TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKA TEADUSKOND
Arvutiteaduse instituut
Infotehnoloogia eriala

Timmu Ründal

Hulgateoreetiliste avaldiste teisendamisega seotud ülesannete tüübid diskreetse matemaatika veebikeskkonna jaoks

Bakalaureusetöö (6 EAP)

Juhendajad: dots. R. Prank
M. Niitsoo

Autor: ..... ″.....″ mai 2012
Juhendaja: ..... ″.....″ mai 2012
Juhendaja: ..... ″.....″ mai 2012

Lubada kaitsmisele
Professor .... ″.....″ mai 2012

Tartu 2012

Sisukord

Sissejuhatus

Tartu Ülikooli aines „Diskreetse matemaatika elemendid“ kasutatakse mitmeid õpiprogramme, mille eesmärk on tudengite ja õppejõudude elu lihtsamaks teha. Rakendused kontrollivad vigu ja annavad kohest tagasisidet. Seega võimaldavad need hoida õppejõudude aega kokku, sest vastasel juhul tuleb kõikide tudengite lahendused käsitsi kontrollida. Ka tudengile on parem, kui ta saab oma veast aimu kohe pärast selle tegemist, mitte mõne päeva või nädala möödumise järel. Kasutusel on mitmeid lausearvutuse programme, aga mitte ühtegi hulgateooria ülesannete jaoks.

Käesoleva lõputöö põhiülesandeks on realiseerida olemasolevale veebis töötavale süsteemile hulgateoreetiliste avaldiste teisendamisel põhinevad tõestusülesanded: seoste $A = B$ ja $A \subseteq B$ tõestamine, kus A ja B on hulgateooria avaldised. Teisendamise käigus saab kasutada nii hulgateooria kui lausearvutuse samaväärsusi.

Antud töö koosneb kolmest peatükist. Esimeses neist tutvustatakse õpiprogrammide eesmärke ja räägitakse üldiselt hulgateooriast. Teises peatükis kirjeldatakse programmile lisatud ülesannetetüüpidest ja kirjeldatakse rakendust kasutaja poole pealt. Kolmandas peatükis keskendutakse programmi implementatsioonile. Lisana on kaasas CD plaat rakenduse lähtekoodiga.

1. Probleemi püstitus ja taust

1.1. Aine ”Diskreetse matemaatika elemendid”

Diskreetse matemaatika elemendid on Tartu Ülikooli aine, mida tavaliselt võtavad esimese aasta informaatika, infotehnoloogia, matemaatika ja arvutitehnika bakalaureuse tudengid. Õppeaine eesmärgiks on omandada põhiteadmised informaatikas ettetulevatest peamistest teoreetilistest mõistetest ning arendada nende kasutamisoskust.

Aine võib jagada kaheks suuremaks osaks. Esimeses osas õpitakse lausearvutuse põhitehteid, lausearvutuse valemite tõeväärtuste leidmist, lausearvutuse samaväärsusi ja valemite teisendamist. Teine osa keskendub hulkadele: hulga ja funktsiooni mõisted ning põhiomadused, tehted hulkade ja funktsioonidega, nende matemaatiline esitus [1].

1.1.1. Probleemid

Paljud tudengid ei tegele kursusega järjepidevalt ja seetõttu on neil probleeme eksamiteks kogu materjali omandamisega. Probleemi leevendamiseks antakse kursuse jooksul kohustuslikke koduseid töid, mis sunnivad üliõpilasi ainega tegelema. Suur osa antavatest ülesannetest põhinevad lausearvutuse ja hulgateooria valemite teisendamisel ning sellega on tudengitel tihti probleeme. Koduseid ülesandeid saab lasta vormistada käsitsi paberil või arvutis tekstifailina. Kuigi viimane variant on mugavam failide esitamiseks ja tagaside andmiseks, on tudengitel keeruline erinevaid lausearvutuse ja hulgateooria sümboleid seal esitada. Samas paberil parandatud töödele ei tule tudengid tihti järgi ja sedasi ei saada tagasisidet oma vigade kohta. Mõlemal juhul peavad õppejõud igale tudengile käsitsi vastama ja see võtab aega.

Valemite teisendamise läbivateks vigadeks on teisendussammude vahelejätmine ja valede sammude kasutamine. Esimesel juhul on küll kõik tudengi kirjutatu õige, aga kuna mõni teisenduse etapp on vahele jäetud, tekib küsimus, kas tudeng ikka sai aru, mida ta kirjutas. Samuti kirjutatakse mõnikord valesid tõestusi ja jõutakse sedasi näiliselt õige vastuseni.

1.1.2. Lahendused

Antud probleemide lahendamiseks on matemaatilise loogika kursuste jaoks loodud mitmeid programme. Kõigi rakenduste põhimõte on analoogne: tudengile esitatakse ülesanne, mille ta peab sammhaaval lahendama. Kui tudeng üritab mittekehtivaid seoseid kasutada, näidatakse kohe veateadet.

Programmide kasutamisel on mitmeid eeliseid. Kohene tagasiside tähendab, et tudeng näeb juba ülesannet lahendades oma vigu ja on suurem tõenäosus, et ta õpib sellest. Kui üliõpilane peab esitama tervikliku lahenduse õppejõule ja saab alles mitme päeva pärast parandatud töö tagasi, võib tal kaduda huvi üldse oma vigu üle vaadata või on tal vahepeal ununenud, mis probleemid tal üleüldse olid. Seega üritatakse õpiprogrammidega pikendada seda osa kursusest, kus tudeng saab tehtule kiiret tagasisidet.

Samuti võimaldab programmi kasutamine harjutada suvalisel hetkel. Tudeng ei pea ootama praktikumi. Rakendused pakuvad ülesannete genereerimise võimalust, mis on kasulikud, kui tahetakse harjutada mõne kindla ülesandetüübi lahendamist.

1.1.3. Olemasolevad programmid

Aine ”Diskreetse matemaatika elemendid” praktikumides kasutatakse praegu kahte programmi: Java-programm teisendusülesannete jaoks ja üle veebi töötav programm tõeväärtustabeli ülesannete lahendamiseks[2]. Mõlemad rakendused oskavad diagnoosida otseseid vigu ning annavad nende kohta tagasisidet kohe pärast vea tegemist.

Teisendusprogramm võimaldab lausearvutuse avaldiste teisendamist. Rakendusel on vahetu ja reeglipõhine režiim. Esimese puhul peab tudeng valima sobiva alamavaldise ning kirjutama käsitsi sellega samaväärse avaldise. Reeglipõhise režiimi puhul on üliõpilasele ette antud teatud hulk teisendusreegleid, mida ta saab rakendada pärast alamavaldise valimist. Ülesandetüüpidest on programmil olemas valemi disjunktiivse ja konjunktiivse normaalkuju leidmine, valemi avaldamine teatud tehete kaudu, eituste viimine vahetult lausemuutujate ette ning valemi eitusega prefikskujule viimine. Veatüüpidest oskab rakendus avastada näiteks alamavaldise valimist või vale reegli kasutamist. Programm ei oska diagnoosida ebaotstarbekate sammude

kasutamisest tingitud vigu olukorras, kus tudeng on saanud küll õige vastuse, aga tema teisendus oleks võinud olla tunduvalt lühem.

Veebipõhine tõeväärtustabeli ülesannete programmi põhiosa moodustab lausearvutuse avaldise põhjal tõeväärtustabeli täitmine. Olemasolevad ülesandetüübid on tõeväärtustabeli arvutamine, samaselt tõesuse kontroll, kehtestatavuse kontroll, samaväärsuse kontroll ja järeldumise kontroll. Neist neli viimast ehk kontrollimisega seotud ülesanded nõuavad tõeväärtustabeli täieliku või osalist leidmist ning selle põhjal otsuste tegemist. Lisaks on ülesandetüüp, kus tuleb leida olemasolevale tõeväärtustabelile sobiv lausearvutuse valem. Analoogselt eelmise programmiga oskab programm diagnoosida otseseid vigu, näiteks vead tõeväärtustabeli täitmises või vastuse andmises.

