TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mark Dzotsenidze 222831IAIB
Markus Vrager 213079IAIB

Stringdiagrammi redaktori kasutajaliidese arendamine domeeni spetsiifiliste keelte kasutuseks

Bakalaureusetöö

Juhendaja: Paweł Maria Sobociński
PhD

Kaasjuhendaja: Anton Osvald Kuusk
BSc

Tallinn 2025

Autorideklaratsioon

Kinnitame, et oleme koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autorid: Mark Dzotsenidze ja Markus Vrager

06.06.2025

Annotatsioon

Antud töö eesmärgiks on edasi arendada string diagrammi redaktori kasutajaliidest. Peamiselt keskendutakse kasutajakogemuse parandamisele ja TikZ koodi genereerimise võimaluse loomisele. Samuti uuritakse graafilise ülevaate mõju koodi mõistmisele ja kirjutamisele. Rakendus on loodud kasutades Python'i programmeerimiskeelt ning graafilise kasutajaliidese loomiseks kasutatakse Tkinter'i teeki. Rakenduse eesmärgiks on lubada kasutajal mugavalt töötada string diagrammidega ning samuti lubada programmeerimist läbi rakenduse.

Töös kirjeldatakse arenduse planeerimise jaoks läbi viidud analüüsi teiste sarnaste rakenduste kohta, rakendusele lisatud funktsioonide integreerimist ning valideerimise tulemusi. Valideerimiseks viiakse läbi kasutajatestimine ja uuriti läbi rakenduse programmeerimist.

Rakenduse lähtekood on peale arendust lisatud avalikku GitHub'i hoidlasse: `https://github.com/madzot/ivaldi-diagram-editor`.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 6 peatükki, 17 joonist, 2 tabelit.

Abstract

GUI Development for a String Diagram Editor for the Use of Domain Specific Languages

The purpose of this thesis is to further develop the GUI of a string diagram editor. With the main focus on improving user experience, enabling TikZ generation from diagrams, and researching the effects of a graphical overview on code understanding as well as code writing. The further usage for the application is enabling the use of different DSLs through the application. The application is written in Python and uses Tkinter to create a graphical interface.

There are very few existing applications that are focused on string diagrams. This application is developed to be usable throughout different fields that require the use of string diagrams. It does not only contain visually creating diagrams, but is more focused on supporting the mathematical side of string diagrams.

This thesis will cover an analysis of 3 existing diagram editing applications as well as the MVP application provided to authors as a starting point. Give an overview of added functionalities with explanations as to why and how they have been implemented, as well as validation of done work.

For analysis, authors compared Cartographer, TikZiT, Diagrams.net (aka Draw.io) and the MVP application. Analysis was mainly focused on user experience, variety of usable components, and ability to work with string diagrams specifically. Based on the results of the analysis and goals of the thesis, authors added functionality to improve user experience and allow the generation of TikZ code from diagrams, among other functionalities. Along with this, the authors implemented a system for writing code in the application to conduct the research previously mentioned.

The validation of this project consisted of 3 parts. The first part focuses on user tests carried

out using both the MVP application as well as the final version of the application on people who had not used this application before to assess whether the user experience had been improved. The second part focuses on coding in the application to assess the benefits and drawbacks of coding through diagrams. In the final part, authors replicated string diagrams used in existing scientific papers to prove usability in a scientific context.

The developed application is available on GitHub at `https://github.com/madzot/ivaldi-diagram-editor`.

The thesis is in Estonian and contains 34 pages of text, 6 chapters, 17 figures, 2 tables.

Lühendite ja mõistete sõnastik

CI	Pidev integratsioon (*Continuous integration*)
ConTEXt	Üldotstarbeline dokumendiprotsessor (*General-purpose document processor*)
DSL	Domeeni spetsiifiline keel (*Domain Specific Language*)
Flowchart	Vooskeem(*Diagram representing workflow or process*)
GUI	Graafiline kasutajaliides (*Graphical User Interface*)
HTML	Hüperteksti märgistuskeel (*HyperText Markup Language*)
JSON	Andmevahetusvorming (*JavaScript Object Notation*)
LATEX	Dokumentide ettevalmistussüsteem (*Document preparation system*)
MVP	Minimaalne töötav toode (*Minimum Viable Product*)
PDF	Porditav dokumendivorming (*Portable Document Format*)
PGF	Graafiline formaat (*Portable Graphics Format*)
PNG	Porditav võrgugraafika (*Portable Network Graphics*)
Regex	Regulaaravaldis (*Regular Expression*)
SQL	Struktuurpäringukeel (*Structured Query Language*)
TEX	Tekstilao süsteem (*Typesetting program*)
TikZ	Vektorgraafika keel (*Language for producing vector graphics*)
UML	Unifitseeritud modelleerimiskeel (*Unified Modeling Language*)
XML	Laiendatav märgistuskeel (*Extensible Markup Language*)

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

String diagrammid on üha enam populaarsust koguv esitusviis, mis on lisaks kategooriateooriale viimastel aastatel kasutust leidnud nii arvutiteaduses, füüsikas kui ka lingvistikas [1]. Samas aga puuduvad head rakendused string diagrammidega töötamiseks. Olemasolevad rakendused on mõeldud kas diagrammide joonestamiseks, pakkumata string diagrammidele vajalikke piiranguid, või on vastupidiselt liiga piiravad, lubades rakenduse kasutamist ainult mingis kitsas valdkonnas.

Lõputöö raames edasi arendatud rakendus on mõeldud olema üldotstarbeline redaktor string diagrammidega töötamiseks. Lisaks string diagrammidega töötamisele oleks üks rakenduse kasutustest veel võimalus kasutada domeeni spetsiifilisi keeli (DSL) läbi rakenduse. Rakenduses diagrammide loomine toimub *drag and drop* põhimõttel, kasutaja saab luua nuppudelt objekte, mida soovib diagrammidesse sisestada ning hiirega neid liigutada. See eemaldab matemaatiliste teadmiste vajaduse diagrammide loomiseks.

Antud bakalaureuse töö käigus arendatakse edasi string diagrammi redaktori MVP (*Minimum Viable Product*) versiooni. Töö peamisteks eesmärkideks on parandada kasutajakogemust, lisades rakendusse funktsioone ja uuendades rakenduse visuaalset külge ning võimaldada rakenduses loodud diagrammide eksportimist TikZ (*Language for producing vector graphics*) koodi formaadis. Samuti uuritakse töö käigus graafilise ülevaate mõju programmeerimisele, keskendudes uurimisele, kas graafiline ülevaade teeb koodi mõistmise ja kirjutamise intuitiivsemaks.

Eesmärkide saavutamiseks analüüsitakse olemasolevaid rakendusi, mis lubavad string diagrammidega töötamist. Analüüsi tulemusel otsustatakse, mida peaks lisama lõputöö rakendusse, et parandada kasutajakogemust. Lisaks arendatakse originaalseid funktsioone rakendusele, et võimaldada kasutust erivaldkondades, pakkuda kasutajale head kasutajakogemust ja mugavat diagrammide loomist.

Peale arendusfaasi viiakse läbi kasutajatestimine inimeste hulgas, kes ei ole süvenenult

tutvunud string diagrammidega ning kes kasutavad rakendust esimest korda. Peale kasutajatestimise läbiviimist saab määrata edukust kasutajakogemuse parandamisel ning saadakse testijatelt tagasisidet tehtud töö kohta.

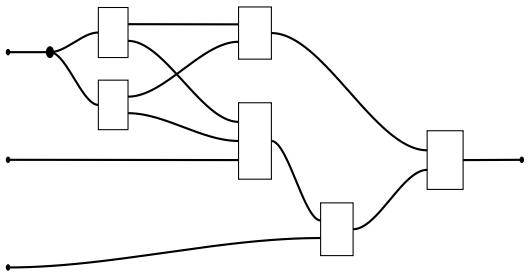
Lisaks viiakse peale arengufaasi läbi uuring seoses programmeerimisega rakenduse sees. Uuringu jaoks proovitakse lahendada läbi rakenduse programmeerimisülesanne. Lahendamise käigus hinnatakse selle praktilisust ja tõhusust, samuti saadakse teada, kas see on üldse võimalik.

Peale arendust lisati rakenduse lähtekood GitHub'i hoidlasse: `https://github.com/madzot/ivaldi-diagram-editor`

2 Taust

2.1 String diagrammid

String diagrammid pärinevad kategooriateooria abstraktsest matemaatilisest raamistikust, mis on kujutav notatsioon monoidse kategooria morfismide kirjeldamiseks. Praeguseks on string diagrammide kasutusvaldkonnad märkimisväärselt laienenud, ulatudes valdkondadesse nagu arvutiteadus, kvantfüüsika, lingvistika, tõenäosusteooria, masinõpe ja paljudesse muudesse valdkondadesse. [2]



Joonis 1. Näide string diagrammist.

Üks tähtis põhjus string diagrammide kasutamiseks on ressursitundlikkus [3]. String diagrammid lubavad kergelt jälgida, kuidas algoritmides ressursse kasutatakse, see aitab kokku hoida näiteks materjale või muid ressursse, mis omakorda aitab luua efektiivsemaid protsesse. Näiteks on suudetud kasutada string-diagrammidel põhinevaid ZX-diagramme kvantahelate lihtsustamiseks, mis aitab vähendada nii vajamineva riistvara hulka kui ka parandada seadme veakindlust [4].

String diagrammide atraktiivsus seisneb veel lisaks ka nende mitmekülgsuses, nad pakuvad keerulistele teaduslikele ideedele lihtsat visuaalset esitust ning samas võimaldavad ka matemaatilist käsitlust [2]. Visuaalne esitlus saab aidata pakkuda intuitiivsemat ja lihtsamat arusaama nii algoritmidest kui ka programmidest. Seda on ära kasutanud nii Tao Gu, kes oma töös *"Categorical Modelling of Logic Programming"* kasutas string diagramme loogiliste programmide struktuuri kujutamiseks kategooriateooria kontekstis, kui ka Dusko Pavlovic, kes raamatus *"Programs as Diagrams: From Categorical Computability to Computable Categories"* kasutas string diagramme programmide kujutamiseks, et pakkuda

lugejale intuitiivsemat lahendust arvutusprotsesside mõistmiseks [5] [6].

2.2 Kasutajakogemus

Olenemata sellest, et string diagrammid on intuitiivsed, ei pruugi sellest piisata, kui puuduvad rakendused string diagrammide loomiseks või rakenduste kasutamine on keerukas ja ebamugav. See võib viia olukorrani, kus kasutaja otsustab mitte kasutada string diagramme põhjusel, et nende koostamine on ebaratsionaalselt keerukas või aeganõudev. Seega on ka olulisel kohal rakenduse kasutajakogemus.

