TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika eriala

Sander Stroom

Lause- ja predikaatarvutuse valemite teisendamise õpiprogrammi täiendamine

Bakalaureusetöö (6 EAP)

Juhendaja: dots. Rein Prank

Autor: ..... „...“ mai 2013
Juhendaja: ..... „...“ mai 2013

Lubada kaitsmisele
Professor: ..... „...“ mai 2013

TARTU 2013

Sisukord

1. Avaldiste teisendamise harjutused matemaatilise loogika kursustes 4
1.1. Avaldiste teisendamise ülesannete liigid 4
1.2. Diskreetse matemaatika elemendid (DME) 5
1.3. Sissejuhatus matemaatilisse loogikasse (SML) 6
2. Õpiprogrammi eelmise versiooni kirjeldus 7
2.1. Programmi üldine kirjeldus 7
2.2. Ülesande lahenduse salvestamine 9
2.3. Ülesannete lahendamise tulemuste näitamine 10
2.4. Viimase versiooni puudused 12
2.5. Bakalaureusetöö ülesanne 12
3. Bakalaureusetöö raames tehtud täiendused eelmisele versioonile 14
3.1. Täienduste nimekiri 14
3.2. Sisu kirjeldus 14
3.2.1. Ülesande lahendamise andmete salvestamise täiendamine 14
3.2.2. Programmile tõlkimisvõimaluse lisamine 16
3.2.3. Automaatlahendamine ja soovitamine 16
3.2.4. Kahe üleliigse teisendusreegli eemaldamine 18
3.2.5. Vahetu ülesannete lahendamise keskkonna täiendus 18
3.3. Automaatlahendaja realisatsiooni kirjeldus 19
3.3.1 Automaatlahendaja kahe tehte kaudu avaldamise kohta 20
3.3.2 Automaatlahendaja täieliku disjunktiivse normaalkuju kohta 21
Kokkuvõte 24
Summary 25
Kirjandus 26
Lisad 27
Lisa 1. Tõlkefailide loomise juhend 27
Lisa 2. CD programmi, selle lähtekoodi ja tõlkefailide loomise juhendiga 28

Sissejuhatus

1987. aastal valmis H. Viiral lausearvutuse valemite teisendamise programmi esimene versioon [1]. Seda kasutati matemaatilise loogika praktikumides lausearvutuse valemite teisendamise õpetamisel. 2003. aastal valmistas Vahur Vaiksaar oma bakalaureusetööna õpiprogrammi, millega saab teisendada nii lause- kui ka predikaatarvutuse valemeid [2]. Võrreldes varasema programmiga oli parandatud mitmeid puudujääke ja lisatud funktsionaalsust. Uuemat programmi kasutatakse Tartu ülikoolis ainetes Diskreetse matemaatika elemendid (edaspidi DME) ja Sissejuhatus matemaatilisse loogikasse (edaspidi SML). V. Vaiksaare õpiprogrammil esineb teatud puudusi, mille eemaldamine on käesoleva bakalaureusetöö teemaks [3].

Bakalaureusetöö eesmärgiks on täiendada V. Vaiksaare programmi, parandades ilmnenud vigu ja lisades õpiprogrammile funktsionaalsust. Tehtavatest täiendustest kõige mahukam osa on automaatlahendaja loomine. Lahendaja peab vastavalt ülesandetüübile teisendama valemeid vajalikule kujule ning kasutaja soovi korral juhendama teda järgmise teisendussammu tegemisel. Valemite teisendamisel tehtud teisendussammud ja tekkinud vead tuleb korralikult salvestada ning kasutaja soovi korral talle sellest ülevaade anda. Õpiprogrammile peab ka lisama võimaluse muuta kasutuskeelt.

Töö on jaotatud kolme peatükki. Esimeses peatükis tutvustatakse õppeaineid, milles lause- ja predikaatarvutuse valemite teisendamist õpetatakse, ning ülesandetüüpe, mille lahendamise üliõpilased omandavad. Teine peatükk kirjeldab V. Vaiksaare õpiprogrammi esialgset versiooni ja toob välja selle tähtsamad puudused. Kolmandas peatükis on esitatud programmi olulisemad täiendused ning kirjeldatakse põhjalikumalt loodud automaatlahendaja tööpõhimõtet. Bakalaureusetööga on kaasas CD, millele on kirjutatud programm, selle lähtekood ning tõlkefailide loomise kasutusjuhend (vt. lisa 2).

1. Avaldiste teisendamise harjutused matemaatilise loogika kursustes

1.1. Avaldiste teisendamise ülesannete liigid

Esialgu õpetati Tartu ülikoolis lause- ja predikaatvalemite teisendamist teise kursuse aines Sissejuhatus matemaatilisse loogikasse. Kuna osadel esimese semestri ainetel läheb vaja teadmisi lausearvutusest, hakati lausearvutuse ülesandeid õpetama ka aines Diskreetse matemaatika elemendid.

Tartu ülikooli matemaatilise loogika ainetes lahendatavad lausearvutuse valemite teisendamise ülesannete tüübid on järgmised.

- avaldada valem konjunktsiooni ja eituse kaudu
- avaldada valem disjunktsiooni ja eituse kaudu
- avaldada valem implikatsiooni ja eituse kaudu
- leida valemi täielik disjunktiivne normaalkuju
- leida valemi täielik konjunktiivne normaalkuju
- viia valem disjunktiivsele normaalkujule
- viia valem konjunktiivsele normaalkujule
- leida valemi eitusega prefikskuju
- viia eitused vahetult lausemuutujate ette

Praktikumides lahendatavad predikaatarvutuse valemite teisendamise ülesannete tüübid on järgmised.

- avaldada valem konjunktsiooni, eituse ja üldisuse kvantori kaudu
- avaldada valem konjunktsiooni, eituse ja olemasolu kvantori järgi
- avaldada valem disjunktsiooni, eituse ja üldisuse kvantori kaudu
- avaldada valem disjunktsiooni, eituse ja olemasolu kvantori järgi
- avaldada valem implikatsiooni, eituse ja üldisuse kvantori kaudu
- avaldada valem implikatsiooni, eituse ja olemasolu kvantori järgi
- leida predikaatloogika valemi prefikskuju

Kõigepealt õpetatakse üliõpilastele lausearvutuse valemeid teisendama ning avaldama valemeid eituse ja mõne kahekohalise tehte kaudu. Seejärel tegeletakse eituse

lausemuutujate ette viimisega ja disjunktiivsete normaalkujude leidmisega. Järgmiseks tegeletakse ka predikaatloogika ülesannetega.

Selleks, et õpetada valemi viimist täielikule disjunktiivsele normaalkujule (TDNK) teisendamiste teel, kasutatakse järgmist algoritmi [4].

1) elimineerida implikatsioonid ja ekvivalentsid;
2) viia eitused vahetult lausemuutujate ette;
3) korrutada disjunktsioonid läbi;
4) kaotada samaselt väärad konjunktsioonid ja sama liikme mitmekordsed esinemised konjunktsioonides;
5) lisada konjunktsioonidele puuduvad muutujad;
6) järjestada muutujad konjunktsioonides ja kaotada korduvad konjunktsioonid.

1.2. Diskreetse matemaatika elemendid (DME)

Informaatika eriala õppekavas on Diskreetse matemaatika elemendid esimene õppeaine, milles tutvutakse matemaatilise loogikaga. Seda ainet õpivad valdavalt esimese semestri tudengid. Kuna teistes ainetes on vaja teadmisi lausearvutusest, on vajadus seda üliõpilastele varakult õpetada. Kõigepealt tegeletakse tõeväärtustabelitega, seejärel valemite teisendamisega.