1.2. Loodav programm

Mõlemad eelpool kirjeldatud programmid võimaldavad lahendada vaid lausearvutuse ülesandeid. Praegu ei ole kasutusel programme hulgateooria ülesannete jaoks. Bakalaureusetöö põhiülesandeks on realiseerida olemasolevale veebis töötavale süsteemile hulgateoreetiliste avaldiste teisendamisel põhinevad tõestusülesanded: seoste $A = B$ ja $A \subseteq B$ tõestamine, kus A ja B on hulgateooria avaldised. Teisendamise käigus saab kasutada nii hulgateooria kui lausearvutuse samaväärsusi.

Laiendataval tõeväärtuse leidmise programmil on ülesannete koostamise ja lahendamise keskkond. Esimene on mõeldud õppejõule ning võimaldab uusi ülesandeid lisada. Samuti käib seal tudengite haldamine, tulemuste vaatamine ja õppejõudude haldamine. Lahendamise keskkond on üliõpilasele. Kuna programm töötab üle veebi, ei pea tudengid seda eraldi alla laadima, tuleb lihtsalt oma veebibrauser suunata õigele aadressile. Samuti ei pea üliõpilased ise oma lahendusi õppejõule saatma. Näiteks Java-programmi puhul peab üliõpilane ülesanded alla laadima, need ära lahendama ja õppejõule saatma. Veebipõhise süsteemi puhul on sisse logides alati kõige värskemad ülesanded olemas, lahenduskäigud salvestatakse automaatselt ja õppejõul on kohe näha tudengi tulemused. Sellepärast otsustatigi laiendada tõeväärtustabeli programmi, kuigi sisu poolest on teisendusprogramm antud tööle lähedasem.

1.2.1. Sarnased programmid

Loodav programm on analoogne Amhersti kolledžis loodud Java-rakendusele nimega Proof Designer [3], mis võimaldab erinevaid teisendusülesandeid lahendada. Paraku on antud rakendus DME jaoks liiga keeruline, kasutades reegleid, mida selles kursuses ei õpita. Samuti puuduvad programmil eelpool loetletud eelised, näiteks ülesannete genereerimine, automaatne lahenduskäikude salvestamine, tagasiside.

Loodavale programmile väga sarnane on Java teisendusprogramm lausearvutuse jaoks, sest mõlemad põhinevad avaldiste teisendamisel. Seepärast oleks isegi lihtsam täiendada just nimetatud rakendust, aga eelpool loetletud põhjustel sai otsustatud veebipõhise süsteemi kasuks.

1.2.2. Võrdlus Proof Designeriga

Proof Designer on loodud õpetamaks tudengitele tõestuste kirjutamist. Seega on ta lihtsalt teisendamiseks liiga keerukas. Hulgateooria avaldiste esitamisel lausearvutuse tehete kaudu kasutatakse kvantoreid, mida Diskreetse matemaatika elementide kursuses puudutatakse põgusalt, ning on seega tudengitele liiga keeruline. Tõestamise tulemusena tekib lahenduseks lisaks avaldistele ka selgitavat teksti, mis aitavad tõestusest paremini aru saada. Tõestatav teoreem võib olla ükskõik milline predikaatloogika valemi.[4]

Loodava programmi eesmärk on avaldiste teisendamise harjutamine ja samaväärsuse ning alamhulga seose tõestamine. Seega on ta lihtsam ja vähemate võimalustega. Näiteks ei ole programmis kasutatud kvantoreid.

Proof Designer võimaldab lahendamisel kasutada eeldusi ning lisaks tõestatavale avaldisele teisendada ka neid. Lahendamisel on kasutatav tühja hulga mõiste, tehetest on olemas otsekorrutis ning lausearvutuses saab kasutada implikatsiooni ja ekvivalentsi. Loodaval programmil need puuduvad, kuid see-eest on tal olemas hulkade täiend, mis puutub Proof Designeril.

1.3. Hulgateooria ülesanded

1.3.1. Hulgateooria valemite teisendamine

Toome ära hulgateooria tehted.

- Kahe hulga A ja B ühendiks nimetatakse hulka $A \cup B$, mis koosneb nii hulga A kui ka hulga B elementidest: $A \cup B = \{x: x \in A \text{ või } x \in B\}$.
- Kahe hulga A ja B ühisosaks nimetatakse hulka $A \cap B$, mis koosneb hulkade A ja B ühistest elementidest : $A \cap B = \{x: x \in A \text{ ja } x \in B\}$.
- Olgu fikseeritud teatav universaalhulk X. Hulga A täiendiks nimetatakse kõigi nende hulga X elementide hulka A', mis ei kuulu hulka A: $A' = \{x: x \notin A\}$.
- Kahe hulga A ja B vaheks nimetatakse hulka $A \setminus B$, mis koosneb kõigist niisugustest elementidest, mis kuuluvad hulka A, aga ei kuulu hulka B: $A \setminus B = \{x: x \in A \text{ ja } x \notin B\}$.
- Kahe hulga A ja B sümmeetriliseks vaheks nimetatakse hulka $A \Delta B$, mis koosneb kõigist elementidest, mis kuuluvad kas hulka A või hulka B, aga mitte mõlemasse korraga: $A \Delta B = \{x: (x \in A \text{ ja } x \notin B) \text{ või } (x \notin A \text{ ja } x \in B)\}$.

Olgu X universaalne hulk ja $A, B \subseteq X$. Peale selle olgu $x \subseteq X$. Toome sisse lausete tähistused:

- $\boldsymbol{A}$: $x \in A$
- $\boldsymbol{B}$: $x \in B$

Nüüd saame hulgateooriate tehete seosed lausearvutuse tehetega.

- Väitele $x \in A \cup B$ vastab valem $\boldsymbol{A} \vee \boldsymbol{B}$.
- Väitele $x \in A \cap B$ vastab valem $\boldsymbol{A} \,\&\, \boldsymbol{B}$.
- Väitele $x \in A$' ehk $x \notin A$ vastab valem $\neg\boldsymbol{A}$.
- Väitele $x \in A \setminus B$ vastab valem $\boldsymbol{A} \,\&\, \neg\boldsymbol{B}$.
- Väitele $x \in A \Delta B$ vastab valem $\boldsymbol{A} \,\&\, \neg\boldsymbol{B} \vee \neg\boldsymbol{A} \,\&\, \boldsymbol{B}$.

Antud seoseid kasutades saab hulgateooria avaldised viia üle lausearvutuse kujule. Edasi saab kasutada lausearvutusest tundud põhisamaväärsusi.[5,6]

1.3.2. Avaldiste võrdsuse tõestamine

Avaldiste võrdsuse tõestamise ülesannetes on antud kaks hulgateooria avaldist ning tudengi ülesanne on ühte või mõlemat avaldist teisendades viia need samale kujule. Alati töötav lahendus oleks mõlemad avaldised viia disjunktiivsele või konjunktiivsele normaalkujule, kuid see ei ole iga kord vajalik. Samuti võib piirduda ainult ühe avaldise teisendamisega, viies selle samale kujule teise valemiga.

Olgu näiteks vaja tõestada hulgateooria samaväärsus $A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C)$. Alustame teisendamist võrduse vasakust poolest. Teisendussammud koos reeglitega oleksid järgnevad:

1) $x \in A \setminus (B \cup C)$,
2) $x \in A \,\&\, x \notin B \cup C$ ehk $x \in A \,\&\, \neg\, (x \in B \cup C)$ – hulkade vahe definitsioon,
3) $x \in A \,\&\, \neg(x \in B \vee x \in C)$ – hulkade ühendi definitsioon,
4) $x \in A \,\&\, \neg(x \in B) \,\&\, \neg\, (x \in C)$ – De Morgani seadus.

