldi ei testitud, kuid need jäeti rakenduse lähtekoodi siiski alles.

[17] https://jestjs.io/ (22.04.2022)

[18] https://www.cypress.io/ (22.04.2022)

Läbivtestimise eesmärk oli kontrollida, kas rakendus töötab tervikuna. Nende testidega kontrolliti järgnevat.

1) Kõik vajalikud elemendid teisenduste tegemiseks on olemas.
2) Kasutusjuhendi vaatamine on võimalik.
3) Teisendussammude tegemine töötab, kui valida terve valem või väiksem osavalem.
4) Teisendussammude kustutamine ühekaupa ja kõik korraga töötab.
5) Vigade korral kuvatakse veateated.
6) Sümbolite sisestamine toimib nuppude abil ja klaviatuurilt.
7) Teisenduste ajal või failide allalaadimisel töötab kasutajalt sisendi küsimine.

Joonisel 3 on toodud lausete ja harude katvus läbivtestimise põhifookuses olnud failide puhul.

File		Statements		Branches	
ConvButton.vue		100%	58/58	100%	17/17
ConvTypeMarker.vue		100%	1/1	100%	0/0
DeleteAllConfirmationPrompt.vue		100%	2/2	100%	0/0
DeleteButtons.vue		100%	26/26	100%	7/7
Downloaders.vue		48.27%	28/58	57.14%	8/14
ErrorMessages.vue		100%	5/5	100%	3/3
FilenamePrompt.vue		100%	8/8	100%	0/0
Help.vue		100%	2/2	100%	0/0
Main.vue		100%	5/5	100%	0/0
NewSubformulaPrompt.vue		100%	33/33	100%	3/3
Output.vue		100%	4/4	100%	0/0
PredButtons.vue		100%	1/1	100%	0/0
PropButtons.vue		100%	1/1	100%	0/0
SymbolButtons.vue		100%	7/7	100%	3/3
UserInput.vue		64.17%	43/67	60%	9/15

Joonis 3. Lausete ja harude katvus läbivtestimisel.

Failide *Downloaders.vue* ja *UserInput.vue* puhul ei testitud automaatselt failide üles- ja allalaadimisega seotud funktsioone. Neid testiti kasutajatestimise käigus.

6.2 Kasutajatestimine

Selleks, et enne rakenduse valmimist saaksid seda katsetada ka potentsiaalsed kasutajad, korraldati aprillis 2022 kahenädalase perioodi jooksul arvutiteaduse instituudi ning matemaatika ja statistika instituudi üliõpilaste ning matemaatilise loogika õpetamisega seotud

õppejõudude[19] seas kasutajatestimine. Kasutajatel paluti rakenduse funktsionaalsusi katsetada ning vastata küsimustikule. Uuriti eelkõige kasutajamugavust ning küsiti soovitusi rakenduse parandamiseks. Kasutajatestimise küsimustik ja saadud vastused on kaasapandud failide hulgas (lisa 1).

Küsimustikule vastas kokku 24 inimest, kellest 79,2% olid tudengid, 12,5% õppejõud ja 8,3% väljastpoolt Tartu Ülikooli. Kasutajatest 91,7% arvas, et rakenduses on lihtne või väga lihtne navigeerida, kasutusjuhendi lihtsuse puhul oli sama näitaja 79,2%. Loogikasümbolite sisestamine oli 91,7% ning teisendussammude kuvamisest aru saamine 83,3% kasutajate jaoks samuti lihtne või väga lihtne. Nendest, kes olid varem mõnda rakendust loogikavalemite teisendamiseks kasutanud, arvas 72,2%, et nende kogemus selle rakendusega võrreldes eelnevalt kasutatud rakendustega oli parem või palju parem. Samas puutus 58,3% kasutajatest teisenduste tegemisel kokku erinevate probleemidega ning 70,8% kasutajatest lisasid ka kommentaare ja soovitusi. Põhilised probleemkohad ja parandusettepanekud olid järgmised:

1) samaväärsuste rohkuse tõttu vajaliku reegli leidmine;
2) see, et reeglid ei mahtunud korraga vaatevälja;
3) sulgude lugemine;
4) kahe samaväärsuse puudumine ($F \wedge \neg F \equiv 0$ ja $F \vee \neg F \equiv 1$);
5) veateadete üldisus, puudumine ja kuvamise asukoht;
6) ainukese või kõikide teisendussammude kustutamisel kustub ka algne sisend, milles oli vaja ainult üks koht muuta;
7) sulgude lisamine ei toimu alati korrektselt;
8) seotud muutujate ümbernimetamisel ei arvestata valemi kõikide muutujatega;
9) liiga range indiviidmuutujate ning konstant- ja funktsionaalsümbolite kasutus.

Vaatamata sellele, et kasutajate hulk, kes rakendust testisid, oli väike, saadi siiski väärtuslikku informatsiooni, millega rakenduse viimistlemisel arvestati.

Selles peatükis kirjeldati lühidalt rakenduse arendamise käigus tehtud testimist. Automaatning kasutajatestimine kokku aitasid kaasa sellele, et rakenduses oleks vähem puudujääke ja teisenduste tegemine oleks korrektne. Järgmises peatükis kirjeldatakse loodud rakendust ja tuuakse välja selle eelised võrreldes kolmandas peatükis analüüsitud rakendustega. Lisaks mainitakse, mida muudeti kasutajatestimise tulemuste analüüsi põhjal.

