Vaatimusmäärittely

Ohjelmistotuotantoprojekti Todistustoimitin

Jaakko Nenonen, Teppo Kankaapää, Aki Nyrhinen, Visa Röyskö, Juha-Matti Tapio

14. helmikuuta 2003

Helsingin yliopisto
Tietojenkäsittelytieteen laitos

$Id: vaatimusmäärittely.tex,v 1.6 2003/02/14 00:27:46 jmtapio Exp $

Sisältö

Historia

Päiväys	Versio	Kuvaus	Tekijät
14.2.2003	0.9	Ehdotus hyväksyttäväksi	Kaikki

1 Johdanto

Tämä dokumentti on ohjelmistotuotantoprojekti-kurssin todistustoimitin-ohjelmiston vaatimusmäärittelydokumentti. Dokumentissa määritellään asiakkaan ohjelmistolta vaatimat ominaisuudet, jotka projektiryhmän on toteutettava sekä vapaaehtoisia ominaisuuksia, joita toteutetaan, jos on resursseja.

Viime syksynä luennoitiin laitoksella kurssi Proof Theory and Proof Search, jossa käsiteltiin luonnollisen logiikan tapaista formalismia, jonka etuina tähän nähden on se, että todistamisprosessi vaatii vähemmän intuitiivista päättelyä. Lauseen todistaminen tapahtuu ”käänteisessä järjestyksessä”, lähtien lopputuloksesta ja jakaen sitä yhä pienempiin osiin sääntöjä käyttäen kunnes päädytään aksioomiin tai erityisiin kohtiin, joissa ollaan päädytty ”identtisesti tosiin” lauseisiin, jolloin todistus on valmis.

Aiemmin kurssin käytössä on ollut vain tekstipohjaisia ohjelmia. Projektin tarkoituksena on tuottaa graafinen helppokäyttöinen ohjelma, jolla todistuksia voidaan laatia mukaillen sitä, millä tavalla todistukset kirjoitettaisiin paperille.

2 Yleiskuvaus

Projektin tarkoituksena on rakentaa kaavamuokkain joka toteuttaa ainakin kaksi kurssilla esitettyä sääntökokoelmaa, klassisen, koodinimeltään G3CP, ja intuitionistisen, G3IP, ilman niinsanottua cut-sääntöä ja kvanttoreita. Sääntökokoelmissa on listattu joukko tapoja joilla kaavaa voidaan muokata. Kuhunkin kaavaan voidaan soveltaa useampaa sääntöä, käyttäjän tehtävänä on valita mitä. Ohjelman tulee näyttää ruudulla kaava, josta klikkaamalla käyttäjä voi valita, mihin suuntaan todistus etenee seuraavaksi. Valinnan jälkeen ohjelma laskee todistuksen seuraavan askeleen, josta jälleen voidaan jatkaa todistamista.

Tärkeintä ohjelmassa (tietenkin sen lisäksi että se toimii oikein) on selkeä käyttöliittymä, koska ohjelmaa tullaan käyttämään opetuksen apuna kun kurssia luennoidaan jatkossa. Yhdellä hiiren näpäytyksellä voisi valita kaavasta konnektiivin, jota puretaan auki seuraavaksi. Koska todistuksesta tulee puumainen ja pidemmissä lauseissa varsin laaja, todistuksen näyttäminen ruudulla tulisi olla myös hyvin suunniteltu.

Ohjelmaan pitää sisällyttää myös undo- ja redo-mahdollisuus, koska todistaminen on silti, vaikka näitä sääntöjä käyttäisikin, eri todistuspolkujen kokeilemista. Käyttäjä voi peruuttaa todistusaskeliaan ja valita toisia, tai palata aikaisempaan todistukseen jos uusi tie näyttää huonommalta.

Ohjelman olisi mahdollistettava uusien sääntökokoelmien käyttäminen koko ohjelmaa uudelleen koodaamatta, mutta tässä suhteessa laajennettavuus ei kuitenkaan ole kovinkaan tärkeä. Sen sijaan cut-sääntö ja kvanttorit tulisi ottaa suunnittelussa huomioon niin, että näiden ominaisuuksien toteuttaminen olisi suhteellisen helppoa. Nämä säännöt vaativat käyttäjältä muutakin syötettä kuin säännön valintaa.

2.1 Ohjelmiston jakautuminen

Ohjelmisto jakautuu pääosin toiminnalliseen osaan, tietorakenneosaan ja käyttöliittymäosaan, jossa ensimmäisen tehtävänä on toteuttaa haluttu kalkyyli, toisen tehtävänä tallentaa lasketut lausekkeet johonkin helposti käsiteltävään muotoon ja toteuttaa undo-toiminto, ja viimeisen tehtävänä näyttää ne.

Tietorakenneosa koostuu pääasiassa puurakenteesta, johon on tallennettu jokainen lauseke johon käyttäjä on todistusta vienyt, niin että yksi puun solmu sisältää lausekkeen, sen isäsolmu on lauseke josta se on johdettu, ja lapsisolmuina ovat kaikki ne lausekkeet jotka käyttäjä on todistuksen aikana tästä lausekkeesta johtanut.

Käyttöliittymäosa jakautuu kaiken koossapitävään osaan, todistusikkunaan, navigointi-ikkunaan, tulostusikkunaan ja tallennus- ja latausikkunaan. Kukin ikkunakomponentti sisältää myös siihen ikkunaan liittyvän toiminnallisuuden: todistusikkunassa lähinnä kaavojen näyttäminen ja kohdistus sekä ruudun vieritys, melkein samoin navigointi-ikkunassa. Tulostusikkuna sisältää rutiinit, joiden avulla todistuspuu saadaan tulostettua, ja tallennus/latausikkuna tallentamiseen ja lataamiseen liittyvät rutiinit. Käyttöliittymä on kuvattu tarkemmin omassa kappaleessaan.