Aines DME õpetatakse lahendama järgmiseid ülesandetüüpe [5]:

- avaldada valem konjunktsiooni ja eituse kaudu
- avaldada valem disjunktsiooni ja eituse kaudu
- avaldada valem implikatsiooni ja eituse kaudu
- leida valemi täielik disjunktiivne normaalkuju

Aine juhendajad on märganud, et mõned üliõpilased teevad teisendamisel palju ebavajalikke teisendussamme. Vahepeal jäetakse valemi teisendamisel mõni etapp vahele, mistõttu venib teisendamine pikemaks. Mõnikord ei lihtsustata valemit õigel ajal. Mõnel juhul tehakse ka selliseid teisendusi, mis teisendamise keerulisemaks muudavad. Võiks arvata, et üliõpilane teeb teisendusi suvalises järjekorras, lootes, et valem teisendub soovitud kujule.

Enne 2004. aastat kasutati Tartu ülikoolis lausearvutuse valemite teisendamise õpetamisel H. Viira loodud programmi [1]. Sellel oli olemas funktsionaalsus, mis soovitas kasutajale, kuidas avatud ülesannet lahendada. V. Vaiksaare loodud õpiprogrammil selline võimalus puudub. On võimalik, et selle funktsionaalsuse lisamisega muutub õppimisprotsess üliõpilastele lihtsamaks.

1.3. Sissejuhatus matemaatilisse loogikasse (SML)

Aines Sissejuhatus matemaatilisse loogikasse tuletatakse meelde aines DME õpitut ning lisaks sellele õpetatakse teisendama valemeid konjunktiivsele normaalkujule. Lisaks lausearvutuse avaldiste teisendamisele tegeletakse predikaatloogika ülesannetega [6].

Aine SML on mõeldud õppimiseks teisel kursusel. Üliõpilaste oskused valemite teisendamisel on märgatavalt paremad. On õpitud õigeid teisendusreegleid kasutama ja järgitakse üldisi lahendamise algoritme. Mõnel juhul, kui valemit on lihtsam teisendada ebatavalise teisendussammuga, tehakse ka selliseid teisendusi. Seepärast on SML kursuses käsitletavate ülesandetüüpide puhul vajadus nõuannete järele oluliselt väiksem.

2. Õpiprogrammi eelmise versiooni kirjeldus

2.1. Programmi üldine kirjeldus

Õpiprogrammi on kirjutatud Vahur Vaiksaar oma bakalaureusetöö raames 2003. aastal. Selleks kasutas ta programmeerimiskeelt Java. Programm koosneb kahest osast: ülesannete loomise keskkonnast ja ülesannete lahendamise keskkonnast.

Kuna põhiliselt jääb ülesannete lahendamise protsess samaks, nagu see oli programmi esialgses versioonis, on käesolevas peatükis kirjeldatud, kuidas töötab programmi eelmine versioon.

Ülesannete loomise keskkonnas saab luua ülesannete kogusid. Igal ülesandel saab määrata erinevaid atribuute:

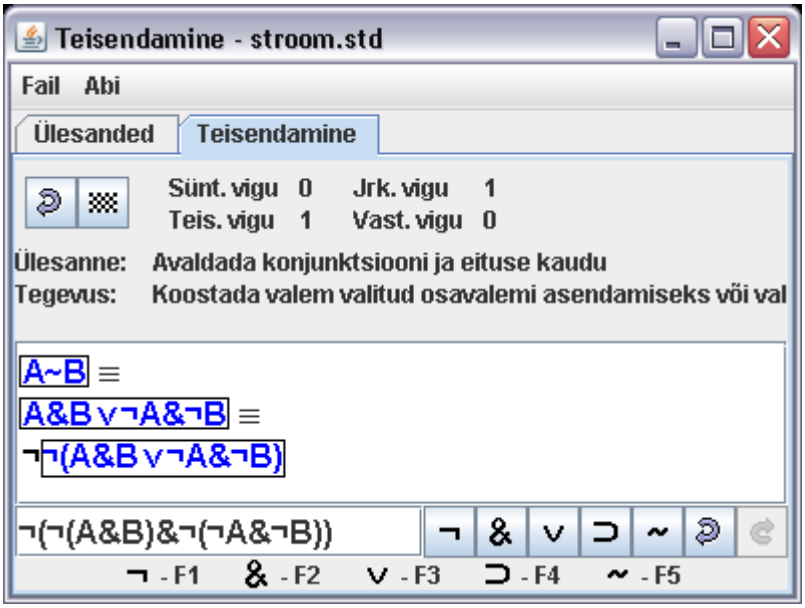
- ülesande tüüp: millisele kujule peab etteantud valemi teisendama
- ülesande tekst
- valem: valemi võib välja kirjutada või anda juhuslikule valemi generaatorile ette parameetrid, mille järgi valem genereeritakse.
- teisendamisviis: kas valemit teisendatakse vahetult või reeglipõhiselt
- maksimaalsed lubatud vigade arvud nelja erinevat tüüpi vigade kohta: mitu viga võib kasutaja iga veatüübi kohta lahendamisel teha, et ülesanne läbituks loetaks.

Ülesannete lahendamise keskkonnas kasutatakse loodud ülesannete faili, mille põhjal luuakse uus õpilasfail. Kasutaja peab faili luues programmi sisestama oma ees- ja perekonnanime ning seminari rühma. Lisaks sellele hoitakse õpilasfailis informatsiooni erinevate ülesannete lahendamise kohta.

Igal ülesandel on määratud, millises keskkonnas seda lahendama peab. Need keskkonnad on nimetatud vahetuks ja reeglipõhiseks keskkonnaks.

Vahetus keskkonnas teisendades peab kasutaja kursoriga valima osavalemi, mida ta tahab teisendada, vajutama nupule Enter ning kirjutama valitud osavalemi asemele samaväärse valemi. Kui tegu on korrektse asendusega, salvestatakse lahendussamm ja kasutaja saab teisendusega jätkata.

Joonisel 1 on näidatud ülesande lahendamine vahetus keskkonnas. Igal real on kirjas valemi kuju, mis saadi pärast edukat teisendust. Valemi ¬¬(A&B∨¬A&¬B) teisendamisel on kasutaja valinud osavalemi ¬(A&B∨¬A&¬B) ning tahab seda asendada valemiga ¬(¬(A&B)&¬(¬A&¬B)). Kasutaja on lahendamise käigus teinud kaks erinevat tüüpi viga.