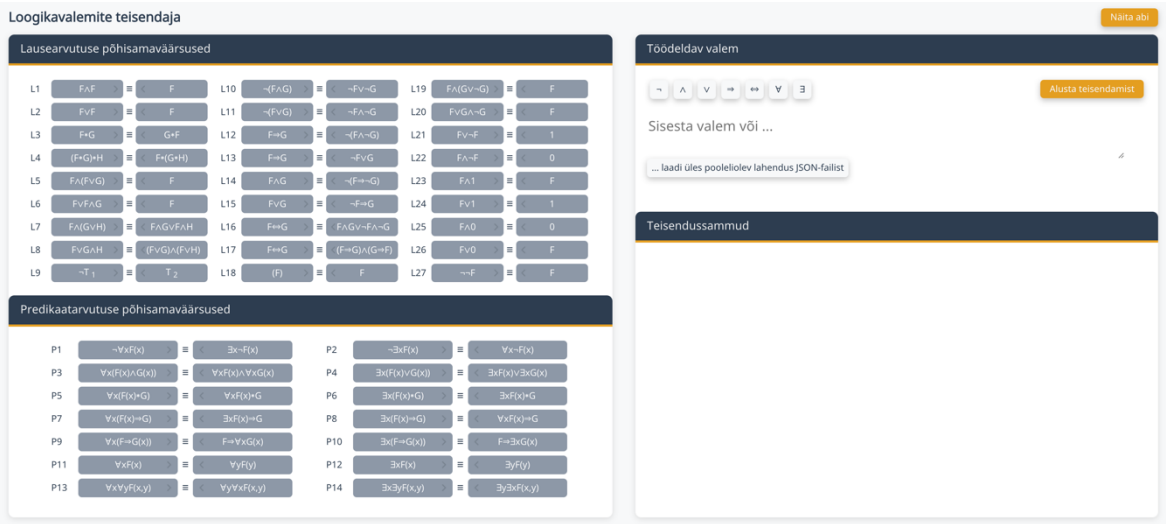
[19] Õppejõudude info on saadud töö juhendajalt.

7. Valminud rakenduse kirjeldus

Töö käigus valmis veebirakendus Loogikavalemite teisendaja[20]. Selles peatükis kirjeldatakse valminud rakendust ja selle funktsionaalsusi. Kõigepealt kirjeldatakse rakenduse põhiosa. Alapeatükkides 7.2–7.7 tehakse läbi näide mille käigus tutvustatakse rakenduse kasutamist. Alustatakse sellega, kuidas valemeid sisestatakse. Teisendusreeglite rakendamise osas kirjeldatakse põhjalikumalt, mida enne reegli rakendamist kontrollitakse ja kuidas teisenduse tegemine toimub. Edasi selgitatakse teisendussammude kuvamist ja kustutamist. Seejärel kirjeldatakse vahe- ja lõpptulemuste salvestamise võimalusi. Juhul, kui kasutajatestimise analüüsi põhjal midagi muudeti, mainitakse ka seda. Selle peatüki viimases alapeatükis tuuakse välja rakenduse eelised ja edasiarendusvõimalused.

7.1 Rakenduse põhiosad

Valminud veebirakendus koosneb kahest põhiosast: vasakul on teisendusreeglite rakendamiseks vajalikud nupud ning paremal on töödeldav valem koos tehtud teisendussammudega. Joonisel 4 on näha rakenduse üldvaade.



Joonis 4. Rakenduse üldvaade.

Enne kasutajatestimist oli valemi sisestamise osa teisendusnuppude kohal ja paremal oli ainult teisendussammude kuvamise lahter. Kuna elementide paigutus oli kasutajate jaoks

[20] Rakendus on kättesaadav leheküljelt https://teisendaja.vercel.app/. Rakenduse lähtekood on kaasapandud failide hulgas (lisa 1).

üks põhilisi murekohti, siis liigutati valemite sisestamise osa paremale. Nuppude arvu vähendamiseks viidi omavahel kokku järgmised eraldi paiknenud reeglid:

1) $F \wedge G \equiv G \wedge F$, $F \vee G \equiv G \vee F$, $F \Leftrightarrow G \equiv G \Leftrightarrow F$ (reegel L3);
2) $(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$, $(F \vee G) \vee H \equiv F \vee (G \vee H)$, $(F \Leftrightarrow G) \Leftrightarrow H \equiv F \Leftrightarrow (G \Leftrightarrow H)$ (reegel L4);
3) $\neg 1 \equiv 0$, $\neg 0 \equiv 1$ (reegel L9);
4) $\forall x(F(x) \wedge G) \equiv \forall x F(x) \wedge G$, $\forall x(F(x) \vee G) \equiv \forall x F(x) \vee G$ (reegel P5);
5) $\exists x(F(x) \wedge G) \equiv \exists x F(x) \wedge G$, $\exists x(F(x) \vee G) \equiv \exists x F(x) \vee G$ (reegel P6).

Veel lisati puuduvad reeglid samaväärsuste $F \wedge \neg F \equiv 0$ ja $F \vee \neg F \equiv 1$ rakendamiseks. Kõik rakenduses olemasolevad lause- ja predikaatarvutuse samaväärsused on toodud lisas 2.

Rakendusele on ka lisatud kasutusjuhend, mida näeb, kui vajutada nupule „Näita abi”. Kasutusjuhendist leiab järgmise info:

1) kuidas valemeid sisestada ja teisendussamme teha;
2) milliseid sümboleid predikaatide, termide ja tõeväärtuste jaoks kasutatakse;
3) kuidas laaditakse alla vahe- ja lõpptulemus;
4) täpsustused kommutatiivsuse ja assotsiatiivsusega arvestamisel.

Kasutusjuhendi sisu on toodud lisas 3.

Rakenduse parempoolses osas, kus tegeletakse teisendustega alustamise, teisendussammude kustutamise, tulemuste allalaadimise ja abi näitamisega, on võimaldatud kõiki nimetatud tegevusi teha ka kiirklahvidega (enne kasutajatestimist olid kiirklahvid ainult sümbolite sisestamiseks). Nende kohta saab informatsiooni, kui hoida hiirt vastava nupu peal. Kuna teisendusreegleid tähistavate nuppude arv on suur, siis teisenduste tegemiseks kiirklahve ei lisatud.

7.2 Valemite sisestamine

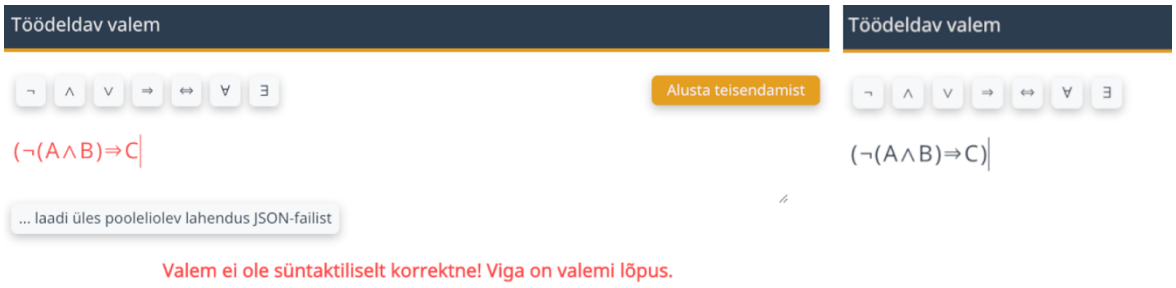
Enne teisenduste tegemise juurde asumist tuleb sisestada loogikavalem. Lausemuutujad, predikaadid, indiviidmuutujad, konstantsümbolid ja tõeväärtused sisestatakse kõik klaviatuurilt. Loogikasümbolite sisestamiseks on lisaks ka nupud. Sisestamise ajal teisendatakse loogikasümbolite puhul numbrid ja TeXi kood automaatselt ümber, et sümbolikasutus valemi sees oleks ühtlane. Tehetele ja kvantoritele vastavad sisestusvõimalused ja asendused on toodud tabelis 7. Enne kasutajatestimist oli sisestusvõimalusi iga tabeli rea kohta kaks

(tabeli 7 teises tulbas esimesed kaks varianti). Kasutajatestimise järel lisati binaarsetele tehetele veel võimalusi.