Teisendame ka paremat poolt:

1) $x \in (A \setminus B) \cap (A \setminus C)$,
2) $(x \in A \setminus B) \,\&\, (x \in A \setminus C)$ – hulkade ühendi definitsioon,
3) $x \in A \,\&\, \neg(x \in B) \,\&\, x \in A \,\&\, \neg(x \in C)$ – hulkade vahe definitsioon,
4) $x \in A \,\&\, \neg(x \in B) \,\&\, \neg(x \in C)$ – sarnaste lauseliikmete koondamine.

Kuna mõlemad pooled on võrdsed, siis samaväärsus kehtib.

1.3.3. Alamhulga seose tõestamine

Olgu vaja tõestada, et $(A \cup B) \setminus B \subseteq A$. Tõestamiseks on vaja näidata, et kui mingi objekt on element hulgast $(A \cup B) \setminus B$, siis on see ka element hulgast A. Seega olgu $x \in (A \cup B) \setminus B$. Hulkade vahe definitsioonist saame, et $x \in A \cup B \,\&\, x \notin B$. Rakendades ühendi definitsiooni, saame $(x \in A \vee x \in B) \,\&\, x \notin B$. Nüüd saab kasutada lausearvutuse distributiivsuse seadust, mille tulemuseks on $x \in A \,\&\, x \notin B \vee x \in B \,\&\, x \notin B$. Kuna disjunktsiooni parem argument on samaselt väär, saab kasutada liikmete elimineerimist: $x \in A \,\&\, x \notin B$. Siit järeldub, et alamhulga seos kehtib.

2. Programmi ülesehitus

2.1. Ülesanded

2.1.1. Lisatavad ülesandetüübid

Programmile lisatakse kaks ülesandetüüpi. Esimene neist on samaväärsuse tõestamine. Lahendajale on antud kaks samaväärset, aga erineval kujul olevat hulgateooria avaldist, mis tuleb teisendusreegleid rakendades viia samale kujule. See tähendab, et vasak ja parem pool peavad olema identsed. Muid nõudmisi ei ole, seega võivad vastuse andmisel olla avaldised esitatud nii hulgateooria kui ka lausearvutuse tehete kaudu.

Teine ülesandetüüp on alamhulga seose tõestamine. Lahendajale on antud kaks hulgateooria avaldist, kusjuures esimene neist on teise alamhulk. Pärast avaldiste teisendamist ja vastuse esitamist palutakse põhjendada, miks alamhulga seos kehtib. Selleks tuleb valida kahe omaduse vahel: $A \cap B \subseteq A$ ja $A \subseteq A \cup B$. Kui avaldised on samal kujul nagu põhjenduseks valitud omadus, loetakse vastus õigeks.

2.1.2. Reeglid

Programmi abil kasutatav teisendusreeglid võib jagada kolme gruppi. Esimene neist on hulgateooria tehete avaldamine lausearvutuse tehete abil. Need reeglid, mida on kokku viis, põhinevad hulgateooria tehete definitsioonidel ning on rakendatavad mõlemat pidi. Seega saab sobival kujul lausearvutuse tehted esitada hulgateooria tehete abil. Antud grupi näiteks võib tuua $x \in A \cup B = x \in A \vee x \in B$.

Teine grupp on lausearvutuse reeglid, mis on kõik teatud lausearvutuse samaväärsused. Neid on programmis kasutusel 19. Esindatud on vaid eitust, disjunktsiooni ja konjunktsiooni sisaldavad reeglid. Mõned neist on rakendatavad mõlemat pidi, näitena võib tuua $\neg(F \ \& \ G) = \neg F \vee \neg G$. Ainult ühte pidi kasutatava reegli näiteks on $F \ \& \ (F \vee G) = F$. Teistest erilisemad on muutuja lisamise reeglid $X = X \ \& \ Y \vee X \ \& \ \neg Y$ ja $X = (X \vee Y) \ \& \ (X \vee \neg Y)$, sest lisaks alamavaldise valimisele peab kasutaja sisestama ka uue lausemuutuja, mida reegli abil avaldisse tuua.

Lõpetuseks on hulgateooria samaväärsuste reeglid. Need põhinevad suuresti lausearvutuse reeglitel, näiteks on $A \cup B = B \cup A$ ja $F \vee G = G \vee F$ analoogsed. Antud tüüpi reegleid on võimalik ülesande lahendamisel keelata, sest hulgateooria tehete omaduste tõestamine võib olla ülesanne omaette. Näiteks võidakse ühe ülesandega nõuda samaväärsuse $A \cup B = B \cup A$ tõestamist lausearvutuse tehete abil, aga mõne keerulisema ülesande puhul on võimalik seda omadust reeglina kasutada, et vältida asjatult pikka teisendust.

2.1.3. Kitsendused

Programmi kasutamisel on teatavad kitsendused. Reeglite kasutamine üsna rangelt piiratud ja nende rakendamiseks peab alamavaldis olema täpselt nõutud kujul. See tähendab, et näiteks reegli $\neg F \ \& \ F \vee G = G$ rakendamisel peab kindlasti eitusega konjunktsiooni liige ees olema. Vastasel juhul tuleb eelnevalt rakendada kommutatiivsuse reeglit. Konjunktsioon ja disjunktsioon on implementeeritud binaarsete tehetena ja seetõttu võib mõnikord lihtsana näiv teisendus nõuda mitu sammu. Näiteks olgu samaväärsuse $A \ \& \ B \ \& \ C = A \ \& \ C \ \& \ B$ tõestamine. Esmapilgul võiks lihtsalt B ja C kohad vahetada, aga tehete järjekord seda ei luba, sulud paiknevad teisiti: $(A \ \& \ B) \ \& \ C = (A \ \& \ C) \ \& \ B$. Seega tuleb võrduse vasakus pooles kasutada assotsiatiivsuse reeglit, saades $A \ \& \ (B \ \& \ C)$. Seejärel saab rakendada kommutatiivsust alamavaldisele $B \ \& \ C$ ja lõpuks jälle assotsiatiivsuse reeglit.

Ühest küljest sunnib antud kitsendust tudengit rohkem mõtlema, sest rõhku peab pöörama tehete järjekorrale. Teisalt võib pikemates tõestustes, kus lausemuutujate järjekord konjunktsioonis või disjunktsioonis on teisejärguline probleem, pidev ümberjärjestamine muutuda tülikaks.

Tehetest ei ole realiseeritud otsekorrutis. Samuti on puudu tühihulga ja universaalhulga mõiste ja vastavad reeglid. Seega ei saa näiteks tõestada omadusi $A \cup A' = X$ ja $A \cap A' = \emptyset$, kus X on mingi universaalhulk.