Toiminnallinen puoli voisi jakautua rajapintaan käyttöliittymän kanssa ja itse säännöt toteuttavaan osaan joka voi sitten olla tehty vaikka LISP-kieltä tai vastaavaa apuna käyttäen. Tässä vaiheessa näyttää kuitenkin siltä, että myös säännöt toteutetaan Javalla, koska laajennettavuus tässä suhteessa ei ole kovin tärkeää.

2.2 Tietosisältö

Järjestelmän pitää pitää muistissa todistettavana olevasta kaavasta kaikki todistuspuut, jotka käyttäjä on siitä todistuksen aikana luonut (paitsi, jos muistinpuutteen takia käyttäjä haluaa poistaa jotain haaroja muistista lopullisesti, mutta tämän tarpeellisuus tutkitaan suunnitteluvaiheessa). Käyttöliittymää varten pidetään muistissa myös, mikä puu käyttäjälle sillä hetkellä näytetään. Lisäksi muistetaan käytetty sääntökokoelma.

Jokainen puun solmu vastaa yhtä kaavaa, joita ohjelman on myös pystyttävä käsittelemään, joten ne on purettu esim. jäsennyspuiksi. Jos halutaan perinteinen undo-toiminto ohjelman vapaamman peruutusmahdollisuuden lisäksi (kuvaus Palvelut-kappaleessa), viimeksi käytetty sääntö (tai muutamia sääntöjä) pitää pitää muistissa. Tämän lisäksi ohjelma tarvitsee vain jotain käyttöliittymäspesifisiä tietoja, kuten ikkunoiden sijainnit, käytetty fontti jne.

Itse sääntökokoelmat voitaisiin lukea myös tietosisältöön kuuluviksi, vaikka ne koodataankin ohjelmaan sisälle, jos halutaan pitää kiinni ohjelman laajennettavuudesta sääntökokoelmien suhteen. Tällöin toteutuksesta tullee niin modulaarinen että säännöt voidaan lukea tietosisältöön kuuluviksi, vaikka niitä ei pääsisikään muuttamaan esim. kääntämättä ohjelmaa uudelleen. Tiukasti ottaen ne eivät siihen kuulu.

3 Käyttötapaukset

Ohjelmalla on periaatteessa kolme eri käyttäjätyyppiä, joiden käyttötapaukset ovat keskenään suurimmaksi osaksi samanlaiset. Nämä ovat opiskelija, kurssin luennoija ja laskuharjoitusten pitäjä. Satunnainen ohjelman kokeilija voidaan lukea opiskelija-ryhmään kuuluvaksi.

1. Opiskelija Matti Mainion mieltä on askarruttanut eräs G3IP-kalkyylillä esitetty looginen väite, jonka hän haluaa todistaa.

- Alkutilanne: Ohjelma on juuri käynnistetty, ruudulla näkyy vain valikko.
- Matti valitsee valikosta Uusi todistus... -valinnan, joka avaa näytölle kyselyikkunan.
- Matti valitsee ikkunasta G3IP-kalkyylin ja kirjoittaa siihen kaavan, jonka haluaa todistaa, ja painaa OK.
- Kaava ilmestyy ohjelman pääikkunaan, ja nyt Matti valitsee kaavasta konnektiivin, johon liittyvää sääntöä hän haluaa soveltaa kaavaan.
- Ohjelma laskee säännön perusteella uudet kaavat ja näyttää ne ruudulla. Nyt Matti voi toistaa edellistä askelta näihin uusiin kaavoihin, kunnes alkuperäinen kaava on todistettu.

2. Proof Theory And Proof Search kurssin laskuharjoitusten pitäjä Jussi Juonio esittelee lauseen todistamista videoprojektorin avulla harjoituksissa.

- Alkutilanne:Ohjelma on avoinna ja ruudulla näkyy jokin todistuspuu. Teksti näkyy kankaalla liian pienenä.
- Jussi avaa ohjelman valikosta asetusikkunan.
- Ikkunassa näkyy ohjelman käyttämä fontti ja sen koko. Jussi valitsee fontin koon sopivammaksi ja painaa OK.

3. Matti Mainio saa valmiiksi harjoitustehtävän G3CP-kokoelmaa käyttämällä. Seuraavaksi hän haluaa tehdä seuraavan harjoitustehtävän, johon käytetään G3IP-sääntökokoelmaa. Hän haluaa ehkä palata katsomaan aiemmin tekemäänsä harjoitusta.

- Alkutilanne:Ohjelma on avoinna ja sisältää todistuspuun.
- Mahdollinen askel: Matti valitsee valikosta "Sulje todistus-valinnan. Ruudulla näkyvä todistuspuu sulkeutuu ja vain valikon sisältävä ikkuna jää ruudulle ja vanha harjoitus poistuu muistista.
- Askeleet kuten käyttötapaus 1:ssä.
- Jos todistusta ei suljettu: Valikossa näkyy listassa kaikki avoinna olevat todistukset. Matti valitsee listasta vanhan todistuksen, jolloin hän pääsee katselemaan sitä.