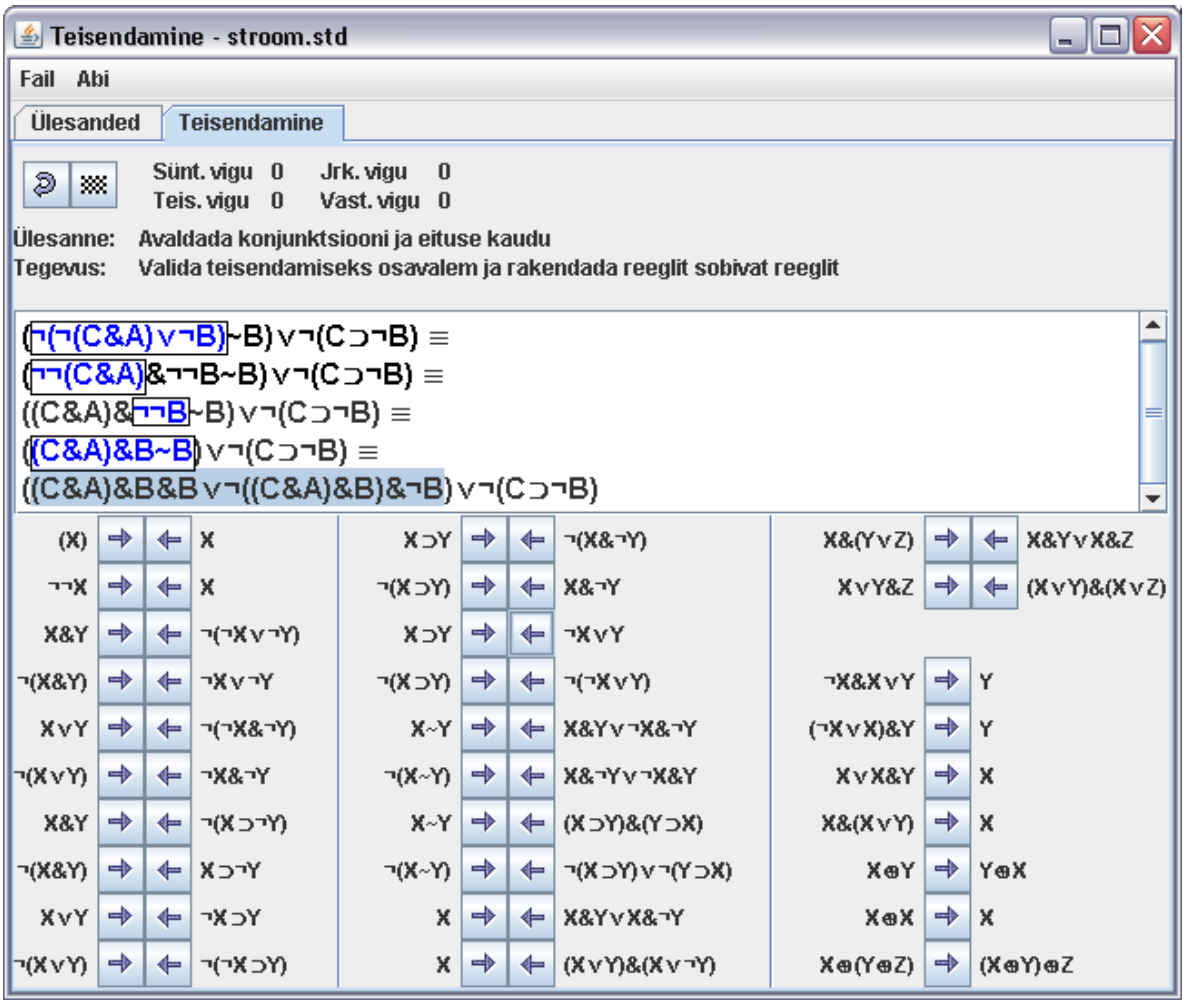
Joonis 1. Ülesande lahendamine vahetus keskkonnas.

Reeglipõhises keskkonnas peab kasutaja samamoodi valima osavalemi, kuid uue valemi kirjutamise asemel peab ta valima etteantud reeglite kogust reegli, mida ta tahab osavalemile rakendada. Teisenduse teeb sellisel juhul ära programm. Kui programm leiab, et teisenduse tegemiseks on vaja kasutajalt lisainformatsiooni, avaneb dialoogiaken, milles saab seda lisada.

Joonisel 2 on näidatud ülesande lahendamine reeglipõhises keskkonnas. Kasutaja on valinud esialgsest valemist osavalemi ¬(¬(C&A)∨¬B) ja rakendanud sellele reeglit ¬(X∨Y) → ¬X&¬Y. Seejärel on ta kaotanud eelmise reegli rakendamise tulemusel tekkinud kahekordsed eitused. Järgmisena on kasutaja valinud osavalemi (C&A)&B~B ja rakendanud sellele reeglit X~Y → X&Y∨¬X&¬Y. Järgmiseks on ta valinud osavalemi (C&A)&B&B∨¬((C&A)&B, aga ei ole veel ühtegi reeglit rakendanud.

Mõlemas ülesande lahendamise keskkonnas toimub vastuse esitamine ühtemoodi. Kui kasutaja arvab, et ülesanne on lahendatud, peab ta vajutama ülesande esitamise nuppu.

Seejärel kontrollitakse, kas vastusena antud valem on sellisel kujul, nagu ülesande tüüp seda nõuab, ning antakse kasutajale vastav teade.



Joonis 2. Ülesande lahendamine reeglipõhises keskkonnas.

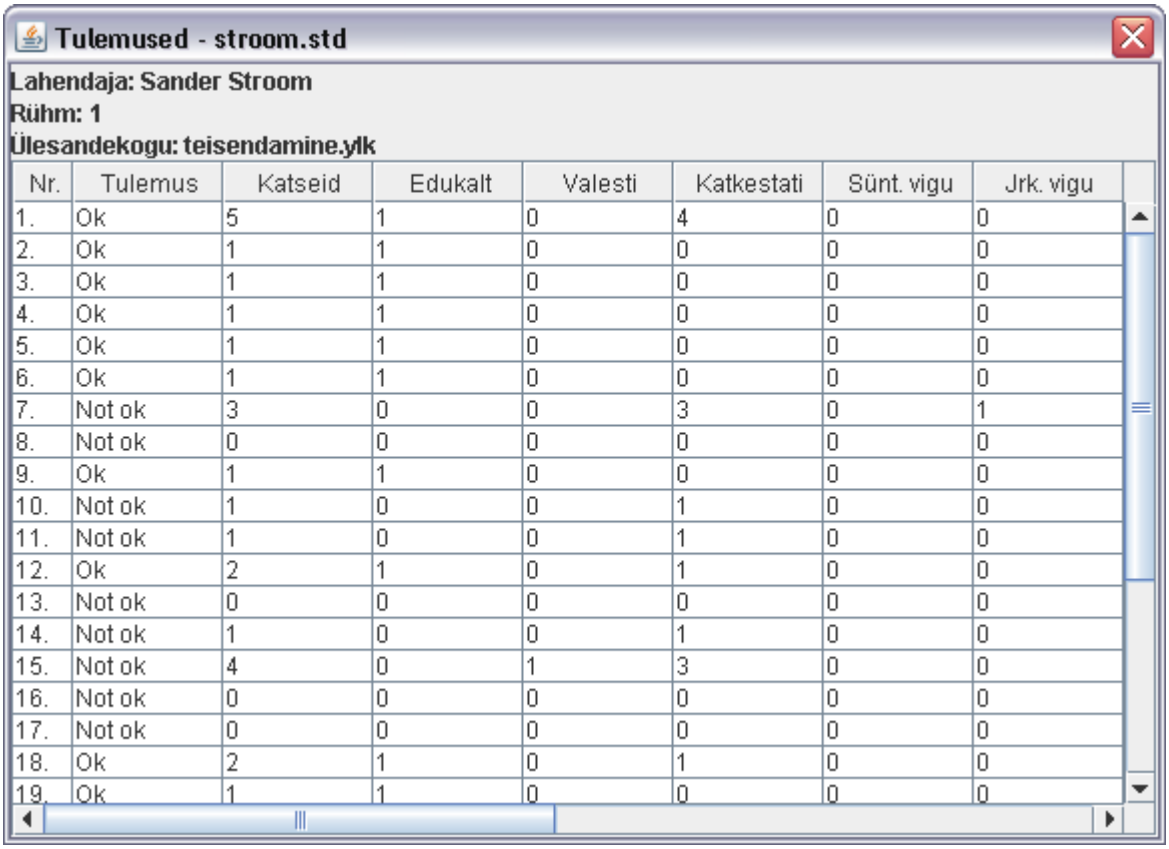
2.2. Ülesande lahenduse salvestamine

Kui kasutaja mõnda ülesannet lahendab, salvestatakse iga sammu järel tekkinud andmed õpilasfaili.

- Ülesande avamisel lisatakse ülesande logisse alustamise kuupäev ja kellaaeg ning esialgne valem sõne kujul.
- Korrektse teisendussammu järel lisatakse ülesande logisse teisenduse abil saadud uus valem.
- Juhul, kui valemi teisendamise ajal tehti vigu, suurendatakse ülesande vastava veatüübi vigade arvu loendajat. Vea kohta täiendavat informatsiooni ei salvestata.

2.3. Ülesannete lahendamise tulemuste näitamine

Kasutaja saab vaadata enda tulemusi ülesannete lahendamiste kohta. Tulemused kantakse tabelisse, kus igal real on ülesanne ja tulpades ülesandele vastavad andmed.