Kasutajakogemust kui terminit hakkas esimesena kasutama Donald Norman, kes juhtis tähelepanu asjaolule, et sõna kasutajaliides ja kasutatavus on liiga piiravad [7]. Donald Norman tõi välja, et kasutajamugavus hõlmab kõiki aspekte toote kasutamise juures, mitte ainult selle disaini ja funktsioone. Tänaseks on palju erinevaid disainimustreid ja reegleid välja töötatud, mis kõik üritavad aidata luua parimat võimalikku kasutajakogemust [8]. Neist enim tuntud on Jakob Nielsen'i 10 heuristikat, mis toovad välja suuremad teemad, mida jälgides on rakenduse edasiarendus lihtsam ja kasutajakogemus parem. Neist heuristikatest lähtutakse rakenduse arendamisel. Erilist tähelepanu pööratakse nii seitsmendale kui ka kaheksandale heuristikale, mis keskenduvad vastavalt paindlikusele ja tõhusale kasutusele ning esteetilisele ja minimalistlikule stiilile. [9]

2.3 Domeeni spetsiifilised keeled

Domeeni spetsiifilised keeled (DSL) on väikesed, tavaliselt deklareerivad keeled, mis on mõeldud mingi kindla valdkonna probleemi lahendamiseks [10]. Praeguseks on loodud DSL'e väga erinevate valdkondade jaoks, näiteks tekstitöötlus, robootika ja paljud teised, millest laialdaselt on tuntud näiteks SQL (*Structured Query Language*), HTML (*HyperText Markup Language*), LaTeX (*Document preparation system*) ja Regex (*Regular Expression*) [11]. DSL'id on hea viis, mingi kitsa osa võtmiseks programmeerimisest ja selle muutmiseks kergesti mõistetavaks, lihtsustades koodi kirjutamist ja muutmist, samal ajal vähendades vigade tekkimise võimalust. Lisaks sellele, et DSL'id aitavad kaasa programmeerimisele, on need ka kasulikud valdkonnaspetsialistide ja programmeerijate vahelises suhtluses. DSL'id võimaldavad esitada süsteemireeglid valdkonnaspetsialistidele tuttavas vormis, lubades neil lihtsamat osalemist nii süsteemi arenduses kui ka hiljem selle muutmises. [12]

Domeeni spetsiifiliste keelte loomisel on äärmiselt oluline läbi viia valdkonna analüüs, kus võib arendajal kasuks tulla string diagrammiline esitus [10]. Samuti on võimalik DSL'ide tööprotsesside kujutamine string diagrammide kujul, aidates sellega kaasa tööprotsesside mõistmisele ja dokumentatsiooni illustreerimisele.

2.4 Probleem ja eesmärgid

Praegu ei ole häid rakendusi string diagrammidega töötamiseks, küll on olemas rakendused, kus on võimalik joonestada string diagramme, kuid need pole mõeldud string diagrammide jaoks. Lisaks sellele on varem loodud string diagrammide rakendusi just mingisse kindlasse valdkonda, kuid neid ei ole võimalik kasutada väljaspool seda kindlat domeeni.

Antud töös arendatava rakenduse eesmärk on olla üldotstarbeline redaktor string diagrammidega töötamiseks, mis võimaldaks joonestada skeeme ning seejärel määrata redaktorile piiranguid ja reegleid vastavalt kasutaja enda soovidele või domeenile. Rakendus võimaldaks veel ka programmeerimist läbi string diagrammide.

Lõputööl on 3 suuremat eesmärki: uurida, kas graafiline ülevaade teeb koodi mõistmise ja kirjutamise intuitiivsemaks, parandada kasutajakogemust ja võimaldada rakenduses loodud diagramme eksportida TikZ formaadis, et neid saaks kasutada LaTeX failides.

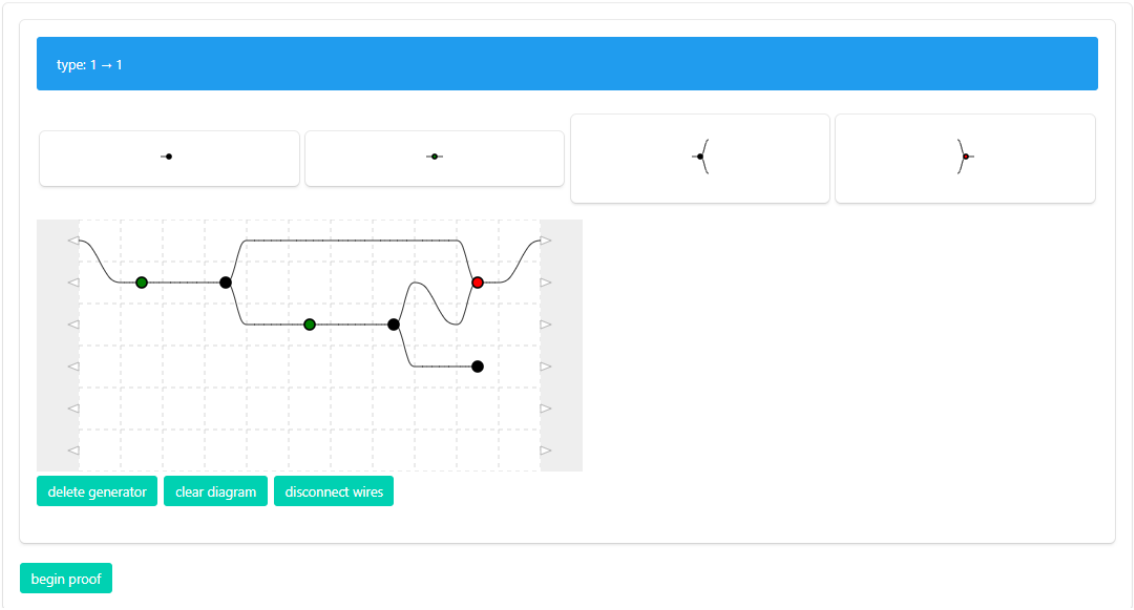
Lisaks eeltoodule on mõned väiksemad eesmärgid, nagu tagurpidi ühilduvuse säilitamine, pseudo-notatsiooni funktsionaalsuse säilitamine, saavutada paindlikkus, mis võimaldaks kirjeldada reegleid diagrammide joonestamiseks ja tõlgendamiseks.

3 Analüüs

Lõputöö käigus analüüsiti erinevaid rakendusi, millega on võimalik luua string diagramme. Kokku analüüsiti nelja rakendust, millest üks oli autoritele edasiarendamiseks antud MVP rakendus. Analüüsi tulemused on all välja toodud.

3.1 Cartographer

Cartographer on rakendus, mille abil saab muuta ja ümberkirjutada string diagramme sümmeetrilistes monoidsetes kategooriates. Tegemist on veebibrauseri rakendusega, mille avatud lähtekood on kättesaadav GitHub'ist [13]. Rakendus on mõeldud inimestele, kes soovivad joonestada ja ümberkirjutada string diagramme vastavalt enda loodud reeglitele. Rakenduses tuleb kasutajal luua kõik kasutatavad komponendid ja ka soovi korral reeglid, mille alusel diagramme ümberkirjutatakse või lihtsustatakse. [14]



Joonis 2. Cartographer kasutajaliides. [15]

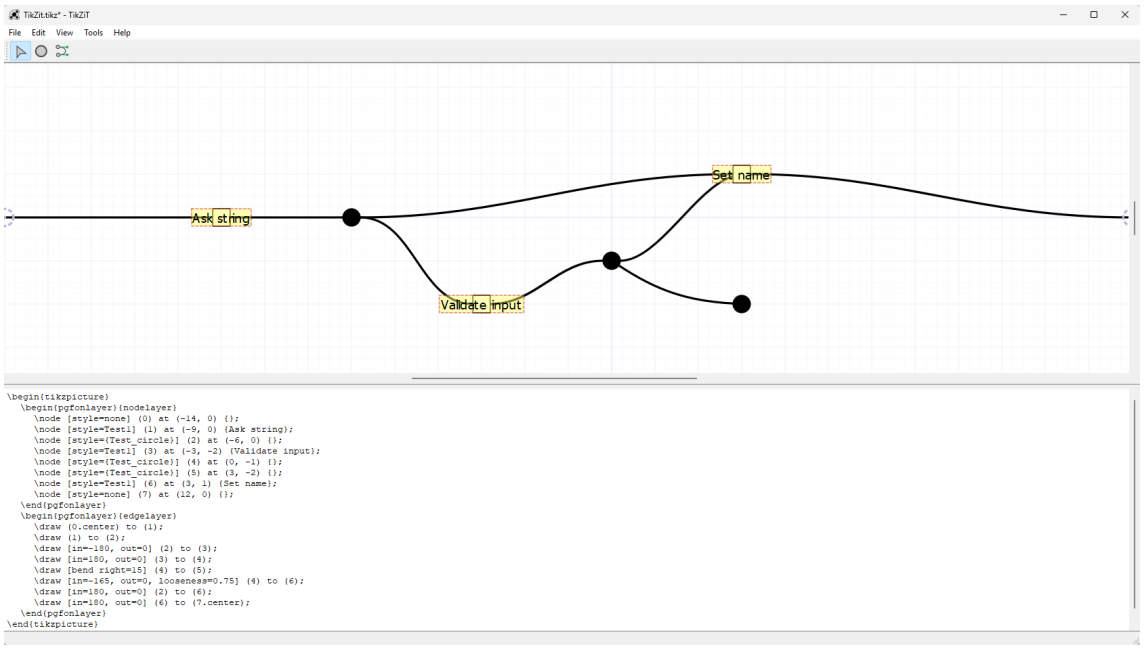
Antud rakenduse suurim tugevus on võimekus luua string diagramme ja reegleid, mille järgi diagramme lihtsustatakse või ümberkirjutatakse. See võimaldab kasutajal läbi viia

graafilisi tõestusi ja näidata, et reegleid järgides on võimalik diagrammi lihtsustamine, jättes diagrammi olemuse samaks. Rakendus on tasuta kättesaadav veebist koos rakenduse kasutamise õpetusega.

Rakenduses esineb palju probleeme. Nendest üks suuremaid on diagrammide jagamisvõimaluse puudumine, mis raskendab rakenduse kasutamist tiimitöös. Samuti puudub diagrammide salvestamise võimalus. Rakenduses puuduvad standardsed komponendid ja reeglid diagrammide joonestamiseks ja ümberkirjutamiseks, muutes diagrammide loomise ja tõestuste läbi viimise aeglasemaks. Võimalik on kasutada rakenduse õpetuses kasutatud komponente ja reegleid, kuid need on üpris piiravad ja ei sobi kasutamiseks enamikes diagrammides. Samuti puuduvad rakenduses paljud elementaarsed kasutajamugavust parandavad funktsioonid, nagu komponentide liigutamine pärast nende diagrammile lisamist, komponentide kopeerimine ja paljud teised. See muudab diagrammis muudatuste tegemise aeganõudvaks, eriti kui kasutaja soovib valmis diagrammi visuaali muuta. Rakenduses puudub alamdiagrammide loomise võimalus, mis raskendab mahukate diagrammidega töötamist. Cartographer'i kasutamine eeldab interneti või tehniliste oskuste olemasolu, et rakendust lokaalselt jooksutada.