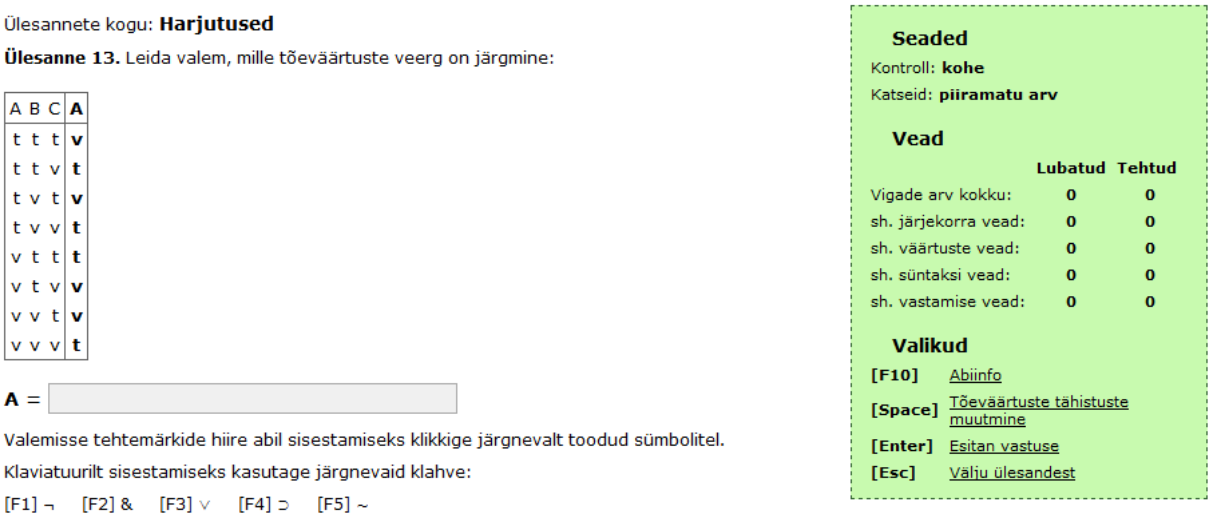
2.2. Olemasolev süsteem

Olemasoleval programmil on ülesannete lahendamise ja koostamise keskkond. Esimene neist on mõeldud tudengitele ja sinna sisse saab üliõpilane logida õppejõult saadud kasutajanime ja

parooliga. Pärast sisenemist kuvatakse ülesannete kogude nimekiri (Joonis 1). Tudeng saab valida ülesannete kogu, pärast mille valimist kuvatakse tabel ülesannete nimekirjaga. Iga ülesande juures kuvatakse ülesande tekst ja natuke statistikat: kas ülesanne on lahendatud, punktid, lahendamise kordade arv, õigete ja valede vastuste arv ning katkestamiste arv. Ülesandele vajutades saab seda lahendada, samuti on võimalik katkestatud lahendust jätkata.

Joonis 1. Ülesannete kogude nimekiri

Ülesande lahendamise vaade on jagatud kaheks osaks (Joonis 2). Suurem osa ekraanist kuulub ülesande tekstile ja lahendamisaknale. Selle kõrval kuvatakse paneeli, kus võib näha hetkel valitud seadeid, lubatud ja tehtud vigu ning erinevaid valikuid seadete muutmiseks või lahendamiseks.

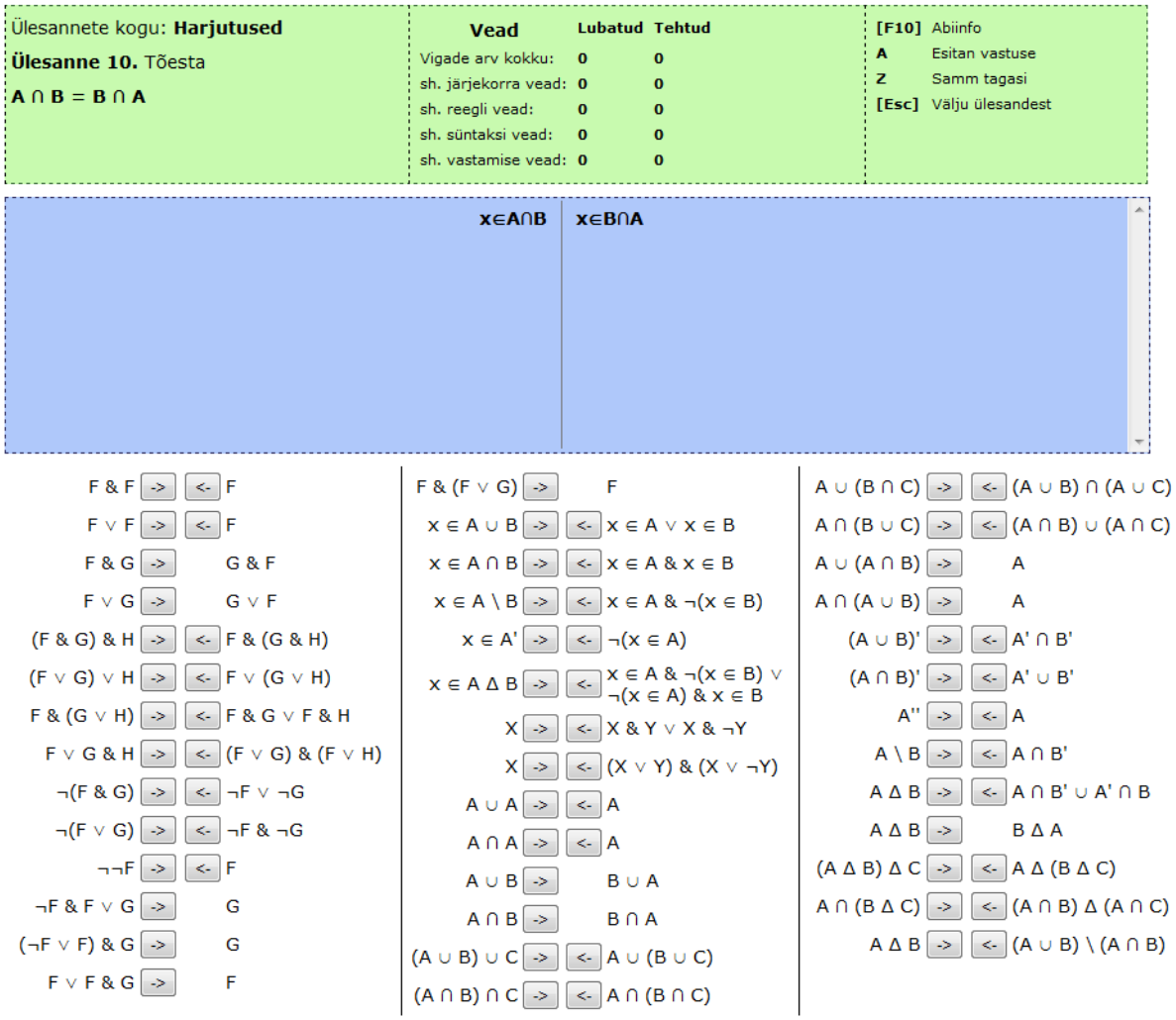


Joonis 2. Ülesande lahendamise vaade

Lisaks ülesannete koostamisele saavad õppejõud hallata tudengeid ja õppejõude, kes süsteemile ligi pääsevad. Samuti saab vaadata iga tudengi tulemusi. Olemas on üldine vaade iga ülesandekogu kohta ning ka detailsemad vaated ühe ülesande erinevate lahenduste ning konkreetse lahenduskäigu kohta.