Tabel 7. Loogikatehete ja kvantorite sisestamine valminud veebirakenduses.

Tehe/kvantor	Number ja TeXi kood	Asendus
Eitus	2, \neg	¬
Konjunktsioon	3, \land, \wedge, \&	∧
Disjunktsioon	4, \lor, \wee	∨
Implikatsioon	5, \Rightarrow, \implies	⇒
Ekvivalents	6, \Leftrightarrow, \iff	⇔
Üldisuskvantor	7, \forall	∀
Olemasolukvantor	8, \exists	∃

Valemi sisestamise ajal kontrollitakse automaatselt, kas selle struktuur vastab grammatikas (lisa 4) määratud struktuurile. Kui mitte, kuvatakse valemit punase värvi ja täpsustava veateatega, vastasel juhul tumesinisena (joonis 5).



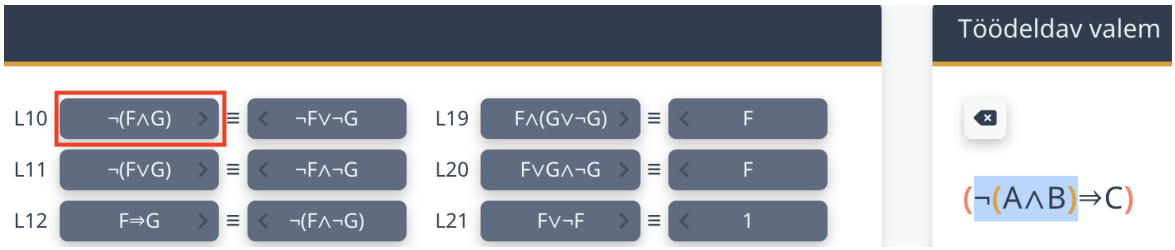
Joonis 5. Vasakul vigaselt sisestatud valem (viimane sulg puudu), paremal korrektselt sisestatud valem.

Veateates antakse info grammatika põhjal koostatud süntaksipuu analüüsis leitud esimese vea kohta. Kui viga satub valemi keskele, kuvatakse ka sümbol ja selle asukohaindeks valemis. Veel antakse veateatega infot juhul, kui mõne predikaadi või funktsiooni argumentide arv mitme esinemise puhul ei klapi või funktsionaalsümbolit on kasutatud ka indiviidmuutuja või konstantsümbolina. Enne kasutajatestimist olid indiviidmuutujate ning konstant- ja funktsionaalsümbolite jaoks eraldatud kindlad tähevahemikud. Kasutajatelt saadud tagasiside põhjal lubati neis rollides kasutada kõiki ladina tähestiku väiketähti.

Pärast valemi sisestamist tuleb vajutada nupule „Alusta teisendamist“, mille abil aktiveeritakse teisendusreeglite rakendamiseks vajalikud nupud ja keelatakse kasutajal valemi edasine muutmine. See on vajalik, et teisendussammude rakendamise ajal ei oleks kasutajal võimalik valemit viia sellisele kujule, mis ei ole esialgselt sisestatud valemiga samaväärne.

7.3 Teisendusreeglite rakendamine osavalemile

Kui valem on sisestatud ja teisendamist alustatud, saab osavalemi töödeldava valemi lahtrist hiirega märgistades. Kasutajatestimisel leidis mitu kasutajat, et sulgude vahel olevat osavalemit on keeruline tuvastada juhul, kui mitu sulgu satub üksteise kõrvale. Et seda oleks lihtsam teha, on lisatud osavalemit alustavale ja lõpetavale sulule sama värv. Termide ümber ja sees asuvate sulgude värvi ei muudeta. Kui osavalem on märgitud, tuleb valida sobiv reegel ja vajutada vastavale nupule. Reeglid on jaotatud lausearvutuse põhisamaväärsusteks ja predikaatarvutuse põhisamaväärsusteks. Iga reegli juures on seda tähistav kood ja kaks nuppu, mille vahel on samaväärsusmärk. Reegli rakendamisel tuleb lähtuda valitud osavalemi kujust, mille rõhutamiseks on nuppudele lisatud noolesümbolid. Osavalemi valimine ja reegli rakendamine on näidatud joonisel 6.



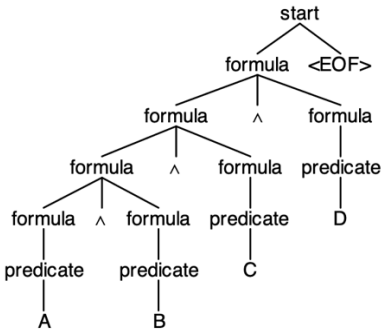
Joonis 6. Valitud osavalem märgitud hiire abil (helesinine taust). Punane joon on ümber tõmmatud valitud teisendusreeglile koodiga L10.

Enne, kui valitud osavalemile reeglit rakendatakse, kontrollitakse järgnevat.