3.2 TikZiT

TikZiT on kergekaaluline graafiline redaktor, mis on mõeldud TikZ jooniste visuaalseks loomiseks ja haldamiseks. TikZiT on suunatud inimestele, kes soovivad joonestada graafikuid, string diagramme või muid struktureeritud skeeme ilma vajaduseta kirjutada käsitsi TikZ-koodi. Tänu sellele kiirendab TikZiT diagrammide koostamist, säilitades samas TikZ-koodi võimekuse ja täpsuse. Tegemist on töölauarakendusega, mida saab kasutada Windows, Linux ja MacOS seadmetes. Rakenduse lähtekood on kättesaadav GitHub'ist [16]. [17]



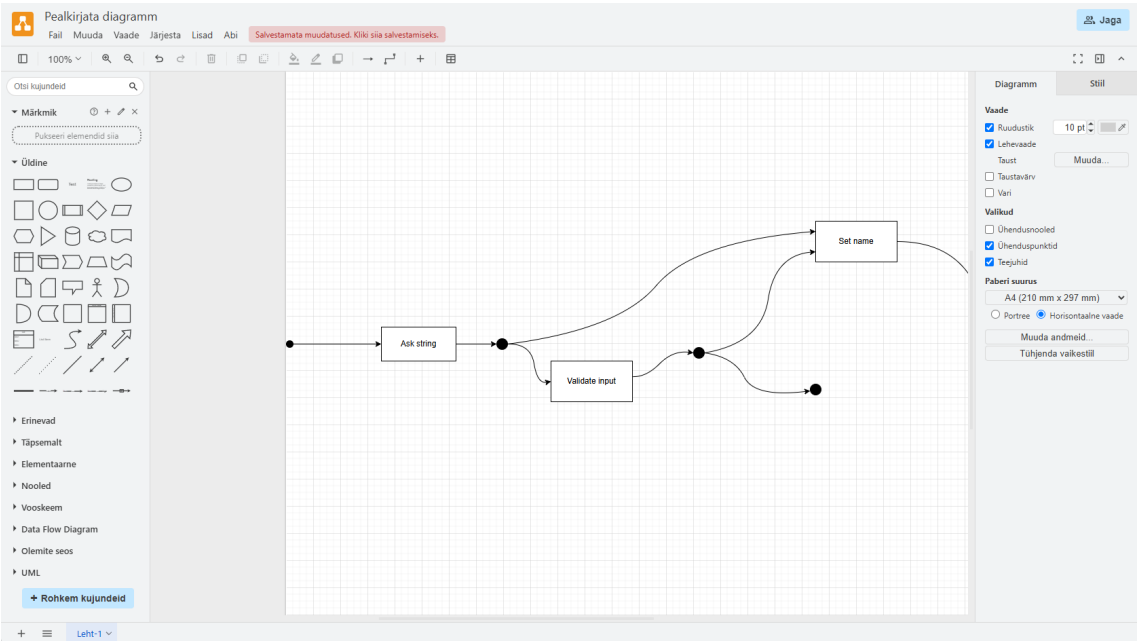
Joonis 3. TikZit kasutajaliides. [17]

Antud rakenduse peamine eelis teiste rakenduste ees, mida antud töö käigus analüüsiti, on võimekus diagramme eksportida TikZ-koodi formaadis, mis võimaldab rakenduses loodud diagrammide kasutamist LaTeX dokumentides. Rakendus on mõeldud string diagrammidega töötamiseks. Nende asjade koosmõjul on rakendust hea kasutada antud valdkonna teadustööde graafikute loomiseks. Rakenduses on võimalik määrata korduvkasutatavaid stiile, mis muudab mitme diagrammi loomise mugavamaks ja tagab, et diagrammide stiil on ühtne. Samuti põhinevad diagrammid vektorandmetel, mis muudab diagrammid skaleeritavaks ja lubab luua ka põhjalikke diagramme. Rakendus on tasuta allalaaditav veebist, kus on välja toodud ka rakenduse kasutamise õpetus. Rakenduse kasutamine ei eelda interneti olemasolu.

Rakenduse üks suurimaid puudusi on väike valik standardseid kujundeid, ikoone ja visuaalseid elemente, mis oluliselt piirab kasutaja võimalusi diagrammide visualiseerimisel, lubades kasutada ainult ruute või ringe komponentide kujutamiseks diagrammil. Samuti puudub rakenduses funktsionaalsus diagrammide ümberkirjutamiseks või lihtsustamiseks. Seega saab rakendust kasutada diagrammide visualiseerimiseks ja TikZ-koodi genereerimiseks, aga mitte formaalseks järeldamiseks ega diagrammide algebralise struktuuri analüüsimiseks. Rakenduses puudub alamdiagrammide loomise võimalus, mis raskendab mahukate diagrammidega töötamist.

3.3 Diagrams.net

Diagrams.net (endise nimega draw.io) on tarkvararakendus, mis võimaldab kasutajatel luua erinevat tüüpi diagramme. Tegemist on platvormiülese lahendusega, mida saab kasutada nii veebibrauseris kui ka töölauarakendusena. Selle paindlikkus ja kasutuslihtsus muudavad selle sobivaks nii teaduslikus kui ka professionaalses kontekstis kasutamiseks. Diagrams.net toetab laia valikut diagrammide tüüpe, sealhulgas: vooskeeme (*flowchart*), UML-diagramme (*Unified Modeling Language*), võrguarhitektuuri skeeme, andmebaasi relatsioone jne. [18]



Joonis 4. Diagram.net kasutajaliides. [19]

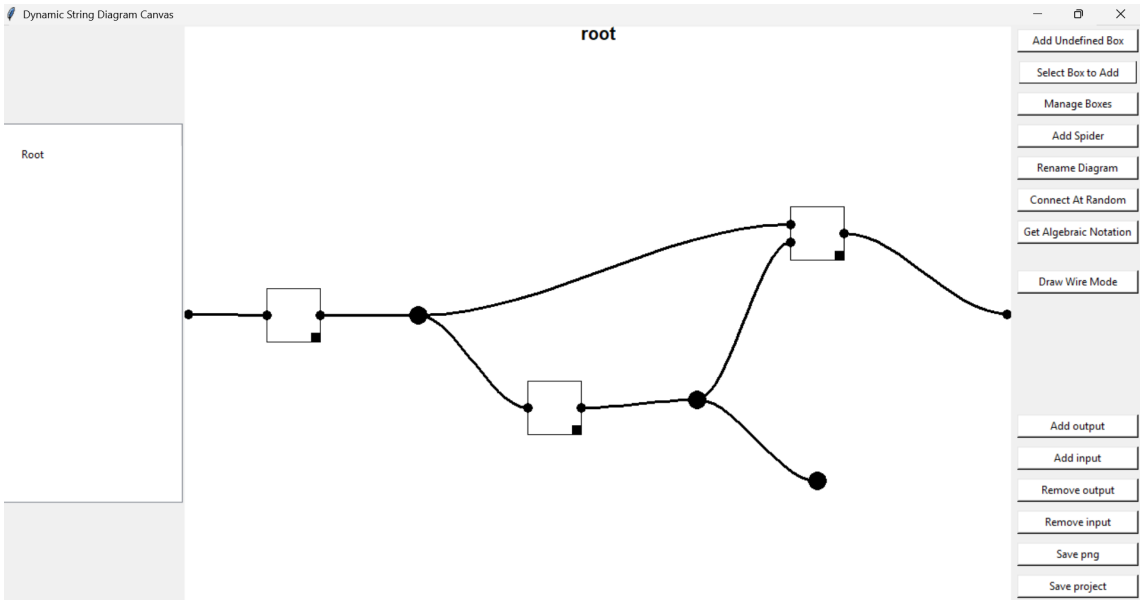
Rakenduse kasutajaliides põhineb lohistamise põhimõttel ning võimaldab kiiret ligipääsu erinevatele eeldefineeritud kujunditele, sümbolitele ja visuaalsetele mallidele. Diagramme saab muuta visuaalselt, kasutades kohandatavaid värve, stiile ja paigutusi. Rakendus pakub mitmeid eksportimise formaate, sealhulgas PNG (*Portable Network Graphics*), PDF (*Portable Document Format*), HTML ja XML (*Extensible Markup Language*). Samuti on võimalik rakendust integreerida erinevate pilveteenustega (nt Google Drive, OneDrive, GitHub), mis lihtsustavad failide jagamist ja koostööd. Sobib kasutamiseks nii individuaalselt kui ka meeskondades. Lisaks on olemas pistikprogrammid ja rakendusevälised tööriistad, mis täiustavad rakenduse funktsionaalsust. Rakendus lubab koostada alamdiagramme. Alamdiagramm tuleb luua teisele lehele ja siis peadiagrammile lisada komponent, mis

sisaldab alamdiagrammi linki.

Rakendus ei ole mõeldud spetsiaalselt string diagrammidega töötamiseks ja ei paku võimalusi diagrammide ümberkirjutamiseks või lihtsustamiseks. Samuti puudub rakenduses võimalus diagrammide eksportimiseks TikZ-koodi formaadis, raskendades diagrammide kasutust teadustöödes.

3.4 MVP rakendus

MVP rakendus, mis anti autoritele lõputöö alguses, on rakendus, kus on võimalik luua string diagramme, neid eksportida ja importida, kuid nende diagrammide loomine on väga algeline ning mahukate diagrammide koostamine on aeglane. MVP rakendus on arendatud programmeerimiskeeles Python ning see on töölauarakendus, mille kasutamiseks pole vaja interneti ühendust. Rakenduse kasutamine eeldab tehnilisi oskusi, sest rakendus on olemas ainult lähtekoodi formaadis.



Joonis 5. MVP rakenduse kasutajaliides.

MVP rakenduse üks tugevustest on lihtne diagrammide muutmine, võimaldades diagrammis komponentide lohistamist ning juhtmete kerget loomist ühenduste vahel. Samuti on rakenduses diagrammide salvestamise funktsioon. Diagramme saab salvestada JSON (*JavaScript Object Notation*) objektidena, mis võimaldab diagrammide jagamist ja hilisemat muutmist rakenduses. Samuti on võimalik diagrammide eksportimine PNG failidena ning

diagrammidele pseudonotatsiooni genereerimine. Rakenduses on võimalik luua alamdiagramme mitme komponendi valimisel. Alamdiagrammi loomisel asendatakse valitud komponendid ühe kastiga ja komponendid liigutatakse automaatselt alamdiagrammile.

Rakenduse üheks puuduseks on võimalus luua ainult nelinurkseid kaste koos ringidega, mille suurust ei ole võimalik muuta vastavalt kasutaja soovidele. Kasutajaliides on algeline ning vanamoodne. Diagrammi komponentidel puuduvad kokkupõrked, mille tõttu mõned komponendid võivad jääda kastide taha peitu. Diagrammist TikZ koodi genereerimine puudub. Rakendusest puuduvad ka paljud kasutajamugavust parandavad funktsionaalsused nagu elementide kopeerimine, kleepimine ja lõikamine, suumimine jne.

3.5 Analüüsi tulemused

Analüüsitud rakendused on kõik üpriski erinevad ja pakuvad erinevaid kasutusvõimalusi. Cartographer lubab kasutada reegleid string diagrammidega töötamiseks, aga tal puuduvad enamus tavapärastest kasutajamugavust parandavatest funktsioonidest, samas kui Diagrams.net on keskendunud just kasutajamugavusele ja laialdastele diagrammide kujutusvõimalustele, jättes kõrvale string diagrammide matemaatilise tausta. TikZiT ja MVP rakendus on justkui nende kahe vahepealsed, pakkudes mõningaid string diagramme piiravaid reegleid, kuid mitte nii head kasutajamugavust kui Diagrams.net. Rakenduste võrdluse tulemusena otsustati võtta rakenduste parimad aspektid ja implementeerida need oma rakendusse. Näiteks hoiti olulisel kohal Diagrams.net puhul nii suumimis-, kopeerimis-, lõikamis- ja kleepimisfunktsiooni kui ka objektide ja nendevaheliste joonte laialdasi stiili valikuid. TikZiT'i puhul võimalust tikz-koodi genereerimiseks ja Cartographer'i puhul võimalust luua ümberkirjutusreegleid, mille implementeerimine jäi antud töö skoobist välja. Allolev tabel (Tabel 1) toob välja lühidalt analüüsi tulemused.

Tabel 1. Diagrammi redaktorite analüüsi tulemused.

	Cartographer	**TikZiT**	**Diagrams.net**	**MVP rakendus**
Erinevad komponendid	Puudub	Ristkülikud ja ringid	Lai valik	Ristkülikud ja ringid
Salvestamine ja jagamine	Puuduvad	Salvestamine	Salvestamine ja jagamine	Salvestamine
Kasutaja-kogemust parandavad funktsioonid	Puuduvad	Piiratud	Lai valik	Puuduvad
Kasutus-valdkond	String diagrammide lihtsustamine	Diagrammist TikZ koodi loomine	Eri valdkonna diagrammide joonestamine	String diagrammidega töötamine

4 Arendus

Lõputöö arendusfaasi jagasid autorid kahenädalasteks sprintideks. Sprindi alguses toimus koosolek, kus planeeriti järgneva sprindi piletid. Kuid vajadusel lisati ka sprindi vältel väiksemaid pileteid juurde. Sprindi lõpus tehti retrospektiivne koosolek, et hinnata, kui edukas on olnud käesolev sprint. Koosoleku tulemusena kirjutati raport autorite individuaalsest panusest projekti. Iganädalaselt toimus koosolek juhendajaga, kus näidati hetkeseisu ning arutati edasist suunda.