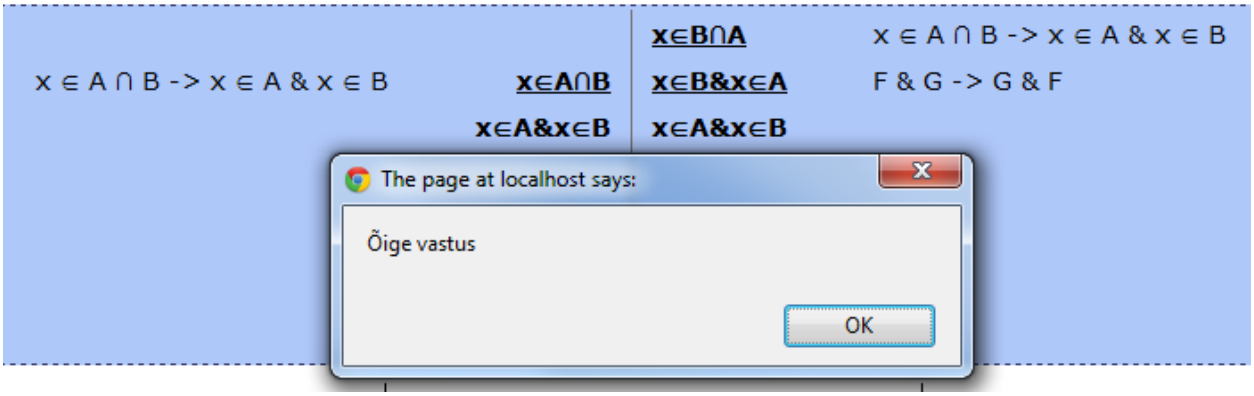
2.3. Tudengi vaade

Uute ülesandetüüpide lisamisel on üritatud säilitada olemasolevat kujundust (Joonis 2). Siiski on tehtud mõningaid muudatusi (Joonis 3). Infokast on viidud paremalt küljelt lehe ülaossa. See jätab kasutajale rohkem ruumi avaldiste teisendamiseks ja teisenduskäigu jälgimine muutub mugavamaks. Lahendamisel saab teisendada esimest avaldist teiseks, teist esimeseks või mõlemad avaldist korraga, nii et lõpptulemusena oleksid mõlemad pooled võrdsed.



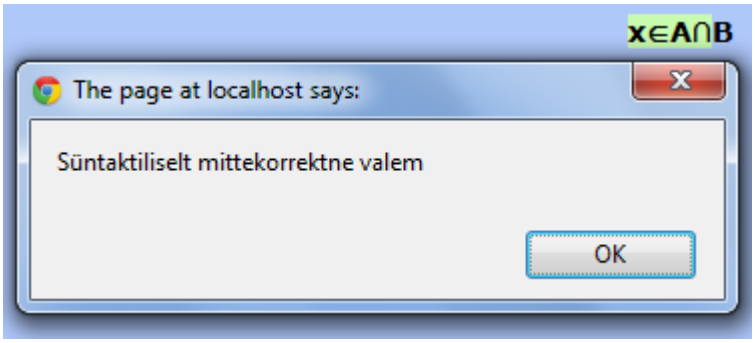
Joonis 3. Ülesande lahendamise leht koos kõikide reeglitega

Ülesannete lahendamisel valib tudeng hiirega avaldise mingi osa, mis seejärel näidatakse teise värviga. Kui alamavaldis on valitud, tuleb hiirega valida rakendatav reegel. Seejärel tehakse teisendus või kuvatakse veateade ning üliõpilane saab jätkata lahendamist. Kui tudeng on teisendanud mõlemad avaldised võrdseks, tuleb esitada vastus ning kui see on õige, loetakse ülesanne lahendatuks (Joonis 4). Tudengil on võimalik tehtud samme tagasi võtta.

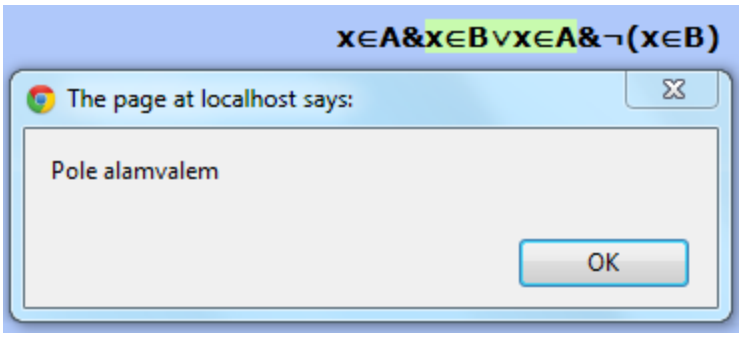


Joonis 4. Õige vastuse teade koos tervikliku lahendusega

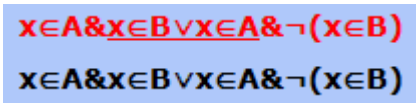
Programm loeb nelja tüüpi vigu. Kaks neist on seotud alamavaldise valimisega. Süntaksi viga tekib, kui valitud osa avaldisest ei moodusta süntaktiliselt korrektset avaldist. Vea tegemisel kuvatakse veateade (Joonis 5). Kui valitud osa on küll süntaktiliselt korrektne, aga ei moodusta antud avaldisest alamavaldist, loetakse see tehete järjekorra veaks ja kuvatakse vastav teade (Joonis 6). Mõlemal juhul kuvatakse lahenduskäigus see rida punase värviga ja suurendatakse antud vea loendurit. Allajoonitud osa avaldises näitab tudengi poolt valitud osa ning see aitab tal oma veast aru saada (Joonis 7).



Joonis 5. Süntaksi viga alamavaldise valimisel

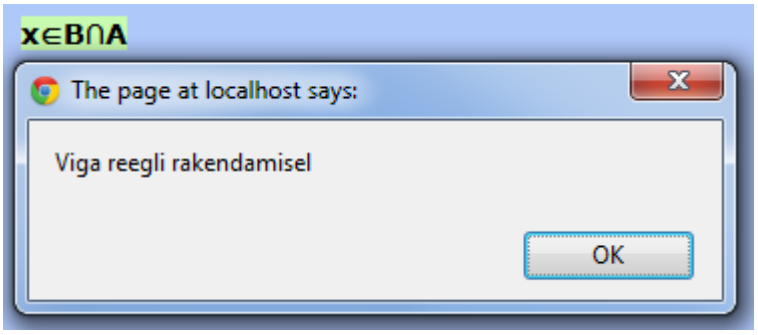


Joonis 6. Järjekorra viga alamavaldise valimisel

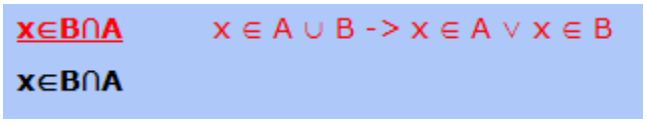


Joonis 7. Valesti valitud alamavaldis

Kolmas veatüüp on seotud reeglite rakendamisega. Kui alamavaldis on õigesti valitud, aga sellele üritatakse rakendada sobimatut reeglit, kuvatakse vastav teade (Joonis 8). Veaga rida märgitakse lahenduses punase värviga ning avaldise kõrval kuvatakse reegel, mida tudeng üritas rakendada (Joonis 9). Samuti suurendatakse reegli vigade loendurit.