1) Osavalem on valitud.
2) Osavalem on valitud töödeldava valemi alt, mitte kuvatud teisendussammude hulgast.
3) Osavalem esineb grammatika põhjal saadud süntaksipuus. Erandina käsitletakse assotsiatiivseid tehteid (konjunktsioon, disjunktsioon ja ekvivalents), mille puhul lubatakse valida ka selline osavalem, mida grammatika põhjal saadud süntaksipuus tegelikult ei esine. Sellised juhud tekivad, kui valemis on kõrvuti kaks või enam sama prioriteediga assotsiatiivset tehet. Siis lubatakse kahe ja enama tehte korral

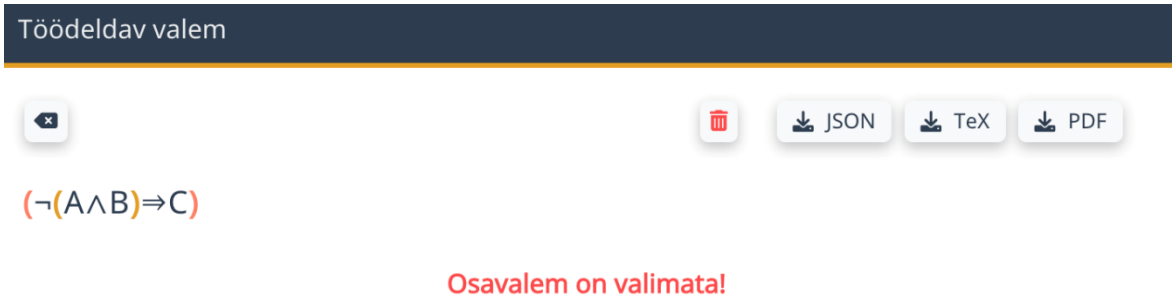
valida ka parempoolseim osavalem ja kolme ja enama tehte puhul ka üks keskmistest osavalemitest. Näiteks valemist *A∧B∧C∧D*, mille süntaksipuu on toodud joonisel 7, on lubatud valida teisendamiseks kõik järgmised osavalemid: *A∧B∧C∧D*, *A∧B∧C*, *B∧C∧D*, *A∧B*, *B∧C*, *C∧D*, *A*, *B*, *C* ja *D*. Nendest ei esine süntaksipuus osavalemid *B∧C∧D*, *B∧C* ja *C∧D*. Selline erand on lubatud selleks, et teisendamine oleks lähedasem käsitsi teisendamisele, mille juures kasutaja saab eraldi sulgude lisamise ja liigutamisega seotud sammud vahele jätta.

4) Valitud teisendus on lubatud. See kontroll kehtib sulgude eemaldamise ja selliste teisenduste rakendamisel, mille puhul valitud osavalem peab olema tõeväärtus. Sulgude eemaldamine on keelatud siis, kui sulgude sees oleva osavalemi tehte prioriteet on madalam kui selle tehte prioriteet, mille osa sulgavaldis on. Näiteks kui valemist *A∧(B∨C)* on valitud osa *(B∨C)* ja proovitakse sulge eemaldada, siis seda ei lubata, sest disjunktsiooni prioriteet on madalam kui konjunktsioonil. Samas valemist *A∧(B∧C)* on sulgude eemaldamine lubatud.
5) Kasutajalt on vaja uue osavalemi sisestamist. Sellisel juhul kuvatakse dialoogiaken, kus palutakse sisestada uus osavalem. Sisestamine on sarnane sellega, kuidas valem sisestatakse enne teisenduste tegema hakkamist. Sel juhul kontrollitakse samuti, kas uus osavalem on sisestatud, korrektse struktuuriga ja ei sisalda predikaate või funktsioone, mille argumentide arv põhivalemis on teistsugune.



Joonis 7. Valemi *A∧B∧C∧D* süntaksipuu. Reeglite nimed vastavad grammatikale (lisa 4).

Juhul kui punktides 1–4 mainitud kontrollidest mõni ei lähe läbi, siis kuvatakse kasutajale töödeldava valemi all sellest informeeriv veateade. Joonisel 8 on toodud näide sellest, kui osavalem on jäänud valimata. Samas kohas kuvatakse vastava sisuga veateated, kui punktides 2–4 mainitud kontrollid ei lähe läbi.



Joonis 8. Veateate kuvamine, kui osavalem on enne teisendusreegli nupule vajutamist jäänud valimata.

Juhul kui punktis 5 nimetatud kontrollid ei lähe läbi, kuvatakse veateade dialoogiaknas, kus uut osavalemit sisestatakse.

Kui kõik teisenduseelsed kontrollid on läbitud, liigutakse edasi teisenduse tegemise juurde. Teisendusi tehakse tööriista ANTLR v4 abil loodud *visitor* isenditega. Iga teisenduse jaoks on mõlemas suunas[21] defineeritud *visitor* klass, mille esindaja luuakse vastavalt teisendusreegli koodile ja teisenduse suunale. Lisas 5 on toodud joonisel 6 näidatud teisendusreeglile L10 vastava *visitor* klassi koodinäide. Sarnaselt sellega on koostatud kõik *visitor* klassid. Saadud isendiga kontrollitakse valitud osavalemist saadud süntaksipuu põhjal vastavust rakendatud reeglis esinevale valemikujule, sobivusel teostatakse teisendus ja tagastatakse tulemus sõnena.

Enne tulemuse tagastamist lisatakse valitud osavalemis esinevatele osavalemitele vajadusel sulud. Sulgude lisamisel arvestatakse tehete prioriteeti ja need lisatakse vaid sel juhul, kui madalama prioriteediga tehe tuleb teha enne kõrgema prioriteediga tehet. Näiteks osavalemis $\neg(A \wedge B \vee \neg C)$ eituse sulgude sisse viimisel saadakse tulemuseks $\neg(A \wedge B) \wedge \neg\neg C$. Kasutajatestimise ajal ei toimunud sulgude lisamine korrektselt sel juhul, kui enam kui kahe sama prioriteediga tehte kõrvuti sattumisel valiti teisendamiseks osavalem keskelt. See viga parandati täiendava sulgude lisamise kontrolliga nimetatud juhtudele.

Kui osavalemi kuju ei sobi, siis tagastatakse väärtus *null*. Sel juhul kuvatakse kasutajale teade mittesobivast teisendusest. Kui teisendus vastab valitud osavalemi kujule, tehakse teisendus ning uuendatakse töödeldava valemi väärtus (joonis 9).

[21] Erandina on mõlemas suunas teisenduse tegemiseks kasutatud ühte *visitor* klassi reeglite L3 ja P11–P14 puhul.

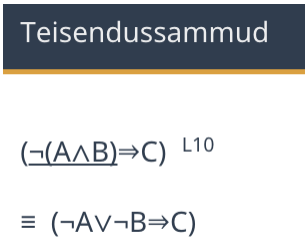
Joonis 9. Töödeldav valem pärast joonisel 3 näidatud teisendusreegli L10 rakendamist.