Projekti jooksul toimus tööjaotus vabalt, peale pileti lahendamist võis vabalt valida järgmise pileti, millega tegeleda. Ei olnud kindlat ette mõeldud jaotust, kes millega tegeleb.

4.1 Suumimine

Suumimine võimaldab rakenduse kasutajal diagrammi suuremaks teha, mis omakorda aitab luua suuremaid diagramme ning aitab nende haldamisega. Samuti tuleb see kasuks kasutajatele, kes soovivad diagrammi nö lähemalt näha ja selle peal liikuda. MVP rakenduses see võimalus puudub ning rakendust kasutades tekib piiratud tunne. Selle kitsaskoha parandamiseks otsustati lisada suumimine rakendusse.

Teek, millega arendatakse rakendust, Tkinter ja selle Canvas objekt, mille peal luuakse diagramme, võimaldavad suumimist, muutes objekte suuremaks või väiksemaks vastavalt hiire asukohale. Selle lisamine osutus lihtsaks, kuid sellega esinesid probleemid, nimelt kuna sisseehitatud suumimine muutis ainult Canvasel olevaid koordinaate, siis peale suumimist, kui prooviti asju liigutada, hüppasid need tagasi oma vanale asukohale. Lisaks sellele selgus, et välja suumides võisid tekkida ootamatud visuaalsed vead, mis muutsid diagrammi kõlbmatuks. Sellest tulenes vajadus välja suumimise piirangu jaoks, et ei saaks suumida välja algsest vaateväljast.

Nende probleemide lahendamiseks arendati suumimise süsteemi, mis muutis nii Canvase koordinaate kui ka objekti koordinaatide muutujaid. See võimaldas asjade liigutamist peale

suumimist. Peale seda, kuna oli vajadus piirata maksimum vaatevälja, loodi diagrammide nurkadesse objektid, mille järgi sai kontrollida, kas välja suumimine on lubatud või mitte. Need nurgad hoidsid algseid nurga asukohti relatiivselt suumiga. Sellega kaasnes probleem, et kui sooviti välja suumida hiire asukohast, siis ei tohi see toimuda nii, et osad nurgad ei naaseks oma originaalsele kohale ja võimaldaksid lubatud vaateväljast välja minemist. Selle lahendamiseks loodi esmalt süsteem, mis salvestas sisse suumimise asukohad ning sooritas välja suumimist samadest kohtadest, et kasutaja vaateväli jõuaks alati tagasi algusesse täpselt nii, et nurgad on õigesse kohta naasnud.

Lahendus oli töötav, kuid välja suumimine tekitas segadust seoses sellega, et suumiti välja teisest kohast kui sooviti. Sellise lahendusega ei oldud rahul ning otsustati mõelda välja arvutus, millega saaks leida, kui palju peab punkti asukohta liigutama, et toimuks korrektne sisse suumimine ning mida saaks ka kasutada välja suumimise jaoks, et leida objekti eelmist asukohta enne suumimist.

$$z - \frac{(z - x_0)}{\alpha} = x \tag{4.1}$$

Valemis 4.1 z tähistab hiire asukoha koordinaate, kust suumimist sooritatakse, x_0 tähistab objekti koordinaati enne suumimist ning α on suumimise kordaja, lõputöö rakenduses on sisse suumimisel α väärtus 0 ja 1 vahel. x tähistab uut asukohta peale suumimist. Kuna zoomimise muutja α väärtus on konstandina määratud rakenduses, on võimalik seda valemit kasutada mõlemat pidi, nii sisse suumimisega kui ka välja suumimisega, sisse suumimisel peab olema suumimise muutuja väärtus 0 ja 1 vahel ning välja suumimisel peab kasutama selle pöördväärtust.

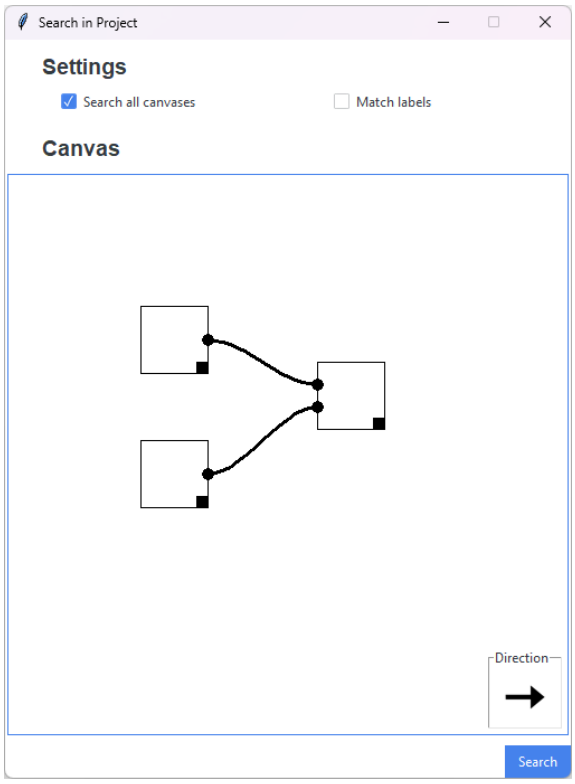
Välja toodud arvutus on tähtis selleks, et kontrollida enne suumimist, kas selle läbiviimine on lubatud. Pärast selle valemi loomist oli võimalik muuta suumimine selliseks, et nii sisse- kui ka välja suumimine toimub hiire asukoha põhjal ning ei minda kuidagi välja algsest vaateväljast.

4.2 Diagrammi sisene otsing

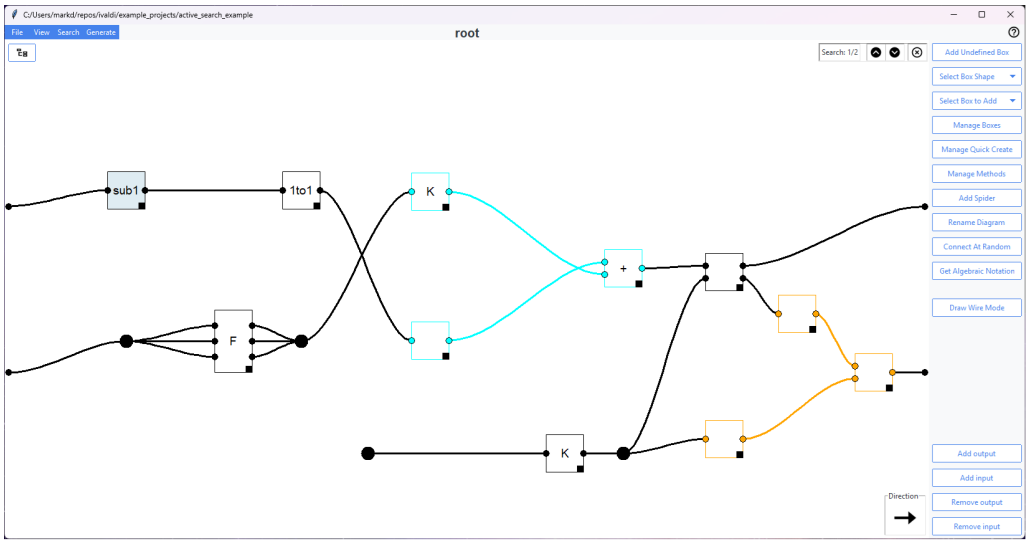
Suurte diagrammidega tegeledes, kus võib olla veel mitu alam-diagrammi, võib kindlate kohtade leidmine osutuda raskeks; selleks on rakendusse lisatud diagrammisisene otsing.

Tänu sellele saab kasutaja kergelt leida oma diagrammist kindlaid osi, mida soovib, ning need tuuakse nähtavale. Peale osade leidmise on see kasulik ka osade asendamiseks, kui kasutaja soovib midagi asendada millegi muuga, ei pea ta ise manuaalselt otsima, vaid saab kergelt kasutada otsingut ning seejärel asendada vastavad kohad diagrammil.

See funktsioon rakenduses on implementeeritud sellisel viisil, mis annab kasutajale võimaluse eraldi aknal ise joonestada osa, mida ta soovib otsida (vt Joonis 6). Seejärel tuuakse diagrammil välja kõik sellised osad ning lubatakse nende vahel liikuda (vt Joonis 7).

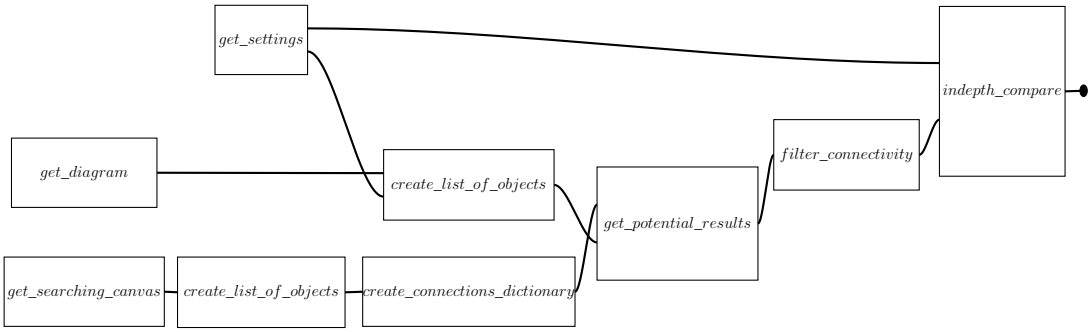


Joonis 6. Otsingu aken.



Joonis 7. Diagramm aktiivse otsinguga.

Diagrammi osade võrdluseks luuakse alguses ühenduste sõnastik nii otsitavast osast kui ka diagrammist ning selle alamdiagrammidest. Seejärel käidakse läbi diagrammi sõnastik ning otsitakse sarnast osa otsitava osa sõnastikule. Kui on leitud kõik sarnased kohad, filtreeritakse välja kõik, mis selgelt ei sobi, näiteks vähemate objektide järgi, ning peale seda minnakse sügavamasse võrdlusesse. Sügavamas võrdluses viiakse läbi objekti võrdlused, ühendatud elementide ja ühenduste hulga võrdlused. Sügav võrdlus on loodud nii, et kui otsitavas kastis ühel poolel ei ole täpsustatud ühenduste arvu, siis võib diagrammis kastil sellel küljel olla mistahes palju ühendusi.



Joonis 8. Illustreeriv diagramm otsingu algoritmist.

Otsingule on loodud piirang, et otsida saab ainult ühendatud osi, ehk ei ole võimalik otsida osa diagrammist, kus kõik objektid ei ole ühendatud.

Otsingu jaoks on veel loodud sätted. Praegusel hetkel on loodud ainult 2 sätet, kuid algoritmi on kerge muuta, kui soovitakse lisada veel sätteid. Esimene säte võimaldab kasutajal valida, kas ta soovib sooritada otsingut ainult praegu kasutajal ees olevas diagrammis või soovib ka otsida läbi alamdiagrammide. Teine olemasolev säte lubab kasutajal täpsustada, kas kastide nimed peaksid ühtima.

4.3 TikZ koodi genereerimine

PGF (*Portable Graphics Format*) ja TikZ on keeled, mida on võimalik kasutada vektorgraafika loomiseks erinevates dokumendi formaatides nagu LaTeX, TeXja ConTeXt. PGF'i ja TikZ'i erinevus on selles, et PGF on madala taseme keel ning TikZ on kõrgema taseme keel, mis kasutab PGF'i, et luua vektorgraafikat.