4. Käyttäjä haluaa seuraavaksi siirtyä nopeasti todistuspuun toisessa päässä sijaitsevaan osaan.

- Alkutilanne:Ruudulla on todistuspuu avoinna. Myös navigointi-ikkuna on näkyvissä.
- Käyttäjä näkee navigointi-ikkunassa puun pienemmässä muodossa (ilman pitkiä kaavoja) ja näkee siitä heti, mihin hän haluaa siirtyä. Käyttäjä klikkaa sopivaa solmua, ja ohjelma kohdistaa pääikkunan niin, että solmua vastaava kaava on keskellä.

5. Matti Mainio haluaa ihan kokeilumielessä testata tehtävän todistuspuuhun uutta haaraa, ja palauttaa tämän jälkeen puun alkuperäiseen tilaan.

- Alkutilanne:Ruudulla on todistuspuu avoinna, ja kaava, johon sovelletun säännön Matti haluaa peruuttaa, on ruudulla.
- Matti valitsee kaavasta toisen konnektiivin kuin oli alun perin valinnut. Kaava ei ole puun lehti (koska todistusta oli tästä jatkettu). Aiempi alipuu häviää ruudulta, ja sen tilalla nähdään uusi alipuu, joka on saatu soveltamalla nyt valittua sääntöä.
- Matti todistaa uutta haaraa ja katsoo mitä siitä seuraa.
- Matti valitsee kaavasta, joka ensimmäisessä askeleessa valittiin, uudelleen säännön, jonka oli alun perin ennen alkutilannetta valinnut. Vanha alipuun tila on muistissa, ohjelma piilottaa uuden alipuun ruudulta ja näyttää sen sijalla vanhan alipuun.

6. Matti Mainio on jo useamman tunnin ajan yrittänyt tehdä todistusta monimutkaiselle lausekkeelle. Lopulta hän toteaa olevansa väärillä jäljillä ja haluaa palata kokeilemaan sitä todistusreittiä mitä yritti jo tuntia aiemmin.

7. Kello on jo 3.00 yöllä ja Matti Mainio haluaa jatkaa todistusta seuraavana päivänä laitoksella.

- Alkutilanne:Ohjelmassa on todistus avoinna.
- Matti valitsee valikosta "Tallenna-valinnan.
- Tallennusikkuna aukeaa ruudulle, ja Matti kirjoittaa ikkunaan tiedostolle haluamansa nimen ja valitsee OK. Jos levyllä ei ole tilaa, tai muu virhe tapahtuu, ohjelma kertoo virheestä ja palaa ikkunaan. Ikkunasta voi silloin poistua Peruuta-napilla, jolloin tiedostoa ei tallenneta.

8. Matti Mainio jatkaa todistusta laitoksella siitä, mihin se yöllä jäi.

- Alkutilanne:Ohjelma on avoinna. Jokin toinen todistus saa olla myös avoinna.
- Matti valitsee valikosta "Avaa-valinnan.
- Ohjelma näyttää ruudulla listan aiemmin auki olleista tiedostoista sekä hakemistolistauksen ja kentän, johon voi kirjoittaa tiedostonimen. Matti valitsee jotain näistä käyttämällä haluamansa tiedoston.
- Todistus aukeaa ruudulle ja aiemmin auki ollut todistus menee taustalle (mutta sen saa myöhemmin palautettua esille).

9. Viimeinkin Matti Mainio saa vaikean todistuksen valmiiksi. Hän haluaa lähettää sen sähköpostilla kurssikaverilleen. Lisäksi Matti haluaa tulostaa todistuspuun iltapäivän laskareihin. Todistuspuu on niin iso, että Matti haluaa tulostaa sen kolmessa osassa.

- Alkutilanne:Ohjelmassa on todistus avoinna.

- Matti avaa valikosta "Vie/Tulosta-valinnan.
- Ruudulle aukeaa tulostusikkuna, josta Matti valitsee sen, haluaako hän tulostuksen ASCII-muotoisena vai jonain joka sopii tulostettavaksi paremmin. Aluksi Matti valitsee ASCII-valinnan ja kirjoittaa kenttään haluamansa nimen tiedostolle, johon tulostus tehdään. Painettuaan "Vie-nappia Matti voi avata tiedoston sähköpostiohjelmaan. Koko puu tulostetaan. Tulostusikkuna ei sulkeudu.
- Matti valitsee ikkunasta LaTeX-tulostuksen ja kirjoittaa kenttään toisen tiedostonimen. Hän valitsee lisäksi ikkunassa näkyvästä navigointipuun näköisestä ruudusta haluamansa puun osan tulostettavaksi. Painettuaan "Vie-nappia Matti voi kopioida syntyneen tiedoston sisällön johonkin LaTeX-dokumenttipohjaan, tulkata tämän ja tulostaa syntyneen .ps-tiedoston tulostimella.
- Matti toistaa edellisen askeleen, mutta valitsee toisen puun osan, ja sen jälkeen kolmannen.

4 Palvelut

Yleiskuvaus-kappaleessa on kerrottu ohjelman tarjoamat palvelut pääpiirteisesti. Tässä kappaleessa palvelut määritellään tarkemmin.

4.1 Todistaminen

Todistus on sääntöjen mukaan muodostettu puu. Puun juurena on käyttäjän syöttämä kaava. Käyttäjä syöttää kaavan ohjelmalle erillisessä dialogi-ikkunassa käyttäen sopivaa merkintätapaa, koska matemaattisia symboleita ei ole näppäimistöllä. Tämä suunnitellaan myöhemmin.