Nr.	Tulemus	Katseid	Edukalt	Valesti	Katkestati	Sünt. vigu	Jrk. vigu
1.	Ok	5	1	0	4	0	0
2.	Ok	1	1	0	0	0	0
3.	Ok	1	1	0	0	0	0
4.	Ok	1	1	0	0	0	0
5.	Ok	1	1	0	0	0	0
6.	Ok	1	1	0	0	0	0
7.	Not ok	3	0	0	3	0	1
8.	Not ok	0	0	0	0	0	0
9.	Ok	1	1	0	0	0	0
10.	Not ok	1	0	0	1	0	0
11.	Not ok	1	0	0	1	0	0
12.	Ok	2	1	0	1	0	0
13.	Not ok	0	0	0	0	0	0
14.	Not ok	1	0	0	1	0	0
15.	Not ok	4	0	1	3	0	0
16.	Not ok	0	0	0	0	0	0
17.	Not ok	0	0	0	0	0	0
18.	Ok	2	1	0	1	0	0
19.	Ok	1	1	0	0	0	0

Joonis 3. Õpilasfaili stroom.std kohta näidatavad tulemused (joonisel ei ole kõiki veerge näha).

Tabelis näidatavad andmed on järgmised.

- Ülesande number
- Tulemus: kas ülesanne on lahendatud või mitte
- Katseid: mitu korda on ülesanne lahendamiseks avatud
- Edukalt: katsete arv, mille korral on jõutud lubatud vigade arvu piires lahenduseni, ning mille korral on vastus esitatud
- Valesti: katsete arv, mille korral on lahendus leitud, aga tehti liiga palju vigu
- Katkestati: katsete arv, mille korral lahendamine jäi pooleli
- Sünt. vigu: vigade arv, mis vastab osavalemite valesti valimistele
- Jrk. vigu: vigade arv, mis vastab ebakorrektselt valitud osavalemite valimistele
- Teis. vigu: vigade arv, mis vastab valede teisenduste tegemistele

- Sulud lisamata (Jrk.): vigade arv, mis vastab olukordadele, kus osavalemi asendamisel ei lisatud asendava valemi ümber sulge
- Samaväärne (Jrk.): vigade arv, mis vastab olukordadele, kus tehakse küll samaväärne asendus, aga eksitakse tehete järjekorra suhtes
- Vast. vigu: vigade arv, mis on tekkinud vastuse esitamisel
- Viimane katse: kellaaeg ja kuupäev, mis vastab ajahetkele, millal ülesanne viimane kord avati
- Viimane edukas katse: kellaaeg ja kuupäev, mis vastab viimasele sellisele ülesande avamise hetkele, mille korral vastav ülesanne edukalt lahendati

Iga ülesande kohta saab vaadata logi, milles kuvatakse, millistel aegadel ülesande lahendamiskatsega alustati ning iga lahendussammu kohta kirjutatakse välja, milliseks muutus valem pärast edukat teisendust.

Logi nägemiseks on vaja tulemuste tabelist kursori abil valida soovitud ülesande rida ja teha sellel topeltklõps. Selle peale avaneb uus aken, millel näeb kõiki ülesande lahendamise katseid.

Joonisel 4 on kujutatud ülesanne 18. Ülesanne on avatud lahendamiseks kaks korda. Esimesel korral ei ole tehtud ühtegi teisendust ja ülesande lahendamine on jäetud pooleli. Teisel korral on ülesanne lõpuni lahendatud. Kuna logist ei ole näha, kas ülesanne ka vastusena esitati, tuleb seda vaadata tulemuste tabelist (vt. joonis 3). Kuna tabelis on tulemuse lahtrisse kirjutatud „Ok“, on ülesanne edukalt lahendatud.

Joonis 4. Ülesande lahenduste logi. Tehtud on 2 katset, neist 1 edukalt.

2.4. Viimase versiooni puudused

2003. aastal loodud programmi üheks suurematest puudustest on see, et ülesande lahenduskäiku ei kujutata ega salvestata piisavalt detailselt selleks, et kasutaja või aine juhendaja saaks uurida, millega valemite teisendamisel kõige rohkem probleeme tekib ning milliseid lahendussamme tehti. Logisse salvestatakse ainult edukad lahendussammud ja iga tekkinud vea puhul suurendatakse veatüübile vastava loendaja väärtust ühe võrra, salvestamata muud informatsiooni tekkinud vea kohta. Logisse ei salvestata seda, milline osavalem teisendussammu tehes esialgsest valemist valiti ning reeglipõhise keskkonna puhul ei jäeta meelde, millist reeglit valitud osavalemile rakendati.

Puudub lihtne võimalus programmi tekste teistesse keeltesse tõlkida. Kõik fraasid, mida graafilise liidese kaudu näidatakse, on koodi sisse kirjutatud.

Selleks, et valemite teisendamise õppimine oleks kergem, on vaja lisada programmile automaatlahendaja, mille abil oleks võimalik kasutajale anda juhiseid järgmiste lahendussammude sooritamisel.

2.5. Bakalaureusetöö ülesanne

Bakalaureusetöö ülesandeks on teha 2003. aastal loodud programmile järgmised täiendused.

- On vaja muuta õpilasfaili struktuuri selliseks, et oleks võimalik salvestada rohkem informatsiooni ülesannete lahendamise kohta: igal teisendussammul salvestatakse ka valitud osavalem ja reeglipõhise lahendamisega puhul ka rakendatud reegel.
- Peab tegema võimalikuks programmi tõlkimise teistesse keeltesse. Tõlkimine peaks toimuma programmiväliste tõlkefailide kaudu, et programmi oleks lihtsam tõlkida. Lisaks sellele on vaja koostada juhend tõlkefailide loomiseks.
- Programmile on vaja lisada automaatlahendaja ja nõuandja, mis oskaks kasutajale järgmise lahendussammu kohta vihjeid anda.

Kuna automaatlahendajat on vaja eelkõige aine DME kursustes, luuakse automaatlahendajad neljale põhilisele ülesandetüübile, mille lahendamise üliõpilased endale selgeks peavad tegema.

- leida valemi täielik disjunktiivne normaalkuju
- avaldada valem konjunktsiooni ja eituse kaudu

- avaldada valem disjunktsiooni ja eituse kaudu
- avaldada valem implikatsiooni ja eituse kaudu

Kuna seda õpiprogrammi kasutatakse ka üliõpilaste teadmiste hindamisel, tekib vajadus, et kontrolltööde ajal ei antaks kasutajatele vihjeid valemite teisendamise kohta. Seetõttu peab täiendama ka ülesannete koostamise keskkonda ning lisama koostatavatele ülesannetele atribuudi, mille abil saab määrata, kas programm võib kasutajat abistada.

3. Bakalaureusetöö raames tehtud täiendused eelmisele versioonile

3.1. Täienduste nimekiri

Bakalaureusetöö raames on tehtud järgnevad muudatused.

- Õpilasfaili struktuur on muudetud selliseks, et sellesse saaks salvestada iga ülesande lahendamise kohta rohkem informatsiooni. Ülesannete lahendamiskäigu kuvamisel näidatakse lisandunud andmeid.
- Programm on muudetud kergemini tõlgitavaks. Programmi kaudu saab valida kasutatavaid keelefaile. On koostatud juhend keelefailide loomiseks.
- On loodud automaatlahendaja ja järgmise lahenduskäigu soovitaja aines DME õpetatava nelja ülesandetüübi kohta.
- Reeglipõhisest teisendamiskeskkonnast on eemaldatud kaks ülearust reeglit.
- Vahetusse teisendamiskeskkonda on lisatud paneel, millel kuvatakse valitud osavalemit, et kasutajal oleks kergem seda asendavat valemit kirjutada.
- Programm avab ka õpilasfaile, mille nime on loomise järel muudetud.