Graafide joonestamine TikZ'is võimaldab teadlastel kiiresti rakendada ja katsetada uusi graafide joonestamise algoritme, mida teadlased saavad koheselt kasutada graafide kvaliteetsete jooniste koostamiseks. [20]

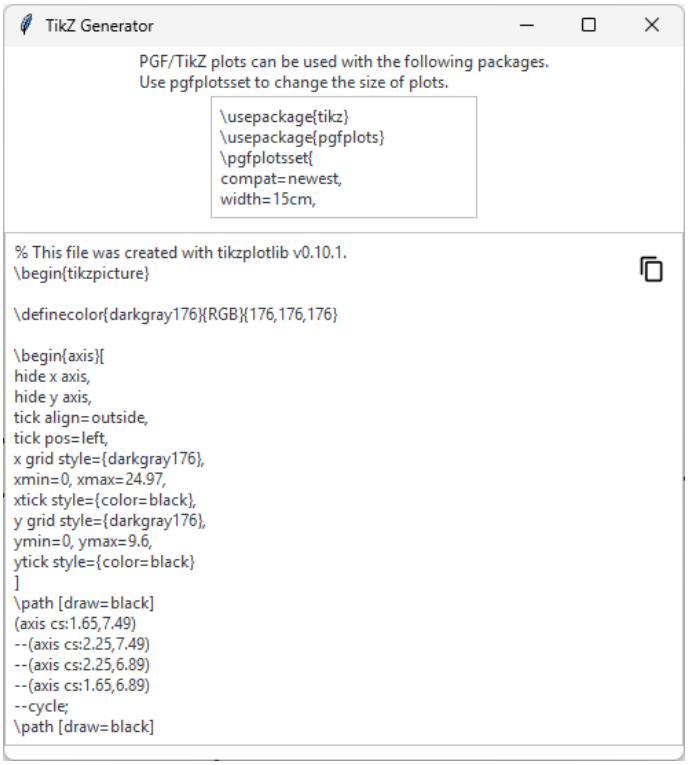
Samuti saab TikZ'i kirjeldada kui süsteemi, mida saab kasutada väga kvaliteetse graafika määramiseks. Näiteks võib selle süsteemi positiivseteks teguriteks pidada pildielementide täpset paigutamist, TeX fontide kasutamist ja matemaatilise kirjapildi kaasamise võimalust. [21]

TikZ on üks levinumaid viise string diagrammide esitamiseks, kuid selle õppimine diagrammide joonestamiseks võtab aega ning käsitsi TikZ koodi kirjutamine võib olla aeglane. TikZ diagrammide käsitsi loomine takistab inimese keskendumist sisule, kuna peab ise kirjutama diagrammi jaoks koodi. Selle vastu just aitab võimalus rakenduses TikZ koodi genereerida, millega saab rakenduses töötatud string diagrammi kergelt LaTeX dokumenti.

Lõputöö raames arendatud rakenduse üks eesmärke oli võimaldada rakenduse sees joonestatud string diagrammi eksportida TikZ koodi, et seda saaks mugavalt esitada LaTeX failides. See sai realiseeritud kasutades Matplotlib ja tikzplotlib teeke. Tikzplotlib võimaldab genereerida TikZ koodi Matplotlib'i graafidest, seega oli vaja luua Matplotlib'i graaf, mis vastas rakenduses joonestatud string diagrammile. Peale Matplotlib'i graafi loomist oli vaja kasutada tikzplotlib, et luua TikZ kood.

Arenduse käigus leiti, et teek tikzplotlib on aegunud ning seoses teiste teekide uuendustega ei ole see enam mõndadel juhtudel kasutatav. Uurimisel leiti, et see oli ainus teek, mis võimaldas rakendusele sobivat funktsiooni. Seetõttu ei jäänud muud üle, kui lisada tikzplotlib lõputöö koodivaramusse ning parandada probleemid ise.

Kui oli leitud viis, kuidas saada TikZ kood diagrammist, tehti rakendusele eraldi aken (vt Joonis 9), mis näitab kasutajale genereeritud TikZ koodi ning selgitab kasutajale, kuidas seda LaTeX failides kasutada saab.



Joonis 9. TikZ generaatori aken.

4.4 Koodi redaktor ning funktsioonide haldamine

Selle lõputöö üks põhieesmärke on uurida, kas graafiline ülevaade muudaks koodi kirjutamise ja mõistmise intuitiivsemaks. Selle uurimiseks peab olema võimalik rakenduses kirjutada koodi. Kuna lõputöö alguses ei olnud rakenduses võimalik kuidagi koodi lisada, pidi looma võimaluse kasutajal kirjutada rakenduses koodi.

Selle jaoks loodi rakendusse koodi redaktor, mille abil saab kastidele määrata koodina funktsioone. Rakenduses loodi võimalus kasutajal kirjutada rakenduse siseses koodi redaktoris Python'i koodi.

Koodi redaktori jaoks otsustati kasutada Chlorophyll koodi redaktori teeki, kuna see on loodud kasutades Tkinter teeki, mida kasutab ka lõputöö rakendus. See tagas koodiredaktori kerge integreerimise rakendusse. Chlorophyll koodi redaktor on avatud lähtekoodiga lihtne redaktor, arendatud Python'is, selle lähtekood on saadaval GitHub'is [22]. Lisaks teegist saadud koodiredaktorile pidi veel looma *wrapper* klassi, mis võimaldas lisada nuppe või teha väikseid muudatusi teegist saadud redaktorile. Redaktorile lisati erinevad salvestamisfunktsioonid ning redaktoris oleva koodi eksportimine Python'i failina.

Joonis 10. Rakenduses olev koodiredaktor.

Kastidel kasutatavaid funktsioone luuakse läbi koodi redaktori, kuid need ei ole otseselt kasti objektiga ühendatud, vaid hoitakse rakenduse põhiklassi muutujas, mis sisaldab JSON faili sisu, kuhu funktsioonid on salvestatud. Põhiklassis hoitakse funktsioone sõnastikus, kus võtmeks on selle funktsiooni nimi. Selle sama nimega määratakse ka kastidele funktsioon, ehk kui muudetakse kasti nimi vastavaks, siis hiljem on selle kasti funktsioon kättesaadav läbi kasti nime. See süsteem võimaldab eraldada kastid ja nende funktsioonid, mis lubab kergelt taaskasutada funktsioone, sest kasutaja peab ainult määrama nime kastile, kui funktsioon on juba loodud.

Seejärel arendati aken, kust kasutajad saavad hallata olemasolevaid funktsioone. Seal saab lisada uusi funktsioone, kustutada olemasolevaid funktsioone, muuta funktsiooni

tähistatavaid nimesid või funktsiooni koodi. Lisada saab funktsioone kahel viisil, kas ise kirjutades või välise Python faili importimisel rakendusse.

4.5 Diagrammi keeramine

Varasemalt oli rakenduses võimalik kujutada diagramme ainult horisontaalselt suunaga vasakult paremale, kuid diagrammide keeramine lubab rakenduse kasutajal kujutada diagramme horisontaalselt mõlemas suunas ning ka vertikaalselt mõlemas suunas (vt Lisa 2). See on oluline just kasutajale, kes kasutusvaldkonnast tulenevalt vajab kindlat suunda diagrammides informatsiooni liikumise suuna näitamiseks.

Antud funtsiooni lisamisel oli tähtsal kohal, kuidas diagramme keeratakse. Diagrammide keeramisel otsustati, et diagramme ei keerata igakord ainult 90 kraadi vaid arvestatakse, et diagrammid jääksid samaseks. Seega omab diagrammi keeramisel olulist rolli diagrammide tõlgendamine. Samasuse tagamiseks otsustati horisontaalse suunaga diagramme kujutada kui üksteise peegeldused vertikaalse kesktelje suhtes. Sarnaselt toimitakse ka vertikaalselt kujutatud diagrammide teisendusega ainsa vahega, et peegeldusteljeks on horisontaalne kesktelg.

Diagrammide keeramiseks lisati objektidele lisa koordinaadid visuaalse asukoha jaoks. Koordinaatide muutmisel muudetakse objekti loogilisi koordinaate, mille põhjal arvutatakse objekti visuaalsed koordinaadid. Selline lahendus sai valitud, et tagada parem ühildatavus juba eksisteeriva loogikaga. See lubas kasutada juba varem loodud reegleid minimaalsete muudatustega, tagades, et reeglid kasutavad loogilisi koordinaate.

Lisaks objekti koordinaatide muutmisele lisati ka objekti enda keeramise. See on oluline just kastide puhul, mis ei ole sümmeetrilise kujuga. Kastide keeramiseks kasutati Stephan D. Evans'i lahendust, mida muudeti vastavalt vajadustele [23]. Kastide kuju on defineeritud kui list punkte, mida liigutatakse kasti keskpunti suhtes vastavalt diagrammi suunale. Peale punktide keeramist punktide asukohad normaliseeritakse, et tagada punktide paiknemine nulli ja kasti mõõtmete väärtuste vahel.

4.6 Juhtmete ning ühenduse eri tüübid

String diagrammide üks kasutusi on algoritmide ja protsesside visualiseerimine. Seoses ressurssitundlike protsesside esitamisega on tihti vajadus näidata, mis ressurss mingi juhtme peal nö liigub. Ilma selle täpsustuseta oleks ressurssitundlike protsesside string diagrammidega väga raske töötada, kuna ei ole teada, mis ressurssidega tegeletakse.

Lõputöö raames arendati ka funktsioon, mis võimaldas kasutajal eristada juhtmeid ja ühendusi ning määrata neile kindlaid tüüpe. See andis võimaluse kasutajale rakendada lisa piiranguid diagrammile, nimelt kuna üksteisega saab ühendada ainult sama tüüpi ühendusi, siis saab kasutaja ise määrata, mis ressursse kuskil lubatakse kasutada.

Rakenduses on kasutajal võimalik ise defineerida kuni 10 erinevat tüüpi ühendusi ning juhtmeid, need on diagrammil kõik ka märgistatud eri viisidel (vt Lisa 2) ning vajadusel on toodud välja tüübi nimi ühenduste juures. Ühenduse ja juhtme tüübid on üksteisega seotud, juhtme tüüpi eraldi ei saa muuta, vaid see sõltub ühenduse tüübist, millega see ühendatud on. Lisaks ressurssi näitamisele muudab see lisandus diagrammi joonestamisvõimalusi. Diagrammi joonestamist muudeti sellega, et lubatakse ühendada ainult samasugust tüüpi ühendusi. Selle reegli erandiks on ühenduste vaikimisi tüüp, kuhu on võimalik ühendada teisi tüüpi juhtmeid, kuid ainult siis, kui vaikimisi ühendusse pole midagi muud ühendatud. Sellisel juhul muudetakse vaikimisi ühenduse tüüp vastavusse sellega, mis sellesse ühendati.

4.7 Erinevad kastide kujud

Rakenduse üks eesmärke on võimaldada kasutust erinevates domeenides, sellega seoses tuleks kasutajatele sõltuvalt valdkonnast kasuks erinevate kujudega kastid, et tähistada eri funktsioone diagrammis. Üks näide oleks, kui soovitakse kasutada rakendust loogika domeenis, siis oleks kasulik, kui saaks loogika väravate kujudega kaste luua.

See funktsioon lisab üldist kohandatavust rakendusele ning võimaldab rohkem domeeni spetsiifilist kasutust, samuti lisab see rakenduse visuaalsele küljele juurde, võimaldades teha mitmekesisemaid diagramme.

Geomeetria loomiseks on *tkinter* teegi sisseehitatud funktsioonid, mida kasutati kastide visuaalseks loomiseks. Algselt oli rakenduses võimalik luua ainult nelinurkseid kaste ning

nende suurust sai kasutaja peale kasti loomist ise muuta, kuid puudus muu geomeetria võimalus kastide jaoks.

Lõputöö käigus otsustati lisada rakendusse loogikaväravate kujudega kastid (vt Lisa 2). Selle implementatsiooniks pidi kirjutama arvutused, et luua vastavaid kujusid. Rakenduses on veel võimalus sättida kasti vaikimisi kuju ning muuta kastide kuju jooksvalt, peale loomist kasutaja soovidele.