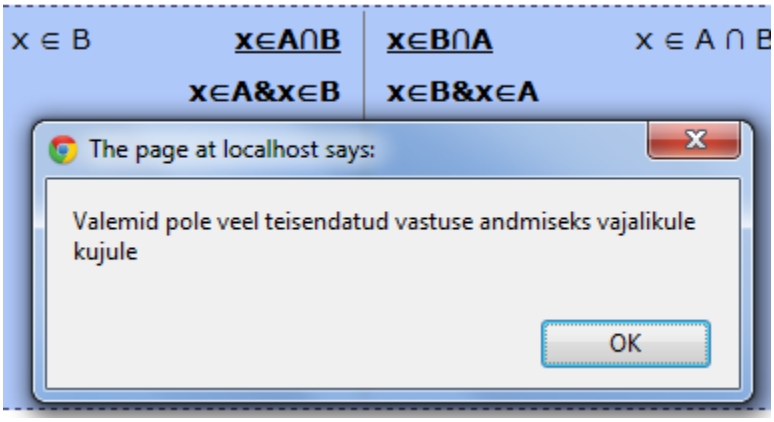


Joonis 8. Reegli rakendamise veateade



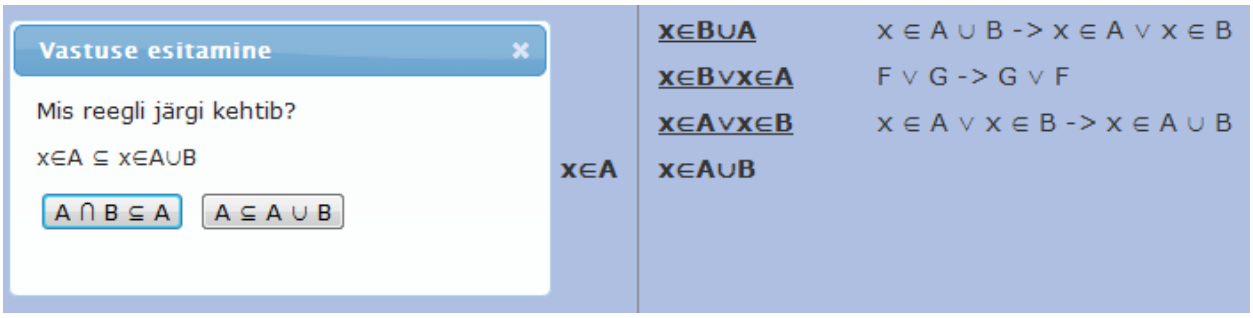
Joonis 9. Valesti kasutatud reegel

Kui tudeng on enda arvates lahendamisega lõpuni jõudnud, peab ta sellest märku andma vastuse esitamisega. Vale vastuse korral kuvatakse vastav teade (Joonis 10) ning suurendatakse vigade loendurit. Õigesti antud vastuse puhul antakse tudengile sellest teade (Joonis 4) ning suunatakse tagasi ülesannete kogu juurde.



Joonis 10. Vale vastuse esitamisel kuvatav veateade

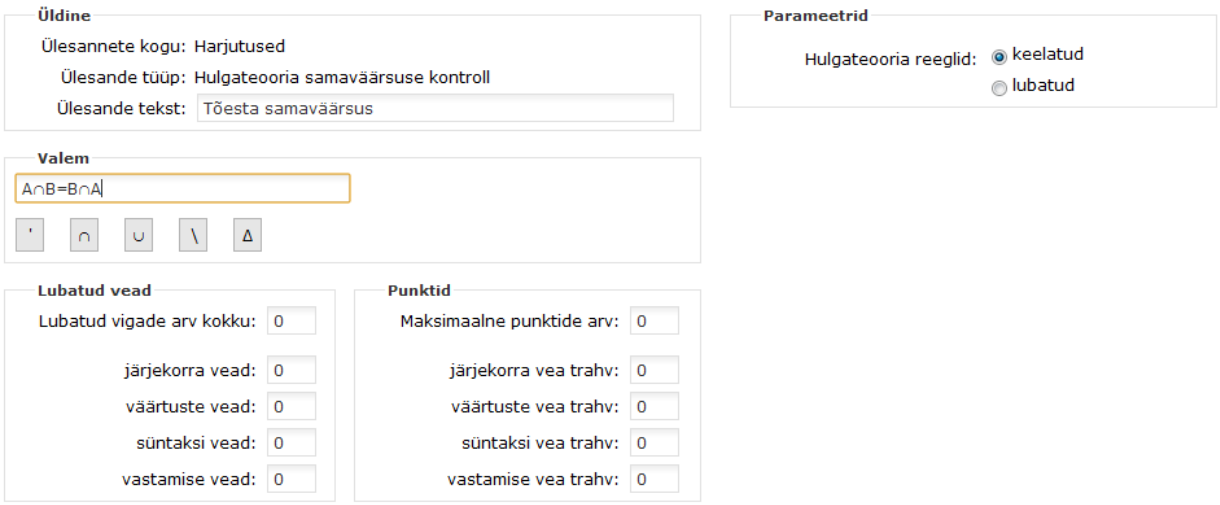
Alamhulga seose tõestamine käib üldjoontes samamoodi. Teisendussammud ja veahaldus on identsed. Erinevus tuleb vastuse esitamisel. Siis kuvatakse kasutajale alamhulga seose põhjendamise dialoog (Joonis 11). Kui tudengi põhjendus on õige, loetakse ülesanne lahendatuks, vastasel juhul antakse vastuse viga märkiv teade.



Joonis 11. Alamavaldise ülesande vastuse esitamise dialoog

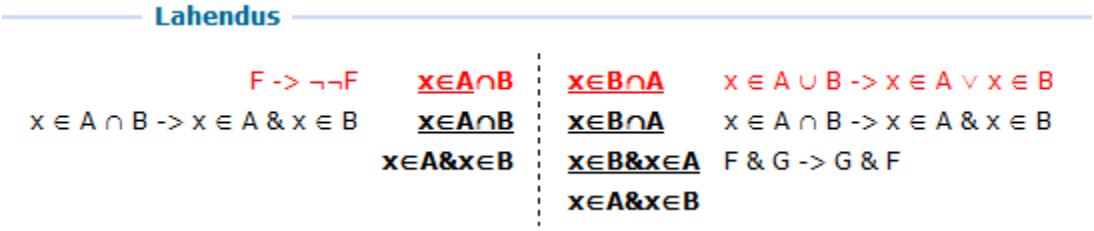
2.4. Õppejõu vaade

Ülesannete lisamise keskkonnale on lisatud kaks uut ülesannete tüüpi: hulgateooria avaldiste samaväärsuse ja alamhulga seose tõestamine. Mõlemal juhul saab õppejõud muuta ülesande teksti, lisada tõestatava avaldise ning muuta mõningaid parameetreid. Neist olulisem on hulgateooria samaväärsuste kasutamise lubamine või keelamine. Joonisel 12 on näha osaliselt täidetud ülesande lisamise vorm. Lubatud vigade arv näitab maksimaalset vigade arvu, mille puhul loetakse lahendus veel õigeks.



Joonis 12. Ülesande lisamine

Õppejõul on võimalik vaadata tudengi tulemusi ja lahenduskäike. Kuvatakse nii tehtud vigade arvud kui ka terve lahenduskäik. Viimasel juhul kuvatakse üksteise kõrval vasaku ja parema poole teisendused. Kui avaldisele on rakendatud reeglit, kuvatakse see avaldise kõrval. Valitud alamavaldis on allajoonitud. Kui alamavaldise või reegli rakendamise puhul on tehtud viga, siis näidatakse vastav rida punase värviga (Joonis 13).



Joonis 13. Tudengi lahenduskäik õppejõu vaates

3. Programmi implementatsioon

3.1. Kasutatavad tehnoloogiad

Olemasolev süsteem, mida täiendama hakatakse, kasutab serveris skriptimiskeelt *PHP*, andmebaasimootoriks on *MySQL*, kliendi poolel *JavaScript*, *HTML*. Programm on loodud *PHP* raamistiku *DomFramework* peale. Raamistik on loodud Inditel Media OÜ poolt veebilehtede loomise protsessi kiirendamiseks[2].