3.2. Sisu kirjeldus

3.2.1. Ülesande lahendamise andmete salvestamise täiendamine

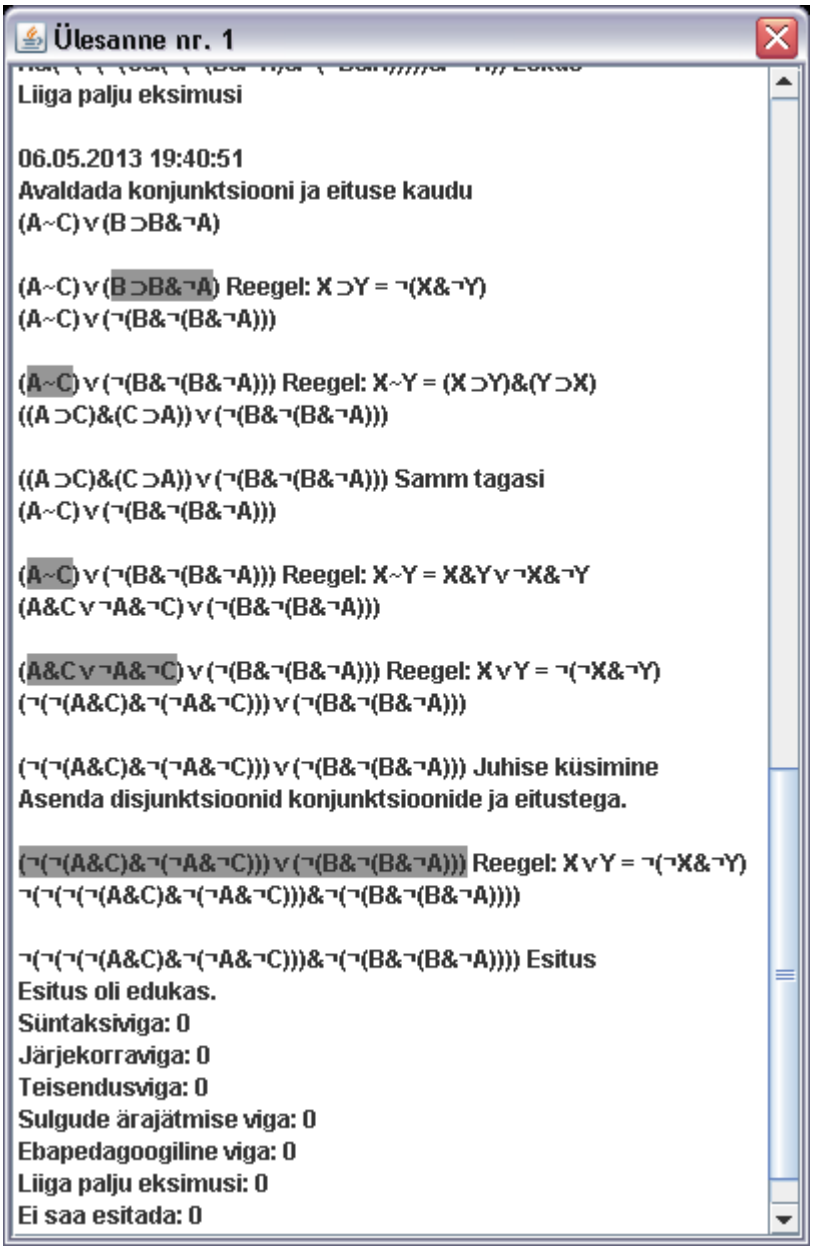
Varem näidati logis ainult ülesande lahendamise alustamise aega ja iga tehtud teisenduse sammu järel tekkinud valemit. Täiendamise järel kasutatakse ülesande teisenduse sammude salvestamiseks põhjalikumat struktuuri, milles sisalduvad järgmised kasutaja poolt tehtud tegevused.

- tehtud teisendussammud
- juhise küsimised
- sammu tagasivõtmised
- vigased teisendussammud
- vastuste esitamised

Iga tegevuse kohta salvestatakse järgmised andmed.

- kuupäev ja kellaaeg
- valem enne teisendamist

- indeksid selles valemis näitamaks, milline osavalem teisendamiseks valiti
- reeglipõhise lahendamise korral: kasutatud teisendusreegel
- korrektse teisenduse korral: teisendamise järel saadud valem
- vea tegemise korral: kasutajale tagastatud veatüüp
- nõu küsimise korral: kasutajale antud juhis



Joonis 5. Ülesande lahendamise logi uus ekraanil näitamise viis.

Joonisel 5 on näidatud ühe ülesande logi. Kasutaja on rakendanud reegleid õiges järjekorras ning ühel korral programmilt abi küsinud. Samuti on võetud tagasi ka üks lahendussamm. Ühtegi viga valemi teisendamise käigus ei tehtud.

Ülesande logi kuvamisel järgib programm järgmisi reegleid.

- Ülesande avamise kohta näidatakse lisaks avamise ajale ka ülesande teksti.
- Kõik salvestatud tegevused kuvatakse ajalises järjestuses.
- Tehtud teisendussammu puhul on eelnenud valemi juurde halli taustavärviga märgitud, milline osavalem valiti. Reeglipõhise ülesannete lahendamise keskkonna puhul on kirjutatud välja ka kasutatud reegel.
- Tehtud teisendussammu lõppu on kirjutatud pärast teisendamist saadud valem.
- Kui teisendamisel tehti viga, on teisendussammu lõpus veatüübile vastav kirje.
- Kui tegevuseks oli juhise küsimine, kuvatakse kasutajale antud juhis.
- Kui tegevuseks oli teisendussammu tagasi võtmine, kuvatakse see valem, millega kasutaja teisendamist jätkas.
- Lahenduse eduka esitamise korral näidatakse iga veatüübi kohta eraldi, mitu viga kasutaja lahendamise käigus tegi.

3.2.2. Programmile tõlkimisvõimaluse lisamine

Programmi menüüsse on lisatud võimalus valida keelefaile. Keelefaili abil asendatakse kõik programmis kasutatavad fraasid vastavate keelefaili kirjutatud fraasidega. Kuna tõlkimisel muutuvad fraaside pikkused, on muudetud graafilise liidese elemente selliseks, et nende suurused vastavalt teksti pikkusele muutuksid. Programmisisene kasutusjuhend on programmi sisse kirjutatud ja selle tõlkimine ei ole realiseeritud. Tegu on pika tekstiga ning seda peaks tõlkima tervikuna mitte fraaside kaupa.

Kuna keelefailid ei kuulu otseselt programmi juurde, on võimalus uusi keelefaile luua ilma koodi muutmata. Selleks, et tõlkija korrektse tõlkefaili looks, on programmi juurde lisatud tõlkefaili loomise juhend (vt. lisa 1).

3.2.3. Automaatlahendamine ja soovitamine

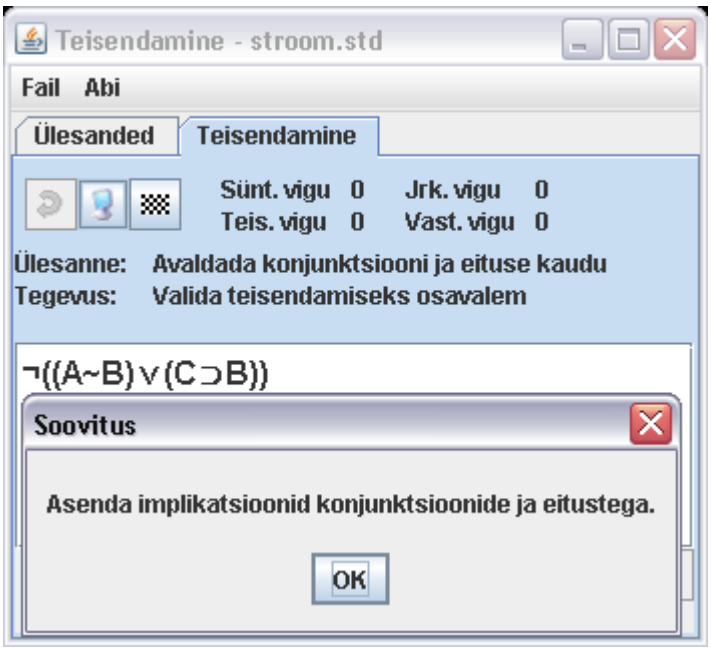
Programmi lahendamiskeskkondadesse on lisatud nupp, millele vajutades tekib juhisega aken. Selles aknas on lühike soovitus, mida järgmisena teha, et valemit õigele

kujule teisendada. Enne soovitamist lahendab automaatlahendaja ülesande ära, kasutades samasid teisendusreegleid, mida kasutaja peab reeglipõhises teisendamiskeskkonnas kasutama. Ülesanne lahendatakse lõpuni ära selleks, et saaks kontrollida, kas ülesande lahendamise algoritm töötas õigesti ning annab kasutajale korrektseid juhiseid. Kui ülesande lahendamise lõpus leiab automaatlahendaja, et saadud valem ei ole õigel kujul, avaneb aken vastava sõnumiga (vt. joonis 6).