4.8 Dokumentatsioon

Lõputöö raames täiendatud rakendust on plaanis peale autorite lõputööd edasi arendada, mistõttu tulevaste arendajate töö lihtsustamiseks otsustati kirjutada rakendusele dokumentatsioon. Dokumentatsioon aitab järgmistel arendajatel rakendusest kergemini arusaada ning mõista, kuidas rakendus on üles ehitatud ja mida erinevad objektid võimaldavad ning mis nende funktsioonid on.

Lõputöö käigus on kirjutatud dokumentatsiooni rakenduse kasutajaliidesele. Dokumenteerimine koosneb informatsiooni kirjutamisest eraldi failidesse. Need failid sisaldavad infot erinevate klasside eesmärkide kohta, nende funktsioonide ja kasutuse kohta. Samuti on loodud kergesti järgitavad tabelid objektide muutujatest, muutuja tüüpidest ning muutuja kirjeldusest. Loodud on ka objektide funktsioonide nimekirjad, kus on kirjeldatud, mida iga funktsioon teeb, milliseid parameetreid kasutab ning mida tagastab.

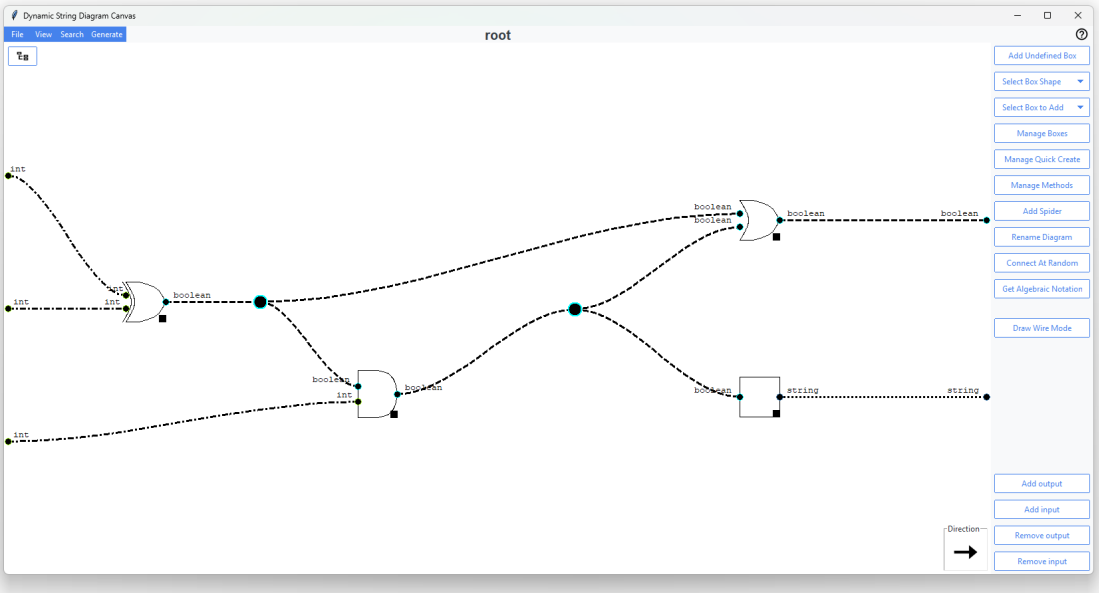
4.9 Muud funktsioonid

Rakenduse arenduse käigus lisati ka mitmeid väiksemaid funktsioone. Väiksemate funktsioonide eesmärgid olid enamasti kasutajamugavuse parandamine või asjade visualiseerimine. All on välja toodud funktsioonid, mis said rakendusse lisatud.

- Topeltklõksuga juhtmete loomis viis.
- Juhtmete visualiseeritud loomine.
- Kokkupõrked diagrammi objektide vahel.
- Rakendusse diagrammide importimine peale käivitust.
- Rakendusse diagrammide importimine alamdiagrammina.
- Modernsema välimuse lisamine kasutajaliidesele (vt Joonis 11).

- Objektide kopeerimine, lõikamine ja kleepimine.
- Objektide ja ühenduste asendamine kleepimisega.
- Väliste Python'i failide importimine rakendusse.
- Abistava informatsiooni akna loomine.
- Topeltklõpsuga alamdiagrammi sisenemine.
- Alamdiagrammide "avamine" tuues sisemuse põhidiagrammile.
- Suurte diagrammide dünaamiline suuruse muutmine.
- Tööriistariba loomine.
- Mitme eraldi oleva objekti selekteerimine.
- Kontekstimenüüga objektide loomine.

Lisaks varem mainitud funktsioonidele tehti arendustöö käigus ka palju veaparandusi, mis olid tulnud nii MVP versioonist kui ka arendusprotsessist. Lisaks veaparandustele ja funktsioonide lisamisele tegeleti veel MVP rakenduse koodi optimeerimise ja struktuuri parandamisega. MVP rakenduses esines palju kohti, kus koodi oli kirjutatud topelt, ning funktsioonid olid liiga laiali jaotatud, muutes probleemide leidmise raskeks. Rakendusele ühikteste luues kasutati pidevat integratsiooni (CI) ühiktestide jooksutamiseks. See oli abiks probleemide tuvastamisel, kui uus funktsioon lisati.



Joonis 11. Rakenduse kasutajaliides.

5 Tulemused

Lõputöö käigus on suudetud saavutada enamus eesmärke, TikZ koodi loomine rakenduses tehtud diagrammist on võimalik, rakendus on arenduse käigus säilitanud uute funktsioonidega tagurpidi ühilduvuse, mis võimaldab teiste arendatud osadega kombineerimist, ning on säilitatud pseudo-notatsiooni funktsionaalsus peale uute funktsioonide lisamist, mis muutsid diagrammide näitamist ja loomist.

Alameesmärkidest jäi paindlikkuse saavutamine osaliselt täitmata. Juurde lisati võimalusi, kuidas saab diagramme joonestada ja tõlgendada, kuid kasutajal endal ei ole seda võimalik muuta.

Kasutajakogemuse eesmärgist ning rakenduses programmeerimise eesmärgist on all kirjutatud täpsemalt. Lisaks kasutajatestimisele ja rakenduses programmeerimise testimisele on rakenduse jaoks veel loodud üle 200 ühiktesti.

5.1 Kasutajate tagasiside

Peale rakenduse arendust viidi läbi kasutajatestid. Selle jaoks koostati viis ülesannet (vt Lisa 4), mida kasutajad saaksid rakendust kasutades lahendada. Nendest ülesannetest kolm olid lahendatavad MVP versiooni peal, seega lasti kasutajatel testida ka MVP versiooni, et saada tagasisidet uute funktsioonide kohta.

Testimine toimus, kas läbi kõne või kohapeal testijaga. Testimisel jälgiti, kuidas testija ülesandeid lahendab ning tehti märkmeid lahenduskäikudest ja probleemidest. Testimine viidi läbi inimeste seas, kes ei olnud varem kokku puutunud selle rakendusega ning kes ei ole varem põhjalikult string diagrammidega tegelenud, kuid olid tuttavad programmeerimisega. Testi lahendamise käigus aidati kasutajaid minimaalsel määral, vihjeid edasi lahendamiseks anti ainult siis, kui testija jäi täiesti kinni millegi taha. Peale ülesannete lahendamist küsiti testijatelt vabas vormis tagasisidet.

Ülesanded sisaldasid diagrammide loomist, muutmist, importimist ja eksportimist, lisaks sellele ka veel koodi lisamist kastidele, kasutades rakenduses olevat koodiredaktorit.

5.1.1 Kasutajatestimise tulemused

Testimise käigus avastati uusi vigu rakenduses, millest korratavad vead parandati ära järgnevate päevade jooksul peale testimist.

MVP versioonis tekkisid probleemid diagrammi importimisega, kuna see oli võimalik ainult rakendust uuesti lahti tehes, siis läks kasutajatel aega enne, kui nad sellest aru said. Probleeme tekkis ka juhtmete loomisega, kuna juhtmete loomist ei olnud visuaalselt näha, ei saanud kasutajad õigesti aru, kuidas juhtmeid luua. Testijad üritasid kasutada klahve, kuid nendele ei olnud ühtegi funktsiooni määratud.

Uues versioonis testülesannete lahendamisel kasutasid testijad intuitiivselt järgmiseid lisatud funktsioone: kontekstimenüüga diagrammi objektide loomine, topeltklõpsuga juhtmete loomine ja topeltklõpsuga alamdiagrammi sisenemine. Samuti kasutati ära klahvidele määratud funktsioone nagu kopeerimine, kleepimine ja kustutamine. Lisaks neile kasutati ka funktsioone nagu uue diagrammi loomine või importimine rakenduse seest, mis olid varasemalt võimalikud ainult rakendust tööle pannes.

Võrreldes MVP versioonis lahendatud ülesannetega tekkis uues versioonis palju vähem vigu, mis tegid rakenduse katki, ja suudeti kiiremini leida lahendused ülesannetele. Kuigi kasutajatel läks uues versioonis paremini võrreldes vanaga, osutus funktsioonide leidmine kohtati keeruliseks. Paljud lisatud funktsioonid ei ole ekraanil otseselt nähtavad ning kuigi on olemas nupp, mis avab abiinfo akna, ei otsustanud testijad seda otsida või kasutada. Mistõttu võttis testijatel rohkem aega, et aru saada, kuidas rakendust kasutada. Selle probleemi lahendamiseks oleks vaja luua rakendusse lühike õpetus, kuidas rakendust alguses kasutada.

Peale testimist küsiti testijatelt tagasisidet nii uue kui ka MVP versiooni kohta. Sellele vastati, et uue versiooni kasutajaliides näeb vanaga võrreldes parem välja, juhtmete kergem loomine ning visuaalne esitlus loomise ajal teeb kasutamise kergemaks ning kopeerimine, kleepimine ja asendamine olid head ja kasulikud lisatud funktsioonid.

Testijatelt saadi soovitusi uute funktsioonide jaoks, nagu automaatne salvestamine, *undo* nupp jne.

Tulemusi vaadates saab öelda, et arenduse käigus on parandatud kasutajakogemust ning muudetud diagrammide loomine intuitiivsemaks.

Lisaks MVP versiooni ja uue versiooni võrdlusele saadi ka tagasisidet uute funktsioonide kohta, mida vanas versioonis ei olnud. Eriti kiideti otsingu funktsiooni ning kopeerimist ja kleepimisega asendamist. Tagasisides saadi veel soovitusi uutele klahvi funktsioonidele, osasid, mida oldi varem arvestatud, ning osasid, millele ei oldud varem mõelnud. Lisaks sellele saadi tagasisidet, et erinevate juhtme tüüpide lisamine võiks teha rohkem visuaalseks ning anda kasutajale võimalus ise valida, mis tuleks järgmise tüübina.

Tabel 2. Kasutajatestimise kokkuvõte.