Yksi kaava muodostuu joukosta lausekkeita, jotka ovat predikaattilogiikan mukaan kirjoitettuja ja sijaitsevat "nuolen" vasemmalla puolella, ja toisesta joukosta lausekkeita, jotka ovat sen oikealla puolella. Todistus alkaa siten, että todistettava lauseke on yksin nuolen oikealla puolella ja vasen puoli on tyhjä. Säännöt muokkaavat näitä vasenta ja oikeaa kokoelmaa.

Kukin sääntö kohdistuu ainakin tässä ohjelmassa tiettyyn konnektiiviin yhdessä kaavan lausekkeista, lukuunottamatta muutamaa erikoissääntöä. Näyttää siltä, että on olemassa myös sääntöjä, jotka kohdistuvat useampaan konnektiiviin, mutta ohjelmamme ei toteuta näitä sääntöjä. Näiden sääntöjen löytäminen voidaan mahdollisesti (ohjelman laajennusversioissa) tehdä automaattisesti, eli kun kaavassa havaitaan sopiva useamman konnektiivin sääntö, käytetään tätä sääntöä tavallisen säännön sijaan.

Erikoissäännöt, joita ei tarvitse erikseen valita, ovat ne säännöt, jotka kertovat, koska todistus on päättynyt. Esimerkiksi jos sama predikaatti on kaavassa nuolen vasemmalla ja oikealla puolella, tähän kaavaan päättyvä puun haara on osoitettu todeksi.

Kuhunkin kaavaan voidaan soveltaa yhtä tai useampaa sääntöä. Käyttäjä valitsee sovellettavan säännön hiirellä klikkaamalla. Tämän säännön perusteella kaavasta johdetaan uusi kaava, joka näytetään edellisen päällä todistuksen "kirjoitusasua" mukaillen. Juuri on siis alimpana, ja muut kaavat tulevat sen päälle, ja ne ovat viivalla erotettu toisistaan. Sääntö voi tuottaa joko yhden uuden lauseen, joka näytetään alkuperäisen kaavan päällä, tai kaksi, jolloin ne muodostavat puuhun haaran ja ne näytetään vierekkäin.

Käyttäjä voi valita myös konnektiivin, joka on kaavassa syvemmällä sulkujen sisällä, jolloin ohjelma pystyy laskemaan yksikäsitteisesti sovellettavat säännöt ja käyttämään näitä perä jälkeen. Tässä ei tarvita älykästä toimintaa, koska seuraavaksi sovellettava sääntö nähdään suoraan.

Jos kaavassa on useita peräkkäisiä ja- tai tai-konnektiiveja, näitä ei tarvitse välttämättä esittää formaalisti oikein, eli niin, että tuloksena on suuri määrä sisäkkäisiä sulkulausekkeita. Kaavaa purettaessa ohjelman on kuitenkin otettava tämä huomioon joko niin, että sulkujen paikka määrätään silloin kun käyttäjä valitsee näistä konnektiiveista jonkin, jolloin sen vasemmalla puolella olevat esim. tai:lla yhdistetyt lausekkeet sulutetaan yhteen, ja vastaavasti oikealla puolella olevat. Esim. lauseke $(A \vee B \vee C \vee D)$ hajoaa lausekkeeksi $((A \vee B) \vee (C \vee D))$, jos käyttäjä on klikannut keskimmäistä tai-merkkiä, ja tähän kaavaan sovelletaan sääntöä. Toinen vaihtoehto on kiinnittää jokin tietty sulkujärjestys (esim. $(A \vee B \vee C) = (A \vee (B \vee C))$), joka dokumentoidaan, mutta kaavat näytetään ilman ylimääräisiä sulkeita.

Todistaminen päättyy, kun jokainen puun lehti on kaava, jossa voidaan soveltaa sääntöä jonka mukaan se puun haara on tosi. Tällöin myös alkuperäinen lauseke, jonka käyttäjä syötti puun juureksi, on tosi.

4.2 Navigointi

Todistuspuista tulee helposti laajoja. Tämän takia käyttäjän pitäisi saada helposti katsottua todistusta myös kokonaisuutena. Suuremmat todistukset eivät mahdu tietokoneen ruudulle, jos ne kirjoitetaan auki kirjoitusasussaan, kuten edellisessä kappaleessa on kuvattu. Myös tätä pääikkunaa on pystyttävä vierittämään, niin että todellinen kirjoitusalue johon puuta voidaan rakentaa on periaatteessa ääretön.

Navigointipuulla, joka on esitetty tarkemmin käyttöliittymäkappaleessa, saavutetaan tämä. Puusta nähdään koko todistus yhdellä silmäyksellä ja solmuihin voidaan käyttää myös jonkinlaista värikoodausta, jos tarvitaan.

4.3 Peruutus ja palautus (Undo/Redo)

Todistusta pitää pystyä muokkaamaan myös niin, että vanhoja todistusaskelia perutaan ja niiden tilalle valitaan uusia. Vastaavasti näitä perumisia pitää pystyä kumoamaan, eli palauttamaan ennen perumisia puussa vallinnut tilanne. Syy siihen, miksi tällaiset palvelut tarvitaan on siinä, että todistuksen laatiminen on yritystä ja erehdystä, ja joskus hyvältä näyttänyt säännön valinta saattaa tuottaa odotettua suuremman puun. Tällöin käyttäjä saattaa haluta kokeilla jotain muuta sääntöä tähän. Mutta jos uudella säännöllä puusta tulee vielä laajempi kuin aikaisemmin, vanha tilanne pitää voida palauttaa. Aiemmin valittu sääntö on ehkä kuitenkin parempi ja todistus on aidosti laaja.