3.1.1. *PHP*

PHP (rekursiivne akronüüm *PHP*: *Hypertext Preprocessor*) on laialt kasutatud vabavaraline üldotstarbeline serveripoolne skriptimiskeel, mis on eriti sobiv veebiarenduseks. Selle lõi Rasmus Lerdorf 1995. aastal. *PHP* koodi süntaks põhineb programmeerimiskeeltel C, Java ja Perl ning seda saab lisada otse *HTML* dokumentidesse. Skriptimiskeele peamiseks eesmärgiks on võimaldada arendajatel kiiresti kirjutada dünaamiliselt genereeritud veebilehti. Samuti saab antud keelt kasutada käsurea skriptide loomiseks ning samuti töölauarakenduste kirjutamiseks, kuigi viimaseks tegevuseks ei ole *PHP* parim valik.[7][8]

PHP töötab enamikel levinumatel operatsioonisüsteemidel, nagu näiteks Linux, mitmed Unixi versioonid, Microsoft Windows ja Mac OS X, ning veebiserveritel, näiteks Apache ja IIS. *PHP* võimaldab kasutada nii protseduraalset kui ka objektorienteeritud programmeerimise paradigmat või segu neist kahest. Skriptimiskeelel on paljude andmebaasimootorite tugi ning suudab suhelda erinevate teenustega kasutades näiteks *LDAP*, *IMAP*, *POP3* ja teisi protokolle.[9]

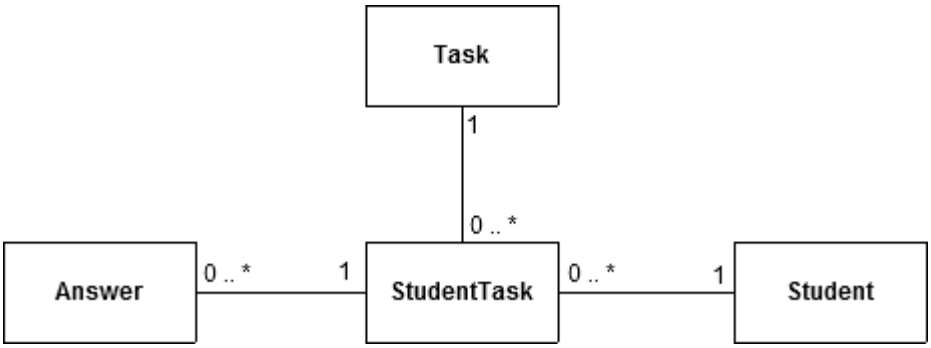
3.1.2. *JavaScript*

JavaScript on Netscape'i poolt arendatud dünaamiline nõrgalt tüübitud skriptimiskeel, mis võib käituda nii protseduraalse kui objekt-orienteeritud keelena.[10] *JavaScripti* standardiks on *ECMAScript*. Veebis on *JavaScripti* tähtsus dokumendi objektmudeli (DOM) manipuleerimisel. Dokumendi objektmudel on liides *XML*, *HTML* ja *XHTML* dokumentidega suhtlemiseks.[11]

jQuery on *JavaScripti* teek, mis lihtsustab *HTML* dokumentide töötlemist, sündmuste haldamist ja *AJAX*-tehnoloogia kasutamist.[12] *AJAX* (akronüüm *Asynchronous JavaScript and XML*) on grupp seotud veebitehnoloogiaid, mis koos kasutades võimaldavad luua asünkroonseid veebirakendusi. Selle abil on võimalik uuendada teatud osa veebilehest ilma tervet lehte uuesti laadimata.[13]

3.2. Olemasolev süsteem

Olemasolevas süsteemis hoitakse andmeid relatsioonilises andmebaasis. Õppejõudu kujutab klass *Lecturer*, tudengit *Student*, ülesannete kogu *Collection* ja ülesannet *Task*. Igale tudengi objektile luuakse automaatselt iga ülesandeobjekti *Task* jaoks klassi *StudentTask* objekt, kus hoitakse tudengi konkreetse ülesande kohta käivat infot. Konkreetse lahenduskäigu jaoks on klass *Answer*. Ülesannete lahendamisega seotud objektide vahelisi seoseid võib näha joonisel 14 toodud klassidiagrammil.



Joonis 14. Klassidiagramm

3.3. Avaldiste esitamine

Avaldiste esitamiseks kasutatakse tehtepuid. Iga tehte jaoks on oma klass. Kõik tehted laiendavad *PHP SetFormula* klassi, millel on väljad vasaku ja parema haru, tehte, prioriteedi ja assotsiatiivsuse suuna jaoks, lisaks staatiline väli operaatorite arvu jaoks. Klassil on meetod sõnekujul esitamiseks ning staatiline meetod *getSubFormula* alamvalemi valimiseks eelnevalt saadud sõne alamsõne järgi. Viimatinimetatud meetod saab argumentideks *SetFormula* tüüpi avaldise ja alamsõne alguse ja lõpu positsioonid täieliku sõne suhtes. Sobivuse korral tagastatakse *SetFormula* tüüpi viit alamavaldisele. Viite kasutamine võimaldab alamavaldist muutes kajastada muudatusi ka täielikus puus, isegi kui alamavaldis ühtib sellega.

Tehetest on täiendi jaoks klass *Complement*, mille operaatorite arv on üks ja parem haru on tühi. Ühendi, ühisosa, vahe ja sümmeetrilise vahe jaoks on vastavalt klassid *Union*, *Intersection*, *Difference* ja *SymmetricDifference*. Lausearvutuse tehted laiendavad abstraktset klassi *SetSentenceFormula*. Eituse jaoks on klass *SetNegation*, mis on paremassotsiatiivne ja tühja vasaku haruga. Konjunktsiooni ja disjunktsiooni jaoks on vastavalt klassid *SetConjunction* ja *SetDisjunction*. Üksiku muutuja jaoks on klass *SetVariable*, mille mõlemad harud on tühjad.

Sõnekujul valemist *SetFormula* objekti loomiseks on klass *SetParser*, mille on staatiline meetod *parse*, mis saab ainsaks argumendiks avaldise sõne kujul. Parsimine põhineb Djikstra sorteerimisjaama algoritmil (*shunting-yard algorithm*)[14]. Sõne töödeldakse sümbolhaaval. Õnnestumise korral tagastatakse *SetFormula* objekt, vea korral visatakse erind, mis püütakse kinni ja kuvatakse kasutajale vastav veateade.

3.4. Teisendusreeglite rakendamine

Teisendusreeglite kasutamiseks on klass *Rule*. Üldiselt on reeglid kujul $A = B$, sellepärast on klassil väljad *left* ja *right*, kus hoitakse vastavalt sõnekujul avaldisi A ja B. Lisaks on lipp, mis määrab, kas valem on kahesuunaline ehk kas saab rakenda reeglit $B = A$.

Java-põhisel lausearvutuse teisendusprogrammil on iga reegli jaoks loodud oma meetod. See tähendab, et uue reegli lisamiseks on vaja ka koodi kirjutada. Antud töös otsustati läheneda üldisemalt. Uue reegli lisamiseks tuleb luua klassi *Rule* isend. Konstruktor saab sõnekujul reegli ja tõeväärtuse, kas valem kehtib mõlemat pidi. Kui reegel vastab teatud tingimustele, näiteks ei tooda valemiga sisse uusi muutujaid, oskab programm reeglit automaatselt kasutada.

Reegli rakendamiseks on meetod *applyRule*, mis saab argumentideks *SetFormula* tüüpi valemi *formula* ja reegli rakendamise suuna. Esimese etapina luuakse reegli mõlema poole jaoks sõnedest *SetFormula* objektid, saades muutujad *left* ja *right*. Kui reeglit rakendatakse vastupidises suunas, vahetatakse need muutujad omavahel. Järgmiseks üritatakse *formula* sobitada vasaku poolega. Selleks luuakse assotsiatiivne massiiv, kus iga reeglis esineva lausemuutuja jaoks on vastavusse seatud *formula* vastav alamvalem. Lõpuks asendatakse reegli vasakus pooles esinevad lausemuutujad varem leitud vastetega.

Olgu meil näiteks reegel $F \& G = G \& F$. Reegli vasakuks pooleks on $F \& G$ ja paremaks pooleks $G \& F$. Olgu vaja reeglit rakendada avaldisel $\neg A \& B$. Sobitamise etapil seatakse reegli vasaku poole lausemuutujale F vasteks alamvalem $\neg A$ ja G-le vasteks B. Kui saadud asendused teha reegli paremas pooles, on tulemuseks avaldis $B \& \neg A$.