Tehtud teisendussammud kuvatakse töödeldava valemi all. Töödeldava valemi ja teisendussammude kuvamise lahtrid hoitakse teineteisest lahus sellepärast, et osavalemi valimine toimuks alati sama koha pealt.

Enne kasutajatestimist ei töötanud korrektselt seotud muutujate ümbernimetamisega seotud reeglid (pärast peatükis 7.1 nimetatud reeglite kokkuviimist tähistega P11 ja P12, joonis 4). Vea parandamiseks lisati uue vaba muutuja valimisel keelatud tähiste valikusse kogu valemi muutujad, mitte ainult valitud osavalemi muutujad, nagu oli tehtud enne kasutajatestimist.

7.4 Teisendussammude kuvamine

Teisendussammude kuvatakse üksteise all. Joonisel 10 on näidatud teisendussammude kuvamine juba alustatud näite varal.



Joonis 10. Teisendussammude kuvamine. Esimesel real on joonega alla tõmmatud valitud osavalem ja väiksema ülaindeksiga on välja toodud valitud teisendusreegli kood. Alumisel real on näha teisenduse tulemus.

Iga tehtud teisenduse korral kajastatakse valitud osavalem, millele on joon alla tõmmatud, ja rakendatud reegel, mis on lisatud valemi järele väiksemas kirjas. Kõige alumise valemi puhul joon ja reegli kood puuduvad, sest sellele ei ole veel teisendust rakendatud.

7.5 Teisendussammude kustutamine

Rakenduses on võimaldatud teisendussamme kustutada ühekaupa või kõik korraga. Selleks on töödeldava valemi kohale lisatud kaks nuppu, millest vasakpoolne on ühe teisendussammu kustutamiseks, parempoolne kõikide teisendussammude kustutamiseks. Nupud on paigutatud suure vahega, et kasutaja ei vajutaks kogemata vale nupu peale. Joonisel 11 on näha punase joonega ümbritsetud kustutamisenuppude asukoht töödeldava valemi suhtes.

Joonis 11. Kustutamisnupud (tähistatud punase joonega). Vasakpoolne nupp on ühe, parempoolne kõikide teisendussammude kustutamiseks.

Pärast ühe teisendussammu kustutamist muudetakse ka kuvatud teisendussamme, millest viimaseks jääb enne kustutamist eelviimasena kuvatu. Selle puhul on eemaldatud allajoonitud osa ning rakendatud teisenduse kood. Kui enne kustutamist on sisestatud ainult üks valem, kustutatakse valem kuvatavate teisenduste alt, kuid valemi sisestuslahtrisse jäetakse valem alles. Kõikide teisendussammude kustutamisel küsitakse kasutajalt kõigepealt kinnitust, et tegevusega jätkata. Kui kasutaja luba ei anna, ei muudeta midagi, vastasel juhul kustutatakse kõik tehtud teisendussammud ning sisendlahtrisse jäetakse esimene valem alles. Enne kasutajatestimist kustutati ka sisendlahtris olev valem, kuid tagasiside põhjal jäeti valem sisendlahtrisse alles. Sel juhul on väikse vea parandamine hõlpsam, sest ei pea tervet valemit uuesti sisestama.

7.6 Vahetulemuse salvestamine ja üleslaadimine

Juhuks, kui kasutajal peaks teisendamine jääma pooleli ning on vajadus sellega hiljem jätkata, on rakendusse lisatud võimalus vahetulemus alla laadida .json vormingus. Selle võimaluse kasutamiseks peab kasutaja olema sisestanud vähemalt ühe valemi ning vajutanud nupule „Alusta teisendamist“. Seejärel on kasutajal võimalik vajutada nupule „JSON“, mis on näha joonisel 12. Enne faili allalaadimist avatakse dialoogiaken, kus palutakse sisestada soovitud failinimi.

Joonis 12. Vahetulemuse allalaadimiseks tuleb vajutada nupule „JSON“.

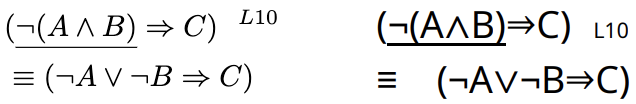
Allalaaditavasse faili salvestatakse teisendussammud massiivi (ingl *array*). Iga teisendussammu puhul lisatakse kaasa info valitud osavalemi ja rakendatud reegli kohta. Lisas 6 on .json vormingus faili sisu, mis on saadud siin peatükis jälgitava näite põhjal.

Saadud faili saab rakendusse uuesti importida esialgsest seisust, mille korral pole veel vajutatud nupule „Alusta teisendamist“. Importimise alustamiseks tuleb vajutada nupule „... laadi üles pooleliolev lahendus JSON-failist“. Seejärel saab kasutaja oma arvutist valida vastava faili. Valiku saab langetada ainult .json vormingus failide hulgast. Rakendus ei luba üles laadida suvalise sisuga faile. Faili sisu sisselugemisel kontrollitakse, et iga massiivi elemendis sisalduva valemi kuju vastaks rakenduses kasutatud grammatikale ja et valitud osavalemit tähistavad indeksid oleksid valemi pikkuse piires. Kui mitte, siis katkestatakse faili sisselugemine ja kuvatakse kasutajale teade vigase sisuga failist. Kui faili sisu on korrektne, näidatakse kasutajale teisendussamme samas seisus, nagu need olid enne töö katkestamist. Pärast faili sisu üleslaadimist saab kasutaja teisendustega jätkata.

7.7 Lõpptulemuse salvestamine

Lõpptulemuse salvestamiseks on rakenduses võimaldatud tulemus alla laadida nii .tex kui ka .pdf vormingus. Joonisel 12 on näha vastavalt nupud „TeX“ ja „PDF“, mille vajutamisel küsitakse kasutajalt failinime ja seejärel laaditakse alla vastav fail. Mõlemal juhul on faili salvestatud tehtud teisendussammud. Joonisel 13 on toodud vasakul .tex vormingus fail pärast faili sisu kompileerimist keskkonnas Overleaf[22] ja paremal .pdf vormingus faili sisu.

[22] https://www.overleaf.com/ (23.04.2022)



Joonis 13. Vasakul .tex faili sisu kompileerituna keskkonnas Overleaf ja paremal .pdf faili sisu.