Peruutus ja palautus saadaan mahdollisimman hienosäätöiseksi käyttöliittymäkappaleessa kuvatulla ratkaisulla, jossa kaikki ikinä todistuksen aikana käyttäjän valitsemat säännöt pidetään ohjelman muistissa, jolloin mielivaltainen aiemmin vallinnut tilanne voidaan palauttaa.

4.4 Tallennus ja lataus

Ohjelman on osattava tallentaa tehdyt todistuspuut tiedostoon ja ladata ne takaisin. Tiedoston muoto on projektiryhmän vapaasti suunniteltavissa. Koko tehty puu (mukaan lukien vaihtoehtoiset ratkaisuvaihtoehdot, ks. Undo-kappale) on tallennettava, ja lisäksi pitää tallentaa tieto siitä, mikä näistä vaihtoehdoista on tallennushetkellä näkyvissä, ja ehkä myös ikkunoiden sijaintitietoja ja tieto siitä, mikä osa puusta näkyy ruudulla (optio). Jos sijaintitietoja ei tallenneta, puuta ladattaessa ikkunoiden sijainnit ruudulla eivät muutu, sisällöt vain, ja lataamisen jälkeen pääikkunan keskitys on puun juuressa.

4.5 Tulostus

Todistuspuut pitää voida tulostaa kirjoittimelle. Jokin mielivaltainen todistuspuun alipuu pitää voida tulostaa yhdellä printtauskäskyllä, mutta puun erillisiä kohtia, jotka eivät liity toisiinsa polulla, ei tarvitse. Käyttäjän vastuulla on tässä tapauksessa se, että hänen valitsemansa puun osa mahtuu paperille.

Tulostus kannattaa toteuttaa esimerkiksi niin että ohjelma tuottaa puun LaTeX-koodina käyttäen proof.sty -makropakkauksen makroja, joiden avulla voidaan tulostaa hyvin ladottuja todistuspuita. Tämä koodi kopioidaan LaTeX-tiedostoon, joko automaattisesti (optio) tai käyttäjän käsin tekemänä. Tiedosto tulkataan ja tulostetaan (jälleen automaatio optiona). Ennen tulostusta käyttäjä voi sopivilla ohjelmilla (esim. xdvi) tarkistaa, että tuloste mahtuu yhdelle paperille, jos hän näin haluaa.

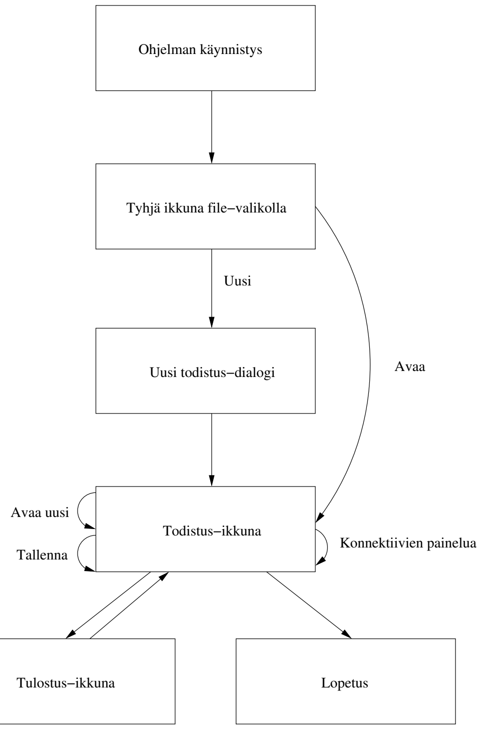
Ohjelman riippumattomuus tästä nimenomaisesta makropakkauksesta pitää voida varmistaa joko siten suunnitellulla tulostusrutiinilla, että sen voi helposti muuttaa käyttämään jotain toista makropakkausta, tai ASCII-tulostuksella, jolloin todistuspuu tulostetaan tavallisilla merkeillä. Tästä on myös se etu että todistus voidaan liittää esim. sähköpostiin.

5 Käyttöliittymä

Käyttöliittymä on ylivoimaisesti projektin tärkein yksittäinen komponentti, joten sen toiminnallisuuden tarkempi hahmottaminen koettiin tarpeelliseksi jo vaatimusmäärittelyn yhteydessä myöhempien yllätyksien riskin pienentämiseksi. Kuva 5 esittää yleisen käyttötapauksen tilasiirtymäkaaviona.

Ensimmäiseksi käyttäjä käynnistää ohjelman jollain tavalla, joka riippuu täydellisesti hänen käyttämästään käyttöjärjestelmästä. Ohjelman käynnistyttyä näkymä on tyhjä ruutu, ainoastaan ylävalikko tarjoaa toimintoja.

Tiedostovalikosta (file) käyttäjä voi valita joko avaa (open), tai uusi (new) joista jälkimmäinen käynnistää kalkyylin valinta ja kaavansyöttödialogin. Avaa-komento käynnistää



Kuva 1: Käyttöliittymän toimintakaavio

tiedostonvalintadialogin, josta käyttäjä voi valita haluamansa tiedoston hiirellä. Tallennusmuoto sisältää tiedon siitä, millä kalkyylillä todistus on tehty.

Kun todistus on syötetty tai luotu, pääikkunaan avautuu kaksi osaa, navigointi-ikkuna, joka on uutta todistusta aloitettaessa tyhjä, ja pääikkuna, jossa on näkymä todistuspuusta suunnilleen tulostusasussaan. Todistuksessa edetään klikkailemalla kaavojen konnektiiveja, jolloin ohjelma tuottaa automaattisesti kalkyyliin sopivan seuraavan vaiheen. Navigointipuussa solmu vastaa yksittäistä klikkausta, ja kyseisen solmun klikkaaminen asemoi pääikkunan siten että klikattu konnektiivi on ruudun keskellä.