Saades ette valemi ja ülesandetüübi, peab lahendaja kasutama programmis etteantud reegleid, et teisendada valem nõutud kujule. Kui automaatlahendaja teisendab valemi õigele kujule, annab see kasutajale vihje järgmise sammu tegemiseks.

Vahetus ülesannete teisendamise keskkonnas piisab valemi asendamisel samaväärse valemi sisestamisest. Reeglipõhises teisendusviisis peab kasutaja kindlasti kasutama etteantud reegleid. Ilmselgelt on reeglipõhine keskkond rangemate reeglitega, mistõttu peab ka automaatlahendaja valemeid teisendama, kasutades teisendusreegleid, mida kasutatakse reeglipõhises keskkonnas.

Selleks, et üliõpilaste teadmise kontrollimisel ei saaks automaatlahendajat kasutada, on lisatud ülesande struktuuri atribuut, mis määrab, kas õpiprogramm tohib ülesande lahendamisel juhiseid anda. Ülesande loomisel saab selle atribuudi väärtust määrata. Kui abistamine on välja lülitatud, ei toimu eelpool mainitud nupule vajutades mitte midagi.

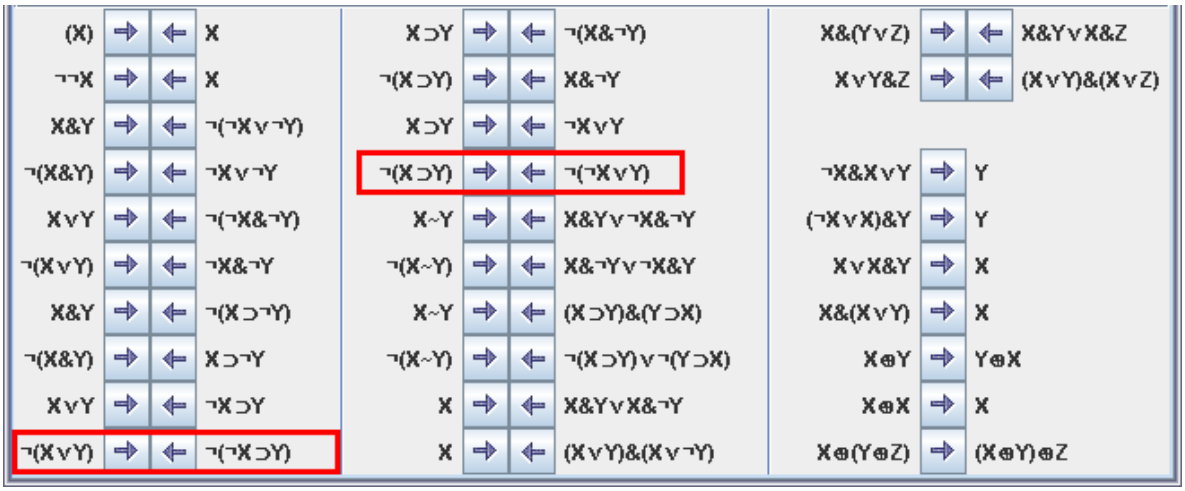


Joonis 6. Juhis järgmise lahendussammu tegemiseks.

3.2.4. Kahe üleliigse teisendusreegli eemaldamine

Õpiprogrammi eelmises versioonis on reeglipõhises ülesannete lahendamise keskkonnas kaks ülearust reeglit (vt. joonis 7). Mõlema reegli rakendamisel on eelduseks, et asendatav valem oleks ümbritsetud sulgude ja eitusega. Ka teisendamise järel saadud valem on eituse ja sulgude sees. Reeglite hulgas on olemas teisendused, mis teevad ära sama teisenduse ilma sulgude ja eitusteta.

Programmi uues versioonis on eemaldatud joonisel 7 märgitud teisendusreeglid.

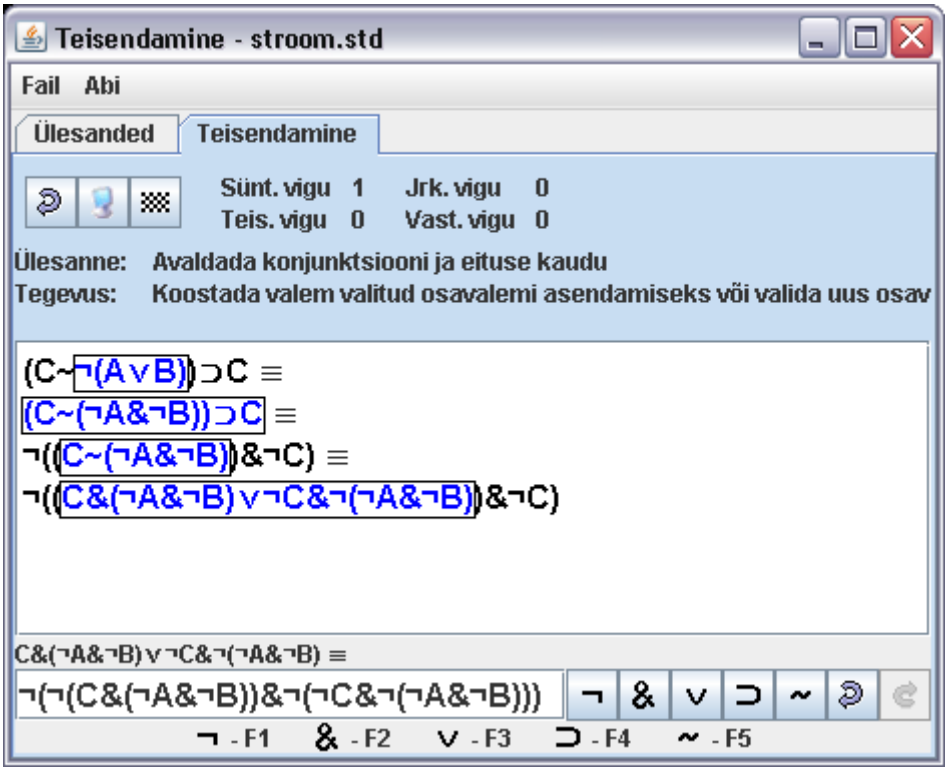


Joonis 7. Eemaldatud reeglid on ümbritsetud kastidega.

3.2.5. Vahetu ülesannete lahendamise keskkonna täiendus

Kui lahendada ülesannet vahetus keskkonnas, on kasutajal kergem, kui valitud osavalem on kuhugi eraldi välja kirjutatud. Uue valemi kirjutamiseks mõeldud ala kõrvale on loodud paneel, milles kuvatakse valitud osavalemit. Tavaliselt kuvatakse paneel tekstiväljast vasakul. Kui valitud osavalem on üle 15 sümboli pikk, kuvatakse paneel kirjutusvälja vasaku külje asemel selle kohal.

Joonisel 8 on kujutatud ülesande lahendamine vahetus keskkonnas. Kasutaja on teinud kolm edukat teisendust ning ühe süntaksivea. Üks teisendamissamm on veel pooleli. Kuna valitud osavalem on pikem kui 15 sümbolit, on see kuvatud kirjutusvälja kohal. Valitud osavalemit asendav avaldis on välja kirjutatud, kuid seda ei ole veel asendamiseks esitatud.



Joonis 8. Vahetu teisendamise keskkonna täiendus.