Positiivne	**Negatiivne**
Osati kergelt luua juhtmeid	Topeltklõpsuga juhtme lisamise asemel eelistatakse klahvi allhoidmist
Kasutati uue diagrammi loomist rakendusest	Ühenduste tüüpimist oli raske leida
Konteksmenüüga asjade loomine oli mugav	Osad levinud klahvide funktsioonid on puudu
Otsingut ja kopeerimist kiideti	Ei leitud vahest funktsioone nagu kopeerimine või kontekstmenüüga asjade loomine
Koodi lisamine kastidele oli lihtsasti mõistetav	Võiks olla rohkem visuaalseid indikaatoreid nagu kasti lukus olemise indikaator
Osati intuitiivselt navigeerida alamdiagrammide vahel	Tekkisid mõned visuaalsed vead

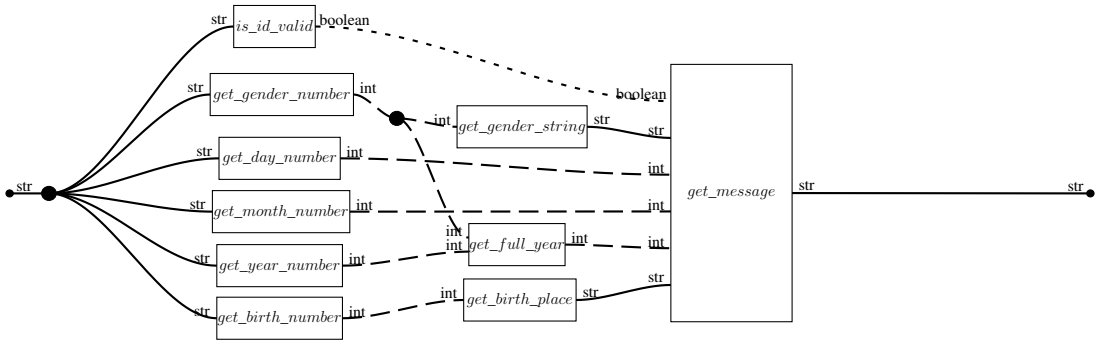
5.2 Programmeerimine rakenduses

Lõputöö käigus uuriti graafilise ülevaate mõju koodi mõistmisele ja intuitiivsusele. Selle uurimiseks arendati esmalt võimalus kirjutada rakenduses Python'i koodi ja seejärel kasutati seda programmeerimisülesannete lahendamiseks ning lahenduste esitamiseks.

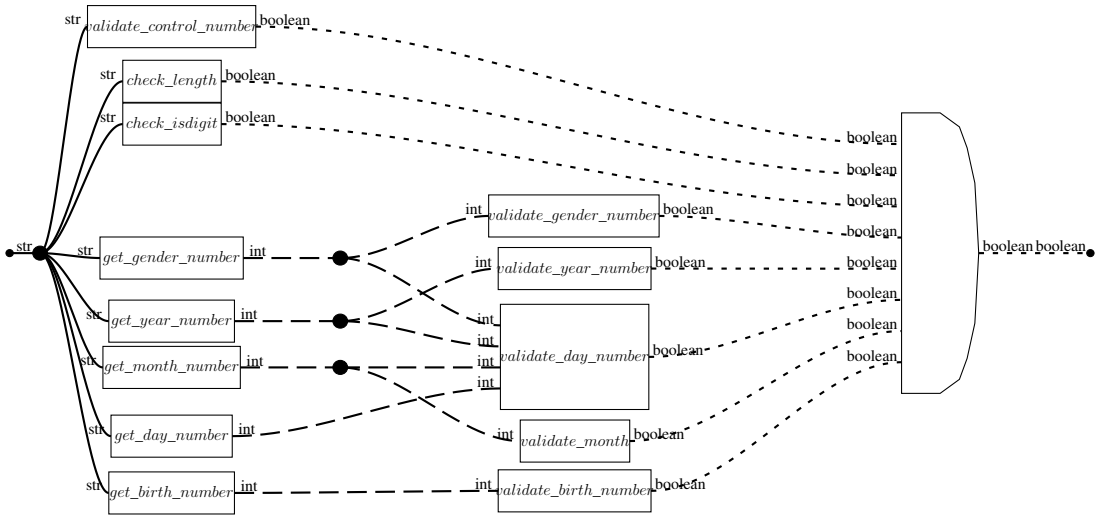
Ülesandeks, mida prooviti lahendada, oli TalTech 2022 programmeerimise algkursuselt võetud isikukoodi ülesanne. Lahendamiseks vaatati eksisteerivat lahendust. See kood oli nö algaja kood ning lahendamise käigus leiti, et koodi struktuuri peab muutma ja selle juurde pidi eraldi funktsioone looma. Koodistruktuuri muutmine antud olukorras oli kasulik, sest see aitas optimeerida algoritmi ja kasutada vähem resursse. See omakorda saab algajale õpetada just optimaalsemat koodi kirjutamist.

Kuigi ülesande lahendamine oli võimalik, ei olnud see väga mugav just seetõttu, et kõikidesse kastidesse eraldi koodi kirjutamine oli aeganõudev ja koodiredaktoris puudus süntaksikontroll ning kuna teiste kastide kood oli n-ö peidetud, siis oli vigade leidmine ja parandamine keerukas.

Sellele vaatamata oli algoritmi kujutamine diagrammina (vt Joonis 12) päris kasulik ning andis hea ülevaate sellest, kuidas algoritm töötab. Selline ülevaade saab aidata algajatel paremini aru saada, kuidas sellist ülesannet peaks lahendama või kuidas algoritm töötab.



Joonis 12. Eesti isikukoodi info kättesaamis algoritm lihtsustatud.



Joonis 13. Eesti isikukoodi valideerimise algoritm. Alamdiagramm "is_id_valid".

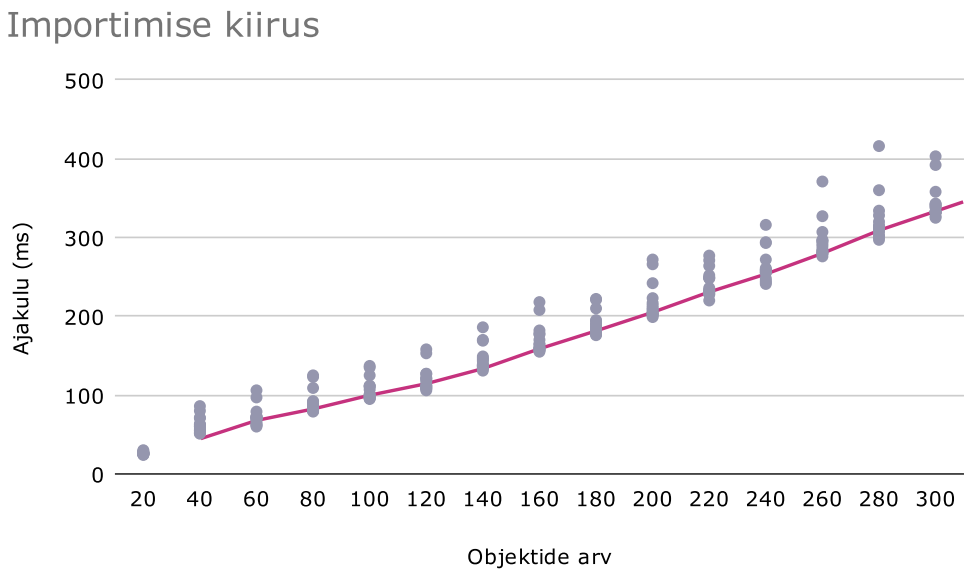
Ülesannet lahendades selgus, et kõige paremini saab rakenduses näidata koodi, mis sisaldab muutujate loomist ja tõeväärtuskontrolle. Kuna rakenduses ei ole võimalik luua tsükleid, siis diagrammis ei ole nende visuaalne esitamine võimalik ning saab tsükli asemel ainult luua kast, kuhu sisse saab tsükli koodina kirjutada. Samuti ei saa hästi visualiseerida funktsioone, mis ei tagasta midagi ega võta parameetreid sisse, kuna nendesse ei tohiks minna ükski juhe, siis need oleksid täiesti omaette kastid.

Lahendatud isikukoodi ülesanne on üpris kerge loogika ja ülesehitusega, ning juba see läheb diagrammil üsna suureks. Üheks võimalikuks lahenduseks on alamdiagrammide loomine nagu kasutati ka antud lahenduses (vt Joonis 13), kuid see lisab diagrammile keerukust ning teeb lugemise raskemaks.

Kokkuvõtvalt on programmeerimine läbi rakenduse võimalik ning graafiline ülevaade teeb koodi mõistmise kergemaks, kuid läbi selle programmeerimine võtab rohkem aega ning vigade parandamine on raskem, kuna ees on graafiline vaade, mitte koodi vaade, kus sisalduvad vead. Peale selle vajab rakenduses programmeerimine vähemalt keskmist programmeerimisoskust, et saaks hästi asju visualiseerida. Programmeerimine rakenduses oleks oluliselt kergem kui kastidesse suudetakse kirjutada kohe funktsionaalne kood nii, et keskenduma peaks ainult kastide ühendamisele ja loogikale.

5.3 Joonestamise efektiivsus

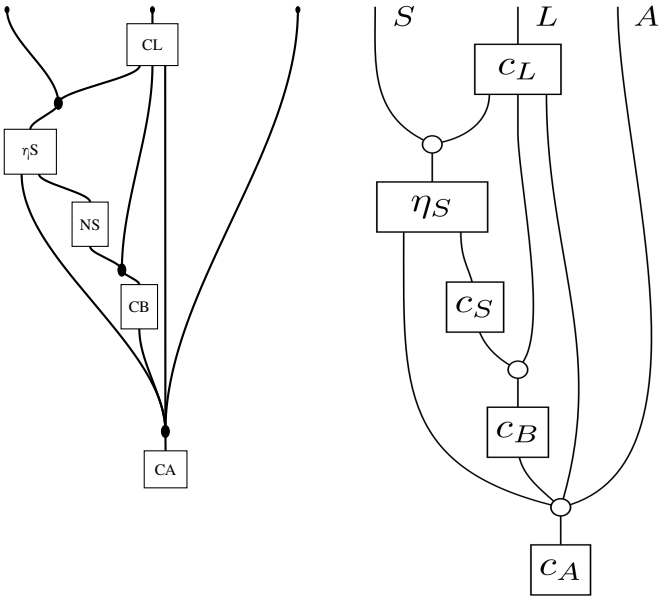
Diagramide joonestamise efektiivsuse leidmiseks mõõdeti rakenduses diagrammide importimise kiirust. Importimine sisaldab objektide (kastid ja *spiderid*), ühenduste ja juhtmete loomist samal viisil kui kasutaja ise looks neid, seega selle järgi on hea mõõta nende efektiivsust. Importimise ajakulu mõõtmisel loodi diagrammid, mis sisaldasid 20 kuni 300 objekti. Objektideks loeti kaste, *spidereid* ja juhtmeid. Juhtmeid oli võrdselt kastide ja *spideritega*. 20 objekti importimisel oli keskmine kiirus 26.2 ms ning 300 objekti importimise keskmine kiirus oli 344.6 ms. Testimise tulemusena selgus, et diagrammi kasvamisel kasvab ajakulu lineaarselt objektide arvuga. Keskmiselt lisandus iga 20 objektiga 22.7 ms aega juurde. Joonis 14 näitab testimise tulemusi, kus ringid näitavad importimise aegu ning läbiv joon näitab keskmist importimise kiirust.



Joonis 14. Importimise kiirus objektide arvu põhjal.

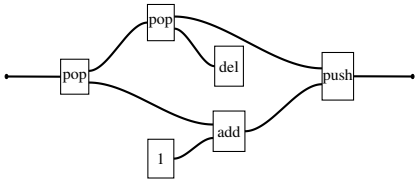
5.4 Diagrammide näited TikZ'iga

Rakenduses loodud diagrammide näitamiseks on kasutatud TikZ koodi, mis on genereeritud rakenduses, demonstreerides, et see funktsioon on olemas rakenduses ning sellega saab näidata diagramme LaTeX dokumentides. Töö jooksul varasemalt välja toodud diagrammid on ka loodud rakenduses ning nende näitamiseks kasutatakse rakendusest genereeritud TikZ koodi.

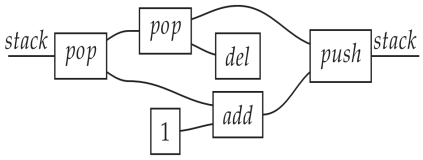


(a) (b)

Joonis 15. Kõrvuti olevad diagrammid: (a) on loodud lõputöö rakenduses, (b) on võetud artiklist [24].