Kun käyttäjä on edennyt todistamisessa niin pitkälle, että haluaa tulostaa todistuksen, tai osan siitä, hän valitsee tiedostovalikosta tulosta-komennon, joka käynnistää tulostusdialogin. Tulostusdialogissa voi valita tulostusmuodon (ASCII tai LaTeX) sekä tulostiedoston nimen. Suoraan tulostimelle tulostaminen ei tule olemaan mahdollista kaikissa (lue: minun) ympäristöissä, vaan käyttäjän on itse käynnistettävä LaTeXkääntämään todistus tulostimen ymmärtämään muotoon, ja sitten tulostettava tulostiedosto.

Todistuksen tallentamiseen on kaksi vaihtoehtoa, tallenna (save) ja tallenna nimellä (save as), joista toinen pyytää tallennettavalle tiedostolle nimen, ja toinen tallentaa nimellä, jolla todistus ladattiin, tai oltiin joskus aiemmin tallennettu.

5.1 Pääikkuna

Varsinainen todistuksen eteneminen tapahtuu pääikkunassa valitsemalla yksittäisiä kaavan konnektiiveja hiiren vasemmalla napilla klikaten. Konnektiivin klikkaaminen toimii listan läpikäynnin tavoin siten, että jos konnektiivin valinta ei määrää yksikäsitteisesti, miten todistus jatkuu, uusi klikkaus saman konnektiivin kohdalla tarjoaa seuraavaa vaihtoehtoa.

Kun konnektiivia on klikattu, pääikkunaan piirtyy, miten todistus jatkuu kun kyseinen valinta on tehty, ja valittu konnektiivi lihavoidaan. Jos huomataan, että valittiin väärin, peruminen tapahtuu yksinkertaisesti klikkaamalla jotain toista konnektiivia. Jos tämäkään valinta ei tyydytä, voidaan palata takaisin klikkaamalla edellistä konnektiivia, jolloin kyseisen konnektiivin alle mahdollisesti rakennettu todistuspuu piirretään näkyviin kokonaisuudessaan. Ohjelma ei siis unohda mitään todistuksen haaraa, joka on tehty kyseisen todistussession aikana (ehkä kuitenkin pitäisi tarjota toiminto, jolla kaikki muut paitsi näkyvä todistus unohdetaan?).

Kuvissa 5.1 - 5.1 esitetään kuvasarjana erään de Morganin kaavan todistuksen toinen puolisko. Pääikkuna on vasemmanpuoleinen, ja navigointipuu oikeanpuoleinen. Klikatut konnektiivit on lihavoitu.

$$\Rightarrow (\neg A \wedge \neg B) \rightarrow \neg(A \vee B)$$

Kuva 2: Aloitustilanne

$$\frac{(\neg A \wedge \neg B) \Rightarrow \neg(A \vee B)}{\Rightarrow (\neg A \wedge \neg B) \boldsymbol{\rightarrow} \neg(A \vee B)}\, R\rightarrow \qquad\qquad R\rightarrow$$

Kuva 3: Oikeaa implikaatiota klikattu

$$\frac{\dfrac{\neg A, \neg B \Rightarrow \neg(A \vee B)}{(\neg A \boldsymbol{\wedge} \neg B) \Rightarrow \neg(A \vee B)}\, L\wedge}{\Rightarrow (\neg A \wedge \neg B) \boldsymbol{\rightarrow} \neg(A \vee B)}\, R\rightarrow \qquad\qquad \begin{array}{c} L\wedge \\ R\rightarrow \end{array}$$

Kuva 4: Vasenta konjunktiota klikattu

5.2 Navigointi-ikkuna

Kokonaisuuden hahmottaminen pelkästään pää-ikkunasta voi olla vaikeaa koska todistuspuu voi olla hyvinkin laaja. Lisäksi tällaisessa puussa navigointi ja haluamansa solmun

$$\dfrac{\dfrac{\dfrac{\neg B \Rightarrow \neg(A \lor B), A}{\neg A, \neg B \Rightarrow \neg(A \lor B)}\,L\neg}{(\neg A \land \neg B) \Rightarrow \neg(A \lor B)}\,L\land}{\Rightarrow (\neg A \land \neg B) \to \neg(A \lor B)}\,R\to \qquad\qquad \begin{array}{c} L\neg \\ | \\ L\land \\ | \\ R\to \end{array}$$

Kuva 5: Vasenta negaatiota klikattu

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg B, A \lor B \Rightarrow A}{\neg B \Rightarrow \neg(A \lor B), A}\,R\neg}{\neg A, \neg B \Rightarrow \neg(A \lor B)}\,L\neg}{(\neg A \land \neg B) \Rightarrow \neg(A \lor B)}\,L\land}{\Rightarrow (\neg A \land \neg B) \to \neg(A \lor B)}\,R\to \qquad\qquad \begin{array}{c} R\neg \\ | \\ L\neg \\ | \\ L\land \\ | \\ R\to \end{array}$$

Kuva 6: Oikeaa negaatiota klikattu

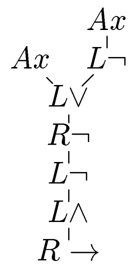
etsiminen voi olla työlästä. Myös käyttäjällä voi olla ongelmia tietää onko hän käynyt jo kaikki todistuksen haarat läpi. Tätä varten ohjelmassa on erillinen navigointipuu.