Mõlema vormingu puhul on sisu sarnane: iga sammu juures on allajoonitult valitud osavalem ja väiksema kirjaga rakendatud teisendusreegli kood. Erinev on teisendusreegli koodi asukoht, mis .tex vormingus failis on valemi järel ülaindeksina ja .pdf vormingus failis väiksemas kirjas valemi järel. Erinevuse põhjus tuleneb npm paketist pdfmake, mis ei toeta ülaja alaindeksite kasutamist. Lisas 7 on toodud joonisel 13 näidatud .tex vormingus faili sisu.

7.8 Rakenduse eelised ja edasiarendusvõimalused

Selle töö raames valminud rakenduse kasutamisel on eesmärk teisendada lause- ja predikaatarvutuse valemeid võimalikult sarnaselt käsitsi teisendamisega. Kolmandas peatükis analüüsiti kuut rakendust, millega on võimalik loogikavalemeid mingil moel teisendada. Selles alapeatükis tuuakse välja loodud rakenduse eelised võrreldes varem analüüsitud tööriistadega.

Valminud rakenduses on võrreldes kolmandas peatükis analüüsitud tööriistadega valemite sisestamine mugavam seetõttu, et loogikasümboleid saab sisestada nii TeXi kujul kui ka nuppude ja kiirklahvide abil. Kuna valemite sisestamisel teisendatakse loogikatähised kohe ümber, on kuvatud valem alati ühtlase välimusega, mis parandab sisestatud valemi loetavust.

Enamiku analüüsitud rakenduste puhul ei olnud võimalik teisendada osavalemeid, mis on selle rakenduse puhul võimaldatud. Lausearvutuse ja predikaatloogika valemite teisendamise õpiprogrammis oli see võimalik, kuid osavalemi valimine oli valemi kuvamisel kasutatud ainult musta värvi tõttu visuaalselt keeruline. Loodud rakenduses on teisenduste tegemise ajal erinevate värvidega välja toodud sulgavaldised, mille alustav ja lõpetav sulg on sama värvi, mis aitab sulgudega piirnevaid osavalemeid kiiremini tuvastada. Lisaks kuvatakse tehtud teisendused kohe koos valitud osavalemi ja kasutatud reegli koodiga. See informatsioon oli mainitud õpiprogrammis salvestatud eraldi faili. Kuigi rakendus eMathHelp ei võimaldanud teha teisendussamme käsitsi, kuvati seal siiski automaatselt tehtud

teisenduste informatsioon. Selle juures oli aga valitud osavalem mõnel puhul ebakorrektselt märgitud.

Enamiku analüüsitud tööriistade puhul oli piiratud võimalike lõppkujude valik. Loodud rakenduses on olemas kõik vajalikud teisendusreeglid, et viia predikaatarvutuse valemid prefikskujule ja lausearvutuse valemid kujudele TDNK, DNK, TKNK ja KNK. Nimetatud kujud on saavutatavad nii terve sisestatud valemi kui ka väiksema osavalemi korral, sest teisendamiseks on lubatud valida tervest valemist väiksem osavalem.

Loodud rakenduses on võimaldatud vahetulemuste allalaadimine ja saadud faili importimine, et hiljem teisendustega jätkata. Lisaks saab tulemuse viia TeXi kujule ja alla laadida ka .pdf vormingus.

Vaatamata nimetatud eelistele, on rakendusel siiski ka puudujääke, mida edasise arenduse käigus võib eemaldada. Kuna loodud rakendus on arhitektuurilt ainult eessüsteemirakendus ilma andmebaasi ja tagasüsteemita, siis võib kasutajamugavust pärssida see, et vahetulemusi ja lõpptulemust peab alla laadima ega saa rakendusesiseselt salvestada. See muudab ka tehtud teisenduste jagamise kaasüliõpilaste ja õppejõududega ebamugavaks. Selle puudujäägi parandamiseks oleks võimalik rakendusele lisada andmebaas, kus hoitakse kasutajate tehtud teisendusi. Sama andmebaasi kasutades saaks võimaldada ka failide mugavama jagamise teiste kasutajatega rakendusesiseselt.

Hetkel on rakendus ainult eestikeelne. Selleks, et rakendust saaksid mugavalt kasutada ka kasutajad, kes eesti keelt ei valda, võib rakenduse tõlkida näiteks inglise keelde. Selle lihtsamaks võimaldamiseks on rakendamise arendamisel kasutatud tõlgete hoidmise ja kuvamise tehnoloogiat vu3-i18n.

Rakenduses on kasutatud andmete hoidmiseks ja komponentide vahel edastamiseks teeki Vuex. Alates sellest, kui see rakenduse tehnoloogiate hulka valiti, on aga ametlik toetatud tehnoloogia muutunud. Nüüd on Vuex asemel soovitatud Pinia[23], mille lisamine rakendusse võib pikas perspektiivis aidata muuta edasise arenduse mugavamaks ja jätkusuutlikumaks.

Selles peatükis kirjeldati valminud rakendust. Keskenduti eelkõige funktsionaalsetes nõuetes toodule ja kirjeldati, kuidas rakenduses nõuded täidetud on. Selleks tehti läbi üks teisendus ning näidati, kuidas see rakenduses toimub. Täpsemalt kirjeldati assotsiatiivsuse ja sulgude lisamisega seotut. Peatüki lõpus toodi välja loodud rakenduse eelised võrreldes kol-

[23] https://pinia.vuejs.org/ (23.04.2022)

mandas peatükis analüüsitud tööriistadega ning edasise arenduse võimalikud suunad. Viimases peatükis võetakse selle bakalaureusetöö raames tehtu lühidalt kokku.

8. Kokkuvõte

Tartu Ülikooli arvutiteaduse instituudis koostatakse uut kursust, milles on olulisel kohal lause- ja predikaatarvutuse valemite teisendamine. Kursusel on vaja, et valemite teisendamine toimuks analoogiliselt käsitsi teisendamisele, mis tagab selle, et kasutaja saab ise teisenduse eesmärgi määrata ja vajaduse järgi ka valemist teisendamiseks välja eraldada väiksemaid osavalemeid.

Töö alguses analüüsiti kuut rakendust, millega on võimalik loogikavalemeid teisendada ja toodi välja nende peamised negatiivsed küljed ja ebamugavad aspektid kasutuse juures. Põhilised puudused olid seotud valemi sisestamisega ja teisendussammude tegemise ning võimalike tulemuste piiratusega. Enamiku rakenduste puhul tehti kogu teisenduskäik kasutaja eest lõpuni ära. Sellest tulenevalt oli selle töö eesmärk luua uus tööriist lause- ja predikaatarvutuse valemite teisendamiseks võimalikult sarnaselt käsitsi teisendamisega.