3.3. Automaatlahendaja realisatsiooni kirjeldus

Automaatlahendaja on realiseeritud nelja põhilise ülesandetüübi kohta, mille lahendamist õpetatakse aines Diskreetse matemaatika elemendid.

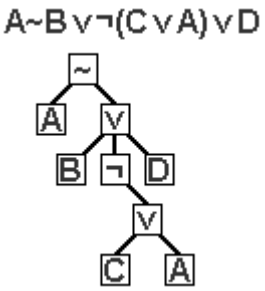
- avaldada valem konjunktsiooni ja eituse kaudu
- avaldada valem disjunktsiooni ja eituse kaudu
- avaldada valem implikatsiooni ja eituse kaudu
- leida valemi täielik disjunktiivne normaalkuju

Automaatlahendaja asub klassis *Solver*. Selles on määratud kõik võimalikud kasutajale antavad soovituse juhised ning meetodid, mille abil valemeid soovitud kujule teisendatakse ja õige juhis välja valitakse.

Teisendamise ajal käsitletakse valemit kui tehtepuud. Puul võib olla olenevalt tehetest ja nende paigutusest erinev arv harusid. Ülesande lahendamisel kasutatakse valemi tehtepuud ja selle omadusi, et avaldist teisendada.

Joonisel 9 on näidatud ühe valemi tehtepuu. Puu tipus on valemi peatehe, selle all tehte argumendid. Tehte argumendiks võib olla muutuja või osavalem uue peatehtega.

Eitusel saab olla ainult üks argument. Tehtepuul (vt. joonis 9) on ekvivalentsi parempoolses alamvalemis disjunktsioonil kolm argumenti, kuna antud osavalemis ei ole peatehe üheselt määratud.



Joonis 9. Näide valemist ja selle tehtepuust.

3.3.1 Automaatlahendaja kahe tehte kaudu avaldamise kohta

Selleks, et teisendada valemit kujule, kus saadud valemis on teheteks eitus ja üks kahekohalistest tehetest (disjunktsioon, konjunktsioon või implikatsioon), on loodud analoogsed automaatlahendajad. Lahendajad koosnevad kahest meetodist.

Esimese meetodi töö kirjeldus on järgmine.

1. Kontrollida, kas sisendiks antud valem on ülesandetüübile vastaval kujul. Kui see on nii, siis avatakse dialoogiaken teatega, et valem on õigel kujul.
2. Kutsuda välja rekursiivne meetod, millega valemit teisendatakse. Sisendiks tuleb anda teisendatav valem, millega meetod välja kutsuti.
3. Kontrollida, kas teisendamisel saadud valem on ülesandetüübile vastaval kujul. Kui see on nii, siis avatakse dialoogiaken juhisega, mis leiti rekursiivse meetodi poolt.

Teine meetod on rekursiivne ja saab sisendiks osavalemi O. Rekursiivse meetodi töö kirjeldus on järgmine.

1. Kontrollida, kas O on muutuja (mitte osavalem, millel on omakorda argumendid). Kui see on nii, siis tagastada O.
2. Kuna valemi peatehet ei saa alati üheselt määrata (vt. joonis 9), peab leidma kõik peatehteks sobivad tehted ja nende alamvalemid.
3. Kutsuda iga alamvalemi kohta rekursiivselt välja sama meetodit, andes sisendiks vastava alamvalemi. Kui meetod tagastab uue valemi, asendada vana alamvalem uue valemiga. Alamvalemeid saab olla mitu, aga peab olema vähemalt üks.

4. Pärast alamvalemite asendamist teisendada O vastavalt selle peatehtele ja ülesandetüübile. Teisenduse järel leitakse, milline juhis hiljem kuvada. Dialoogiakna avamisel näitab programm sellist juhist, mis leiti kõige esimesena.
5. Tagastada eelmises punktis tehtud teisendamise käigus saadud valem.

3.3.2 Automaatlahendaja täieliku disjunktiivse normaalkuju kohta

Selleks, et teisendada valemit täielikule disjunktiivsele normaalkujule (TDNK), kasutatakse algoritmi, mida õpetatakse üliõpilastele aines DME. Kuigi võib leiduda valemeid, mida on optimaalsem mõnel muul viisil lahendada, on lahendaja eesmärk anda kasutajale just selliseid juhiseid, mida on vaja selleks, et ta õpiks algoritmi järgi valemit täielikule disjunktiivsele normaalkujule teisendama.

Kuna TDNK-le teisendamise algoritm on mahukam, on seatud automaatlahendajale teatud piirangud. Juhul, kui valemis on erinimelisi muutujaid rohkem kui 5, ei hakka algoritm ülesannet lahendama, sest see võib liiga kaua aega võtta. Õpetamise eesmärgil antakse üliõpilastele lühemaid valemeid, milles ei ole rohkem kui 5 muutujat. Seepärast ei ole vajadust, et algoritm suuremaid valemeid teisendaks.

Lahendaja koosneb ühest kesksest meetodist, mis kutsub algoritmi iga lahendusetapi kohta välja eraldi meetodi. Lisaks sellele üritatakse valemit etappide vahel lihtsustada.

Sarnaselt automaatlahendajale, mis avaldab valemi kahe tehte kaudu, jätab TDNK algoritm meelde kõige esimese juhise, mida see tahab kasutajale anda.

Keskse meetodi töö kirjeldus on järgmine.

1. Uurida valemit ja leida, millise lahendusetapi juures kasutaja on. Jätta valemi teisendamisel vahele need etapid, mis juba tehtud on.
2. Kontrollida, kas valem on juba ülesandetüübile vastaval kujul. Kui see on nii, siis avatakse dialoogiaken teatega, et valem on õigel kujul.
3. Lihtsustada valemit. Seejärel eemaldada valemist implikatsioonid.
4. Lihtsustada valemit. Seejärel eemaldada valemist ekvivalentsid.
5. Lihtsustada valemit. Seejärel viia eitused vahetult lausemuutujate ette.
6. Lihtsustada valemit. Seejärel rakendada kõikidele disjunktsioone sisaldavatele konjunktsioonidele distributiivsusseadust.
7. Eemaldada konjunktsioonid, milles esineb mõni muutuja nii eitusega kui ilma.

8. Eemaldada konjunktsioonidest korduvad muutujad
9. Lisada konjunktsioonidele puuduvad muutujad
10. Panna konjunktsioonides muutujad tähestikulisse järjekorda
11. Eemaldada korduvad konjunktsioonid
12. Kontrollida, kas teisendamisel saadud valem on ülesandetüübile vastaval kujul. Kui see on nii, siis avatakse dialoogiaken juhisega, mis leiti valemi teisendamise käigus.

Valemi lihtsustamine enne sammude 3-6 tegemist on vajalik selleks, et lahenduskäik pikaks ei veniks. Pärast 6. sammu tehakse selliseid teisendusi, mille järel ei ole lihtsustamise väljakutsumine enam vajalik.

Valemi lihtsustamise meetodi kirjeldus on järgmine.

1. Eemaldada valemist üleliigsed sulud
2. Eemaldada valemist kahekordsed eitused
3. Eemaldada valemist üleliigsed konjunktsioonid ja disjunktsioonid
4. Kui ühega eelmistest sammudest on valemit lihtsustatud, lihtsustada uuesti, kuni enam lihtsustada ei saa.

Ülejäänud meetodid võib jagada kaheks: kahe- ja üheosalised.

Kaheosalised meetodid koosnevad kahest meetodist. Keskses meetodis on need tähistatud numbritega 3-6.

Esimene meetod pöördub teise meetodi poole kuni teisendused on tehtud.

Teine meetod läbib rekursiivselt etteantud valemi tehtepuu, kasutades lõppjärjestust, ja leiab osavalemi, mida on vaja teisendada. Leitud osavalemile rakendatakse vajalikku teisendamisreeglit ja tagastatakse reegli rakendamise tulemus. Vajadusel jäetakse meelde, millise juhise kasutajale andma peab. Teise meetodi välja kutsumisel rakendatakse reeglit ainult ühele osavalemile.