3.5. Ülesannete esitus programmis

Kõikide ülesannete jaoks on klass *Task*, kus on väljad avaldise, vigade ja punktide jaoks. Konkreetse lahenduse jaoks on klass *Answer*, kus on väljad lahendusaegade, vigade arvu ja lahenduse enda jaoks. Viimast hoitakse serialiseeritud PHP massiivina. See võimaldab kergesti andmeid andmebaasist programmi ja vastupidi transportida.

Kui tudeng asub uut ülesannet lahendama, loetakse serveris *Task* objektist sõnekujul ülesanne, näiteks $A \cup B = B \cup A$. Seejärel näidatakse kasutajale lahendamise lehte. Kogu lahenduskäigu kontroll toimub serveris, et välistada pettused. Olemasolevate lausearvutuse ülesannete puhul toimub kogu kontroll kliendi pooles ja serverisse saadetakse vaid tulemus ehk tehtud vigade arv. Lisatud hulgateooria ülesannete puhul valib tudeng kliendi poolel alamavaldise ja reegli ning see info saadetakse serverisse. Seal kontrollitakse, kas saadud info on korrektne, näiteks kas antud reegel eksisteerib. Andmebaasist loetakse *Answer* objekti senine lahenduskäik ja leitakse hetkel teisendatav avaldis, millest üritatakse soovitud alamvalem valida ja sellele reeglit rakendada. Tekkinud vead salvestatakse kohe. Lõpuks tagastatakse teisenduse tulemus või veateade ja tudeng saab lahenduskäiku jätkata. Kui tudeng annab ülesande lõppvastuse, toimub analoogne protsess. Kui vastus on õige, loetakse ülesanne lahendatuks ja tudeng suunatakse tagasi ülesannetekogu lehele. Vastasel juhul tagastatakse veateade ja tudeng peab jätkama lahendamist.

3.6. Paigaldamine

Käesoleva bakalaureusetöö raames täeindatud veebirakenduse jooksutamiseks on tarvis standardinstallatsiooniga Apache serverit ning MySQL andmebaasisüsteemi. Lisas 1 oleva CD sisu tuleb kopeerida serveri veebikataloogi. Konfiguratsiooni seaded asuvad failis *conf/untime.conf.php*. Seal tuleb muuta kuute parameetrit. SITE_URL on rakenduse ja DOM_URL kausta *CODE/libs/* veebiaadress. Ülejäänud neli parameetrit on andmebaasi

seadistused: MYSQL_HOST, MYSQL_USERNAME, MYSQL_PASSWORD ja MYSQL_DATABASE on vastavalt MySQL serveri aadress, kasutajanimi, parool ning andmebaasi nimi. Tühi andmebaas tuleb luua käsitsi.

Kui failid on serveri veebikataloogi kopeeritud, tuleb kaustale *files/* anda maksimaalsed õigused. Nimetatud kaustas hoitakse ajutisi faile, mis kiirendavad keskkonna toimimist. Seejärel tuleb minna veebibrauseris rakenduse veebiaadressile SITE_URL/*install.php?key=2398041687*. Turvalisuse parameetrit *key* saab muuta failis *install.php*. Installeerimisskript loob andmebaasi vajalikud tabelid ning lisab esimese kasutaja õppejõu vaatele. Sisse saab logida aadressilt SITE_URL*/admin/* kasutajanimega **test** ja parooliga **Test1234**. Hiljem võib selle kasutaja eemaldada.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli täiendada olemasolevat veebistöötavat lausearvutuse programmi hulgateooria ülesannetega. Selleks lisati rakendusele kaks hulgateooria teisendamisega seotud ülesannete tüüpi: samaväärsuse ja alamhulga seose tõestamine. Mõlema ülesandetüübi puhul on üldine lahenduskäik sarnane: kuvatakse kaks avaldist, mida saab üksteisest sõltumatult teisendada. Selleks peab lahendaja valima alamavaldise ja rakendama sellele mõnda eeldefineeritud reeglit. Kui avaldised on teisendatud sobivale kujule, tuleb esitada vastus. Alamhulga seose tõestamise ülesannetes tuleb seejärel veel põhjendada, miks seos kehtib. Lahenduse käigus tehtud vead kuvatakse kohe kasutajale ning tehtud vigade arv salvestatakse ka õppejõule. Viimane saab vaadata tudengi tulemusi ülesannete lõikes ning samuti konkreetseid lahendusi.

Töös kirjeldati õpiprogrammidest saadavat kasu, olemasolevaid programme, lisatud ülesandetüüpide hulgateoreetilist tausta ning nende implementatsiooni rakenduses. Samuti tutvustati uusi ülesande tüüpe õppejõudude ja tudengite vaatepunktist.

Loodud ülesannetetüübid on kasulikud nii tudengitele kui õppejõududele. Ülesannete läbi lahendamine aitab üliõpilasel materjali omandada. Lisaks lõppvastuse kontrollimisele annab programm ka vahepealset tagasisidet, näiteks kui üritatakse kasutada mõnda sobimatut teisendusreeglit või eksitakse alamavaldise valimisega. Õppejõud saavad programmi abil ülesandeid koostada ja tudengite tulemusi vaadata.

Set Theory Proof Exercise Types for Discrete Mathematics Web Application

Bachelor Thesis

Timmu Ründal

Summary

There are many propositional calculus programs used in the course of Elements of Discrete Mathematics in University of Tartu. The aim of these applications is to help students acquire practical skills and get instant feedback for their solutions. There were not any programs for set theory exercises, so the goal of the thesis is to add new exercise types to the existing discrete mathematics web application.

The two added set theory exercise types were for proving equality of set theory expressions and subset relationship. For this, programmatic representation of set theory and propositional calculus operations were created. Also, many transformation rules were implemented to re-express formulae using different operations.

The thesis explains the advantages of using programs in Elements of Discrete Mathematics class, gives short overview of set theory exercises and describes the implementation of added types and shows how they work in action.

To implement new types, all of technologies of the existing systems were used. These include PHP and MySQL on the server-side and JavaScript, HTML and CSS on the client-side. PHP framework named DomFramework and jQuery, the JavaScript library, were used.

Viited

Kõik viidatud veebilehed on vaadatud 14.05.2012.

[1] „Tartu Ülikooli õppeinfosüsteem“, https://www.is.ut.ee/pls/ois/tere.tulemast

[2] A. Lukk. Kaasaegse kasutajaliidesega veebikeskkond tõeväärtustabeli lahendamiseks Bakalaureuse töö, Tartu 2010

[3] „Proof Designer“, http://www.cs.amherst.edu/~djv/pd/pd.html

[4] Daniel J. Velleman. How To Prove It. Cambridge University Press, 2009

[5] Peeter Oja. Hulgateooria. Tartu, 2006

[6] M. Kilp, U. Nummert. Hulgateooria elemendid. Tartu, 1994

[7] „PHP: Preface – Manual“, http://www.php.net/manual/en/preface.php

[8] „PHP: History of PHP – Manual“, http://www.php.net/manual/en/history.php.php

[9] „PHP: What can PHP do? – Manual“, http://www.php.net/manual/en/intro-whatcando.php

[10] „About JavaScript – MDN“, https://developer.mozilla.org/en/JavaScript/About_JavaScript

[11] „Introduction – MDN“, https://developer.mozilla.org/en/Gecko_DOM_Reference/Introduction

[12] „jQuery: The Write Less, Do More, JavaScript Library“, http://jquery.com/

[13] „AJAX Tutorial“, http://www.w3schools.com/ajax/default.asp

[14] „Shunting-yard algorithm - Wikipedia, the free encyclopedia“, http://en.wikipedia.org/wiki/Shunting_yard_algorithm

Lisad

Lisa 1: CD plaat programmi lähtekoodiga