Töö tulemusena valmis veebirakendus, mis võimaldab kasutajal teisendada loogikavalemeid sarnaselt käsitsi teisendamisega. Kasutaja saab ette anda loogikavalemi ning valida sellest vastavalt oma soovile osavalemi ning rakendada sellele sobivat teisendusreeglit. Tehtud teisendussammude kuvamisel tuuakse eraldi välja valitud osavalem ja rakendatud reegel. Vigaselt valitud osavalemite või teisendusreeglite korral informeeritakse kasutajat vastavate veateadetega. Lisaks on kasutajal võimalik vahe- ja lõpptulemusi alla laadida, et vastavalt hiljem tööd jätkata või tulemusi kaastudengite ja õppejõududega jagada.

Enne rakenduse lõplikku valmimist viidi läbi kasutajatestimine. Selgus, et kasutajad olid uue rakendusega paljuski rahul, kuid tegid siiski väärtuslikke parandusettepanekuid. Nende põhjal tehti parandusi nii elementide paigutuses, veateadete täpsuses kui ka sulgude automaatsel lisamisel ning lisati kaks puuduolevat reeglit. Veel muudeti sulgude vahel olevate osavalemite tuvastamist lihtsamaks alustavale ja lõpetava sulule värvi lisamisega.

Loodud veebirakenduse võimalike edasiarendustena toodi välja andmebaasi lisamine, rakenduse tõlkimine teistesse keeltesse ja teegi Vuex väljavahetamine uue tehnoloogia Pinia vastu.

9. Viidatud kirjandus

[1] Palm, R., Prank, R. Sissejuhatus matemaatilisse loogikasse. Tartu: Tartu Ülikooli Kirjastus, 2004.

[2] Tammet, T. Logictools. https://logictools.org/about.html (01.04.2022).

[3] Vaiksaar, V. Lausearvutuse ja predikaatloogika valemite teisendamise õpiprogramm. Tartu Ülikool, arvutiteaduse instituut, bakalaureusetöö, 2003.

[4] Stroom, S. Lause- ja predikaatarvutuse valemite teisendamise õpiprogrammi täiendamine. Tartu Ülikool, arvutiteaduse instituut, bakalaureusetöö, 2013.

[5] MDN Web Docs Glossary. https://developer.mozilla.org/en-US/docs/Glossary (22.04.2022).

[6] Vailshery, LS. Most used web frameworks among developers worldwide, as of 2021. https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (22.04.2022).

[7] Patel, J. 5 Best Single Page Application Frameworks To Consider For Web Apps. 2021 https://www.monocubed.com/blog/top-single-page-application-frameworks/ (22.04.2022).

[8] Wohlgethan, E. Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js. Hamburg University of Applied Sciences, Department of Computer Science, Bachelor Thesis, 2018.

[9] Breedis, R. JavaScripti kasutajaliidese raamistike võrdlus. Tartu Ülikool, arvutiteaduse instituut, bakalaureusetöö, 2021.

[10] Vue.js Documentation. https://vuejs.org/guide/introduction.html (25.04.2022).

[11] Vuex Documentation. https://vuex.vuejs.org/ (25.04.2022).

[12] Pdfmake Documentation. https://pdfmake.github.io/docs/0.1/ (25.04.2022).

[13] Parr, T. The Definitive ANTLR 4 Reference. United States of America: Pragmatic Bookshelf, 2013.

[14] Saar, S. Teek predikaatarvutuse väljendamisülesannete lahenduste kontrollimiseks. Tartu Ülikool, arvutiteaduse instituut, bakalaureusetöö, 2017.

[15] Jaggo, J. JavaScripti staatilise koodianalüsaatori loomine teenusena. Tartu Ülikool, arvutiteaduse instituut, bakalaureusetöö, 2013.

[16] Kapałka, K. FOL rewritten for Antlr4. https://github.com/antlr/grammars-v4/blob/master/fol/fol.g4 (07.12.2021).

[17] ANTLR v4 JavaScript target. https://github.com/antlr/antlr4/blob/master/doc/javascript-target.md (25.04.2022).

Lisad

Lisa 1. Kaasapandud failid

Failis *lisad.zip* on järgmine sisu:

1) fail *Loogikavalemite teisendaja kasutajatestimine – Google Forms.pdf* (kasutajatestimise küsimustik);
2) fail *Loogikavalemite teisendaja kasutajatestimine (Responses).xlsx* (kasutajatestimise küsimustiku vastused);
3) kaust *teisendaja* (rakenduse lähtekood).

Rakenduse käivitamiseks lokaalselt peab paigaldatud olema Node.js[24] (rakenduse arendamisel kasutati versiooni 16.15.0). Rakenduse saab käivitada lokaalselt liikudes käsureal kausta *teisendaja* ja jooksutades järgmised käsud:

1) npm install;
2) npm run serve.

[24] https://nodejs.org/en/ (09.05.2022)

Lisa 2. Rakenduses olemasolevad põhisamaväärsused

Lausearvutuse põhisamaväärsused

$F \wedge F \equiv F$

$F \vee F \equiv F$

$F \wedge G \equiv G \wedge F$

$F \vee G \equiv G \vee F$

$F \Leftrightarrow G \equiv G \Leftrightarrow F$

$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$

$(F \vee G) \vee H \equiv F \vee (G \vee H)$

$(F \Leftrightarrow G) \Leftrightarrow H \equiv F \Leftrightarrow (G \Leftrightarrow H)$