Navigointipuu on omassa ikkunassaan. Siinä todistuspuu on esitetty tiiviimmässä muodossa siten, että siinä on kaavojen tilalla ainoastaan aktivoitu konnektiivi. Kun käyttäjä klikkaa hiiren vasemmalla napilla haluamaansa solmua, keskitetään todistuspuu sen solmun kyseisen konnektiivin kohdalle. Navigointipuuhun solmu ilmestyy vasta kun haluttu konnektiivi aktivoidaan varsinaisessa todistuspuussa, eli kun todistuspuussa on vain juuri, navigointipuu on tyhjä. Suljetut haarat on merkitty eri värillä, jotta käyttäjä tietää että kyseinen solmu on lehti. Tästä käyttäjä näkee heti onko puussa osia joita hän ei ole

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg B, \boldsymbol{A} \Rightarrow \boldsymbol{A}}\,Ax \qquad \neg B, B \Rightarrow A}{\neg B, A \lor B \Rightarrow A}\,L\lor}{\neg B \Rightarrow \neg(A \lor B), A}\,R\neg}{\neg A, \neg B \Rightarrow \neg(A \lor B)}\,L\neg}{(\neg A \land \neg B) \Rightarrow \neg(A \lor B)}\,L\land}{\Rightarrow (\neg A \land \neg B) \to \neg(A \lor B)}\,R\to \qquad\qquad \begin{array}{c} Ax \\ \backslash \\ L\lor \\ | \\ R\neg \\ | \\ L\neg \\ | \\ L\land \\ | \\ R\to \end{array}$$

Kuva 7: Vasenta disjunktiota klikattu

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg B, \boldsymbol{A} \Rightarrow \boldsymbol{A}}\,Ax \qquad \dfrac{\dfrac{}{\boldsymbol{B} \Rightarrow A, \boldsymbol{B}}\,Ax}{\neg B, B \Rightarrow A}\,L\neg}{\neg B, A \vee B \Rightarrow A}\,L\vee}{\neg B \Rightarrow \neg(A \vee B), A}\,R\neg}{\neg A, \neg B \Rightarrow \neg(A \vee B)}\,L\neg}{(\neg A \wedge \neg B) \Rightarrow \neg(A \vee B)}\,L\wedge}{\Rightarrow (\neg A \wedge \neg B) \rightarrow \neg(A \vee B)}\,R \rightarrow$$



Kuva 8: Vasenta negaatiota klikattu

vielä käsitellyt.

Koska kaava voi olla suuri, sen todistuspuu ei välttämättä mahdu kokonaan edes pienennettynä, joten ikkunassa on scroll-barit sekä pituus että leveyssuunnassa.

5.3 Tulostus- ja vientidialogi

Todistuksen voi tulostaa tai viedä (*huono soomi*) jonkin toisen ohjelman tukemassa tiedostomuodossa erillisen dialogin avulla. Mahdollisia tiedostomuotoja ovat LaTeX(`proof.sty`), ASCII ja Embedded Postscript. Tuotoksen voi ohjata joko tulostimelle tai käyttäjän valitsemaan tiedostoon. Jälkimmäistä vaihtoehtoa tarvitaan erityisesti silloin kun käyttäjä haluaa liittää tuotetun todistuksen esimerkiksi LaTeX-dokumenttiinsa.

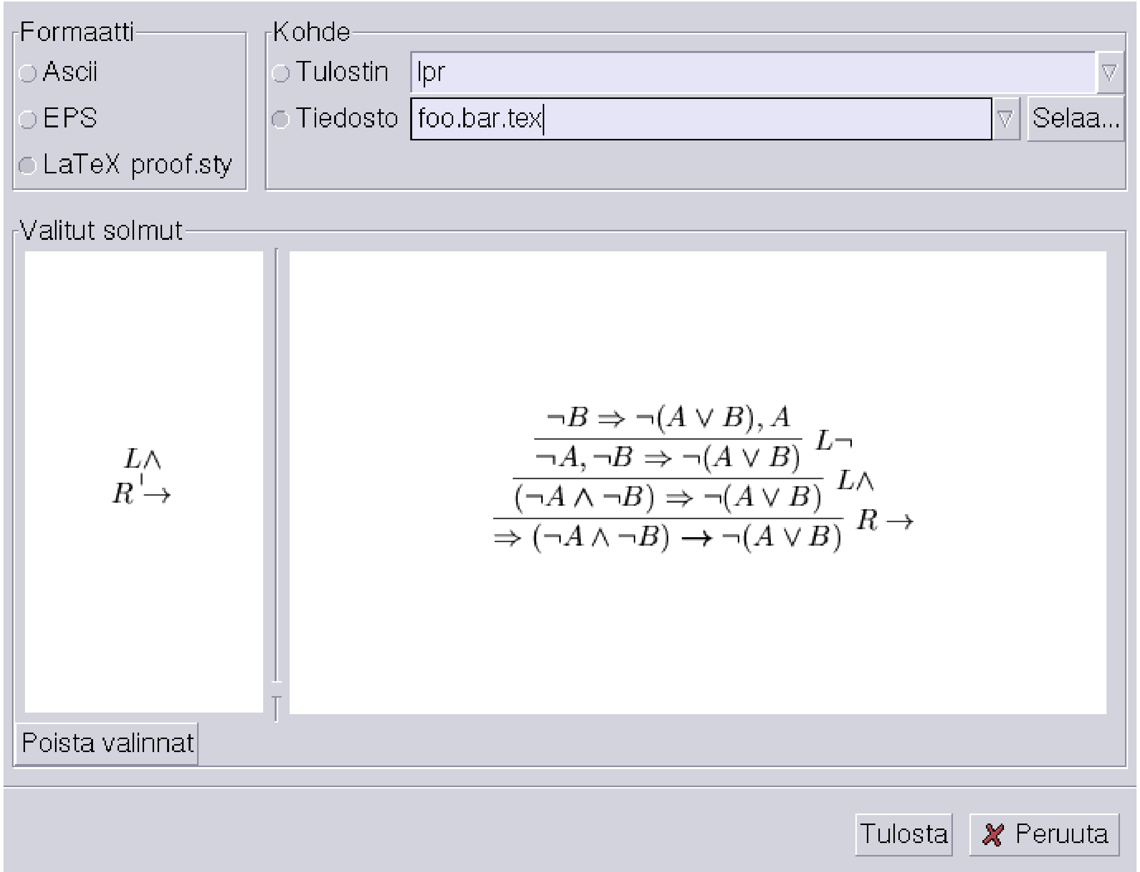
Jotta suurikokoisen todistuspuun voisi tulostaa monessa osassa, näyttää dialogi puun, josta käyttäjä voi valita tulostettavat/vietävät osat. Oletusarvoisesti koko puu tulostetaan. Mikäli käyttäjä valitsee jonkin lausekkeen puusta, tulostetaan lauseke ja siitä ylöspäin jatkuva puu. Mikäli käyttäjä valitsee toisenkin lausekkeen, tulostetaan alempi solmu sekä siitä ylöspäin jatkuvat puun osat ylemmän valitun lausekkeen korkeudelle asti.