Üheosalised meetodid on sellised, mida saab lahendada ühe reegli rakendamisega. Keskses meetodis on need tähistatud numbritega 7-11. Need meetodid eeldavad, et valem on kujul, kus eitused on vahetult muutujate ees ning valem koosneb disjunktsioonidega ühendatud konjunktsioonidest.

Üheosalise meetodi kirjeldus on järgmine.

1. Leida kõik literaalide konjunktsioonid
2. Vastavalt lahendussammule, mida meetod tegema peab, rakendada õiget reeglit kõigile eelmises punktis leitud konjunktsioonidele.

Kokkuvõte

1987. aastal programmeeris H. Viira lausearvutuse valemite teisendamise õpiprogrammi esimese versiooni. Seda hakati Tartu ülikoolis kasutama lausearvutuse õpetamisel. Kuna 2003. aastaks oli programm vananud, valmistas V. Vaiksaar oma bakalaureusetöö raames Java programmeerimiskeeles uue programmi, millega saaks asendada H. Viira versiooni. Uuem versioon asendas vanema ja seda kasutatakse praegugi ainetes Diskreetse matemaatika elemendid ja Sissejuhatus matemaatilisse loogikasse. Antud bakalaureusetöö raames on täiendatud 2003. aastal valminud programmi ning lisatud sellele funktsionaalsust.

Õpiprogrammi põhiliseks täiendusteks on:

- automaatlahendaja, mis annab kasutajale juhiseid valemite teisendamiseks
- parem ülesannete lahenduskäigu informatsiooni käsitlemine
- programmi tõlkimise võimaluse tekitamine

Uue versiooni abiga on võimalik lausearvutuse valemite teisendamist kergemini õppida, sest automaatlahendaja annab kasutajale juhiseid ülesannete lahendamiseks. Võib arvata, et ülesandeid juhenditega lahendades jääb üliõpilastele lahendussammude õige järjekord paremini meelde.

Täiendatud programmis salvestatakse ülesannete lahendamise ajal detailselt iga lahendussamm. Kasutajad saavad eraldi aknas vaadata oma varem tehtud ülesannete lahenduste logisid ja nende kaudu oma vigadest õppida. Matemaatilise loogika kursuste juhendajatel on aga kergem analüüsida üliõpilaste töid ja aru saada, mille juures neil rohkem probleeme tekib.

Algebraic manipulation Assistant for Propositional Logic and Predicate Calculus: Improvements

Bachelor thesis

Sander Stroom

Summary

In 1987, H. Viira created a program for solving simple problems in propositional calculus. The program was used in the University of Tartu for more than 10 years. In 2003, V. Vaiksaar created a new program for algebraic manipulation of propositional formulae as his bachelor thesis. It is now being used at the University of Tartu in the mathematical logic courses. There are many flaws and shortcomings in the first version of the program. For instance, there is no functionality of automatically solving the propositional formulae. For this and other, less significant reasons, this program has to be improved. The updated version of the program is included in this bachelor thesis.

Chapter 1 describes the courses that teach propositional calculus and the types of tasks that students have to learn. Chapter 2 describes the previous version and points out the major flaws that need to be fixed. The main tasks of the thesis are also written out in chapter 2. Chapter 3 contains a list of the improvements and their descriptions. The implementation of the automatic task solver is described in more detail.

There are three important additions to the older version. The first one is an automatic task solver that would give hints to the user about which transformation should be performed next. The solver can solve 4 types of tasks that are being taught to the first-year-students. Another addition is a more detailed structure for the transformation log. This would help the students to learn from their mistakes and the instructor to understand how the students solve the tasks. Lastly, the ability to translate the program for other languages has been implemented. The user interface was slightly changed in the process.

Kirjandus

1. R. Prank, H. Viira (1991). Algebraic Manipulation Assistant for Propositional Logic. *Computerised Logic Teaching Bulletin*, St Andrews Univ, 4(1): 13-18.
http://www.cs.ut.ee/~prank/Publikatsioonid/Standrew.txt (10.05.2013)

2. V. Vaiksaar (2003). Lausearvutuse ja predikaatloogika valemite teisendamise õpiprogramm, bakalaureusetöö.

3. R. Prank, V. Vaiksaar (October 2003). Expression manipulation environment for exercises and assessment. *6th International Conference on Technology in Mathematics Teaching*, Volos-Greece, 342-348.
http://www.cs.ut.ee/~prank/Publikatsioonid/Prank-Vaiksaar.doc (10.05.2013)

4. R. Palm, R. Prank (2004). Sissejuhatus matemaatilisse loogikasse.

5. Lausearvutuse 2. nädala arvutipraktikum. Lausearvutuse 3. nädala arvutipraktikum. Moodle'i kursus MTAT.05.109 Diskreetse matemaatika elemendid.

6. Tõeväärtustabeli ülesanded. Valemite teisendamine. Normaalkujud. Moodle'i kursus MTAT.05.111 Sissejuhatus matemaatilisse loogikasse.

Lisad

Lisa 1. Tõlkefailide loomise juhend

Lause- ja predikaatloogika valemite teisendamise õpiprogramm
Fraaside tõlkimise juhend

Programmi tekste saab tõlkida teistesse keeltesse, kui luua uus tõlkefail ja see programmi abil avada. Seejärel asendab programm kõik esialgsed fraasid neile vastavate fraasidega tõlkefailist. Tõlkefailide laiendiks peab olema ".properties".

Selleks, et fraasid segamini ei läheks, kasutatakse fraaside ees neile vastavaid indekseid kujul "sXXX", kus "XXX" asemel on fraasile vastav arv.

Kõige lihtsam oleks alustada tõlkimist sellega, et teha koopia ühest olemasolevast tõlkefailist ja asendada olemasolevad fraasid vastavate tõlgetega.

Selleks, et programm oskaks tõlkefaili lugeda, on vaja, et tõlkefaili tekst oleks kindlal kujul. Igal real peab olema üks indeks ja talle vastav fraas. Indeksi ja fraasi vahel olgu ka võrdusmärk, mis on eraldatud tühikutega.

[Näide]
s001 = Reegli rakendamine
s002 = Lisada vajalikesse kohtadesse sulud:
[Näite lõpp]

Tõlkefailist fraase lugedes visatakse iga fraasi eest ja tagant ära kõik tühikud. Kui on kindlasti vaja, et uues fraasis oleks ka tühikud olemas, on vaja kasutada tühiku eest kaldkriipsu '\'.

[Näide]
s041 = Osavalemit\
s042 = \ ei saa reegli põhjal välja jätta!
[Näite lõpp]

Tõlkefail on ülesannete koostamise ja nende lahendamise programmile ühine. Kui tõlkida ära kõik fraasid, võib olla kindel, et mõlemat programmi tõlgitakse korrektselt.

Kui ülesandeid luuakse, siis nende kirjeldused salvestatakse String-muutujasse. Neid kirjeldusi ei tõlgita, sest ülesande looja võib kirjelduse ise kirjutada. Kui on vajadus luua ülesandeid, mille kirjeldus oleks teises keeles, võib ülesannete loomise programmis enne keele ära muuta ja ülesannetele pannakse kirjeldused automaatselt valitud keeles.

Tõlkefailis ei tohi olla üleliigseid indekseid ega fraase. Kõik indeksid peavad ka olemas olema. Vastasel juhul ei võta programm tõlkefaili vastu.

Kasutusjuhendit ei ole hetkel võimalik tõlkida, kuna selle tekst on väga pikk ja seda ei ole sobiv tõlkida sellise tõlkefailiga.

Selleks, et testida, kas programm tõlkefaili avab, on vaja see lihtsalt programmi kaudu avada.

Lisa 2. CD programmi, selle lähtekoodi ja tõlkefailide loomise juhendiga

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Sander Stroom

(sünnikuupäev: 25.02.1989)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Lause- ja predikaatarvutuse valemite teisendamise õpiprogrammi täiendamine,

mille juhendaja on dotsent Rein Prank,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **13.05.2013**