$F \wedge (F \vee G) \equiv F$

$F \vee F \wedge G \equiv F$

$F \wedge (G \vee H) \equiv F \wedge G \vee F \wedge H$

$F \vee G \wedge H \equiv (F \vee G) \wedge (F \vee H)$

$\neg 1 \equiv 0$

$\neg 0 \equiv 1$

$\neg (F \wedge G) \equiv \neg F \vee \neg G$

$\neg (F \vee G) \equiv \neg F \wedge \neg G$

$F \Rightarrow G \equiv \neg (F \wedge \neg G)$

$F \Rightarrow G \equiv \neg F \vee G$

$F \wedge G \equiv \neg (F \Rightarrow \neg G)$

$F \vee G \equiv \neg F \Rightarrow G$

$F \Leftrightarrow G \equiv F \wedge G \vee \neg F \wedge \neg G$

$F \Leftrightarrow G \equiv (F \Rightarrow G) \wedge (G \Rightarrow F)$

$(F) \equiv F$

$F \wedge (G \vee \neg G) \equiv F$

$F \vee G \wedge \neg G \equiv F$

$F \vee \neg F \equiv 1$

$F \wedge \neg F \equiv 0$

$F \wedge 1 \equiv F$

$F \vee 1 \equiv 1$

$F \wedge 0 \equiv 0$

$F \vee 0 \equiv F$

$\neg \neg F \equiv F$

Predikaatarvutuse põhisamaväärsused

$\neg \forall x F(x) \equiv \exists x \neg F(x)$

$\neg \exists x F(x) \equiv \forall x \neg F(x)$

$\forall x (F(x) \wedge G(x)) \equiv \forall x F(x) \wedge \forall x G(x)$

$\exists x (F(x) \vee G(x)) \equiv \exists x F(x) \vee \exists x G(x)$

$\forall x (F(x) \wedge G) \equiv \forall x F(x) \wedge G$

$\forall x (F(x) \vee G) \equiv \forall x F(x) \vee G$

$\exists x (F(x) \wedge G) \equiv \exists x F(x) \wedge G$

$\exists x (F(x) \vee G) \equiv \exists x F(x) \vee G$

$\forall x (F(x) \Rightarrow G) \equiv \exists x F(x) \Rightarrow G$

$\exists x (F(x) \Rightarrow G) \equiv \forall x F(x) \Rightarrow G$

$\forall x (F \Rightarrow G(x)) \equiv F \Rightarrow \forall x G(x)$

$\exists x (F \Rightarrow G(x)) \equiv F \Rightarrow \exists x G(x)$

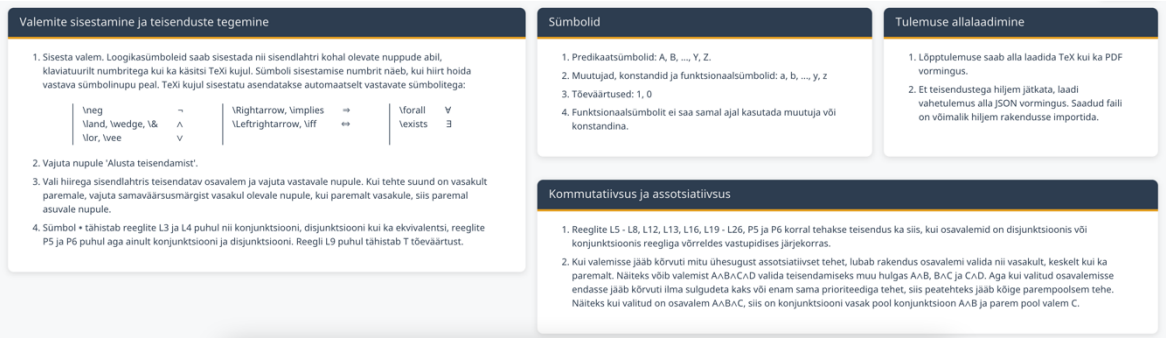
$\forall x F(x) \equiv \forall y F(y)$

$\exists x F(x) \equiv \exists y F(y)$

$\forall x \forall y F(x,y) \equiv \forall y \forall x F(x,y)$

$\exists x \exists y F(x,y) \equiv \exists y \exists x F(x,y)$

Lisa 3. Kasutusjuhend



Joonis 14. Rakenduse kasutusjuhend.

Lisa 4. ANTLR v4 vasakassotsiatiivne grammatika

```
grammar PredGrammar;

start: formula EOF;

formula
    : predicate                              #pred
    | FORALL SYMBOL formula                  #forall
    | EXISTS SYMBOL formula                  #exists
    | NEG formula                            #neg
    | left=formula op=AND right=formula      #and
    | left=formula op=OR right=formula       #or
    | left=formula op=IMPL right=formula     #impl
    | left=formula op=EQ right=formula       #eq
    | LPAREN formula RPAREN                  #paren
    | T                                      #true
    | F                                      #false
    ;

predicate: PRED (LPAREN term (SEP term)* RPAREN)?;
term: SYMBOL | funct;
funct: SYMBOL LPAREN term (SEP term)* RPAREN;
SYMBOL: [a-z];
PRED: [A-Z];
SEP: ',';
LPAREN: '(';
RPAREN: ')';
NEG: '¬';
AND: '∧';
OR: '∨';
IMPL: '⇒';
EQ: '⇔';
FORALL: '∀';
EXISTS: '∃';
T: '1';
F: '0';
WS: [ \t\r\n] -> skip;
```

Lisa 5. Näide *visitor* klassist

```
/* eslint-disable */
// jshint ignore: start
import antlr4 from 'antlr4';
import { addParensNeg } from '@/js/Parentheses';

export default class L10_1Visitor extends antlr4.tree.ParseTreeVisitor {

    visitStart(ctx) {
        try {
            return this.visitNeg(ctx.formula());
        } catch (err) {
            console.log(err);
            return null;
        }
    }

    visitNeg(ctx) {
        if (ctx.constructor.name === "NegContext") {
            if (ctx.formula().constructor.name === "ParenContext") {
                const paren = ctx.formula();
                if (paren.formula().constructor.name === "AndContext") {
                    const and = paren.formula();
                    let left = and.left.getText();
                    const right = and.right.getText();
                    left = addParensNeg(and.left.constructor.name, left);
                    return "¬" + left + "V¬" + right;
                }
            }
        }
        throw "Incompatible input";
    }
}
```

Lisa 6. Näide vahetulemusena salvestatud .json vormingus faili sisust


```
[
    {
        "formula": "(¬(A∧B)⇒C)",
        "selStart": 1,
        "selEnd": 7,
        "ct": "L10"
    },
    {
        "formula": "(¬A∨¬B⇒C)",
        "selStart": 9,
        "selEnd": 0,
        "ct": ""
    }
]
```

Lisa 7. Näide lõpptulemusena salvestatud .tex vormingus faili sisust

```
\documentclass{article}
\usepackage{amsmath}
\begin{document}
\begin{align*}
&(\underline{\neg (A\land B)}\Rightarrow C)^{\;\;\;L10}\\
&\equiv (\neg A\lor \neg B\Rightarrow C)\\
\end{align*}
\end{document}
```

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Norman Pirk,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „Loogikavalemite teisendusredaktor“,

   mille juhendaja on Reimo Palm,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Norman Pirk

09.05.2022