Kuvassa 5.3 on esimerkki tulostus- ja vientidialogista.

6 Testaus

Ohjelmistolle laaditaan testaussuunnitelma, jossa määritellään tarkemmin, miten testaus tapahtuu. Testaus suoritetaan extreme programming (XP) -tyylisesti koodin kirjoittamisen yhteydessä niin, että ennen kuin moduulille kirjoitetaan itse toiminto jota halutaan, sille kirjoitetaan sitä ennen toimintoa testaava rutiini. Myös integrointitestaus suoritetaan näin. Kun kirjoitettu ohjelmakoodi laitetaan muiden saataville (commit), se tehdään aina vain kun sille tehty yksikkötestaus ja integrointitestaus menevät läpi.

Ohjelmiston testaustasot jakautuvat pääpiirteissään V-mallin mukaisesti: white-box-tyyppiseen moduulitestaukseen, sekä black-box-tyyppisiin integrointitestaukseen ja järjestelmätestaukseen (johon liitetään myös validointitestaus).



Kuva 9: Tulostus- ja vientidialogi

Järjestelmätestauksessa ei käytetä XP-testausta, vaan testaus tapahtuu tämän vaatimusdokumentin määrittelemien käyttötapausten ja muiden vaatimusten perusteella. Loppuvaiheessa, kun ohjelman käyttöliittymä alkaa hahmottua, asiakas voi suorittaa myös halutessaan validointitestausta, jonka perusteella ohjelman käyttöliittymään voidaan tehdä pieniä muutoksia (niin pieniä etteivät ne venytä projektin aikataulua), jos näitä ongelmia ei havaittu projektin aikaisemmissa vaiheissa.

7 Rajoitukset

Ohjelma toteuttaa kaksi sääntökokoelmaa: klassisen G3CP:n ja intuitionistisen G3IP:n. Kokoelmista rajataan pois ns. katkaisusääntö sekä predikaattilogiikka. Erityisesti näiden ominaisuuksien lisäämiseen myöhemmin varaudutaan ohjelmaa suunniteltaessa. Myös uusia sääntökokoelmia voisi ohjelmaan lisätä, mutta tähän ei varauduta, koska muuten ohjelman suunnittelu ja toteutus käy vaikeaksi.

Erilaiset muunnosmenetelmät formaalien todistusten ja luonnollisen päättelyn välillä rajataan myös pois ohjelman määrittelystä. Minkäänlaista tekoälyä, joka osaisi itenäisesti löytää todistuksia, ei toteuteta. Näiden ominaisuuksien lisääminen tulisi olla helppoa niin, ettei ohjelmaa jouduttaisi uudelleenkirjoittamaan paljoakaan.

Todistuspuun koolla ei ole rajoituksia ja se saa kasvaa periaatteessa kuinka suureksi tahansa. Todistuspuun järkevästä tulostuksesta joutuu käyttäjä kuitenkin vastaamaan itse. Suoritusympäristölle ohjelma ei saisi asettaa suuriakaan rajoituksia, koska ohjelmaa käytetään Proof Theory And Proof Search -kurssilla, ja sen tulisi toimia laitoksen koneilla sekä kurssin oppilaiden kotikoneilla. Ohjelman pitäisi toimia kaikilla nykyisillä Java-alustoilla, minkä vuoksi se toteutetaan Javan varhaisemmalla versiolla (esim 1.2).

Koska kyseessä on melko raskas Java-sovellus, ohjelma vaatinee jonkin verran konetehoa ja muistia. Muistinkäyttöä voidaan vähentää rajaamalla undo-redo-askelmien määrää sopivasti. Ohjelman käyttämiseen tarvitaan näppäimistö, hiiri sekä tulostin, mikäli haluaa käyttää tulostuspalvelua. Ohjelma toimii Linux- ja Windows-käyttöjärjestelmissä.