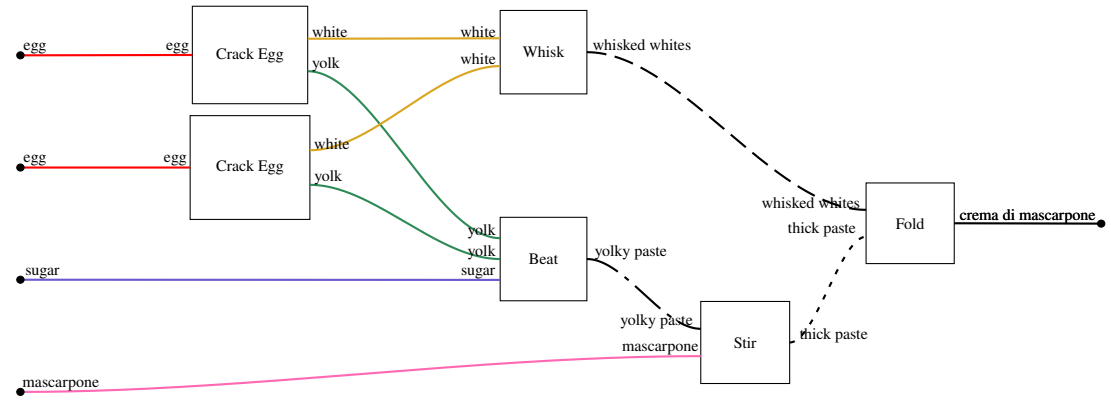


(a)

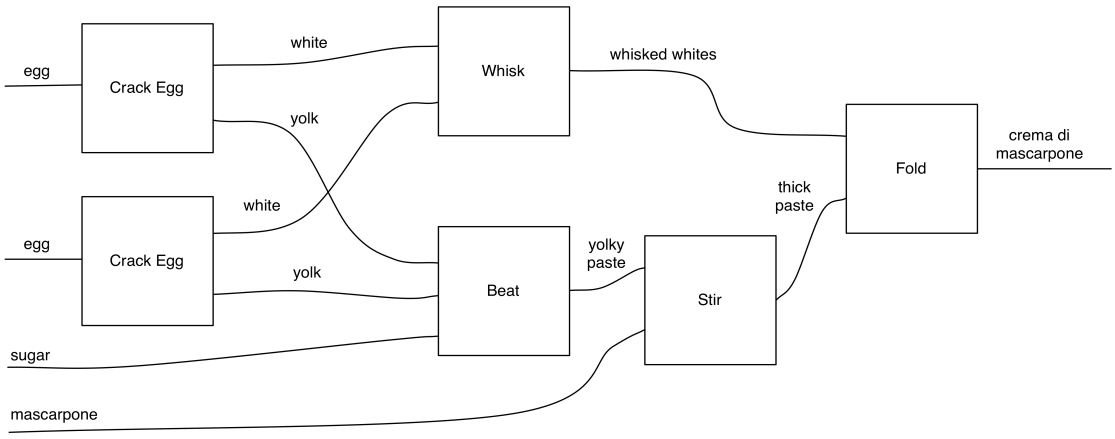


(b)

Joonis 16. Kõrvuti olevad diagrammid: (a) on loodud lõputöö rakenduses, (b) on võetud raamatust [2].



(a)



(b)

Joonis 17. Kõrvuti olevad diagrammid: (a) on loodud lõputöö rakenduses, (b) on võetud lehelt [25].

5.5 Autorite eneseanalüüs

Lõputöö jooksul on õpitud palju kasutajakogemuse parandamisest, seda nii teisi rakendusi analüüsides ja neid omavahel võrreldes, kasutajakogemuse parimate praktikatega tutvudes kui ka rakenduse kasutajatestide tagasisidest saadud informatsiooni kokkuvõttest. Lisaks õpiti, kuidas viia läbi kasutajateste.

Samuti tutvuti string diagrammide taustaga ja nende kasutusvõimalustega. Leiti, et läbi string diagrammide koodi visualiseerimine aitab kaasa algoritmide ja koodi struktuuri mõistmisele, kuid koodi kirjutamine läbi string diagrammide realiseeritud viisil pole optimaalne ja ei paku programmeerijale lihtsamat lahendust programmeerimiseks.

Lisaks õpiti töö käigus töölauarakenduse arendust kasutades Pythonit. Tutvuti töölauarakenduse arenduse eripäradega ja tugevdati ühiktestide kirjutamise oskusi keeruliste rakenduste jaoks.

Kasuks oleks tulnud põhjalikum tutvumine kasutatavate teekide dokumentatsiooniga, et kohe alguses luua optimaalseid lahendusi. Samuti leiti, et parem oleks olnud viia läbi kasutajatestimist erinevates arendusfaasides, et saada rohkem tagasisidet ning leida varem probleemseid kohti rakenduses.

6 Kokkuvõte

Käesoleva lõputöö eesmärk oli arendada edasi string diagrammide redaktorit, mida saaks kasutada erinevates valdkondades, kus on vaja string diagramme. Lõputöö põhieesmärkideks oli parandada kasutajakogemust, võimaldada string diagrammist TikZ koodi genereerimist, et diagramme saaks kasutada LaTeX failides ning uurida, kas graafiline ülevaade teeb koodi mõistmise ja kirjutamise intuitiivsemaks. Lisaks neile oli eesmärgiks veel tagurpidi ühilduvuse säilitamine, pseudonotatsiooni säilitamine ja paindlikkuse saavutamine, mis võimaldaks määrata reegleid, kuidas diagramme saab joonestada ja tõlgendada.

Lõputöö eesmärkide saavutamiseks analüüsiti teisi sarnaseid rakendusi, et leida, kuidas on varasemalt lahendatud sarnaseid probleeme, ning autoritele antud MVP rakendust, et leida rakenduses olevaid probleeme. Analüüsi tulemusel vaadati, kuidas teised rakendused on sarnaseid probleeme lahendanud ning kasutajakogemust paremaks teinud. Arenduse käigus loodi ka võimalused rakenduses programmeerimiseks. Valminud lahendus on töölauarakendus, kirjutatud Python programmeerimiskeeles.

Rakendusel viidi läbi kasutajatestid inimeste hulgas, kes ei olnud varem rakendusega tutvunud. Testimise tulemusel selgus, et arendustöö käigus lisatud funktsioone kasutatakse intuitiivselt ning samuti tekib diagrammide loomisel vähem probleeme võrreldes eelneva versiooniga. Tagasisides kiideti arendustöö käigus lisatud funktsioone, millest kõige enam toodi esile otsingu funktsiooni.

TikZ genereerimine sai rakendusse lisatud ning seda on demonstreeritud dokumendi sees olevate diagrammidega.

Lõputöö käigus uuriti ka graafilise ülevaate mõju programmeerimisele. Uuringu läbi viimiseks arendati rakendusse selleks vajaminevad süsteemid, nagu koodi redaktor rakenduse sees. Selle uurimiseks lahendati programmeerimisülesanne läbi rakenduse. Uurimise käigus leiti, et graafiline ülevaade aitab palju kaasa koodi mõistmisele ja algoritmide struktuuri efektiivsele loomisele, kuid programmeerimine võtab rohkem aega ning vigu on olemas-

olevas koodis raskem parandada. Samuti leiti, et graafiline ülevaade aitab ainult mingil määral lihtsustamisele kaasa ning string diagrammidega pole võimalik hästi näidata koodi, mis kasutab tsükleid. Lisaks leiti, et see sobib hästi ainult algoritmidele, kus kasutatakse palju abimeetodeid.

Lõputöö käigus suudeti kõik põhieesmärgid saavutada. Alameesmärkidest jäi paindlikkuse saavutamine osaliselt täitmata. Lisati juurde võimalusi, kuidas saab diagramme joonestada ja tõlgendada, kuid kasutajal endal ei ole seda võimalik muuta.

Lõputöö käigus loodud rakenduse lähtekood on saadaval GitHub'i hoidlas: `https://github.com/madzot/ivaldi-diagram-editor`

Rakendusel on veel võimalusi edasi arendamiseks. See sisaldaks näiteks rohkemate klahvifunktsioonide lisamist, automaatset salvestamist ja veel rakendusse sisseehitatud juhendite loomist.

Kasutatud kirjandus

[1] Kenji Nakahira. „Diagrammatic category theory“. *arXiv preprint arXiv:2307.08891* (2023). URL: https://arxiv.org/abs/2307.08891.

[2] Robin Piedeleu ja Fabio Zanasi. *An Introduction to String Diagrams for Computer Scientists*. Elements in Applied Category Theory. Cambridge University Press, 2025.

[3] Paweł Sobociński. *Why string diagrams? | Graphical Linear Algebra*. Accessed: 2025-05-04. URL: https://graphicallinearalgebra.net/2017/04/24/why-string-diagrams/.

[4] Ross Duncan *et al.* „Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus“. *Quantum* 4 (juuni 2020), lk. 279. ISSN: 2521-327X. DOI: 10.22331/q-2020-06-04-279. URL: https://doi.org/10.22331/q-2020-06-04-279.

[5] Tao Gu. „Categorical modelling of logic programming: coalgebra, functorial semantics, string diagrams“. Doktoritöö. UCL (University College London), 2023.

[6] Dusko Pavlovic. *Programs as diagrams: From categorical computability to computable categories*. Springer Nature, 2023.

[7] Mihkel Uukkivi. *KASUTAJAKESKNE VEEBIDISAIN: ÕPPEVAHENDI LOOMINE JA KASUTAJAKESKSUSE TESTIMINE*. Accessed: 2025-05-08. URL: https://minitorn.cs.tlu.ee/instituut/opilaste_tood/magistri_tood/2006_kevad/Mihkel_Uukkivi/Mihkel_Uukkivi_Magistri_Too.pdf.

[8] Ali Mohamed Ali Mohamed. „Assessment of Usability of Web-Based Applications' User Interfaces“. Doktoritöö. American University of Sharjah, 2017. URL: https://repository.aus.edu/server/api/core/bitstreams/adb943a4-01c8-40a2-9669-7e60a1232679/content.

[9] Jakob Nielsen. „Enhancing the explanatory power of usability heuristics“. Teoses: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '94. Boston, Massachusetts, USA: Association for Computing Machinery, 1994, lk. 152–158. ISBN: 0897916506. DOI: 10.1145/191666.191729. URL: https://doi.org/10.1145/191666.191729.

[10] Arie Deursen, Paul Klint ja Joost Visser. „Domain-Specific Languages“. *ACM SIGPLAN Notices* 35 (juuni 2000), lk. 26–36. DOI: 10.1145/352029.352035.

[11] Paul Hudak. „Domain-specific languages“. *Handbook of programming languages* 3.39-60 (1997), lk. 21. URL: https://ncatlab.org/nlab/files/Hudak-DSLs.pdf.

[12] Martin Fowler. *Domain-specific languages*. Pearson Education, 2010.

[13] Paul Wilson. *cartographer-string-diagrammatic-reasoning*. Accessed: 2025-05-02. URL: https://github.com/statusfailed/cartographer-string-diagrammatic-reasoning.

[14] Paweł Sobociński, Paul Wilson ja Fabio Zanasi. *CARTOGRAPHER: a tool for string diagrammatic reasoning*. Accessed: 2025-05-02. URL: https://cartographer.id/cartographer-calco-2019.pdf.

[15] Paul Wilson. *CARTOGRAPHER*. Accessed: 2025-06-04. URL: https://cartographer.id/.

[16] TikZiT arendajad. *tikzit*. Accessed: 2025-05-04. URL: https://github.com/tikzit/tikzit.

[17] Aleks Kissinger ja contributors. *TikZiT*. Accessed: 2025-05-04. URL: https://tikzit.github.io/.

[18] JGraph Ltd. *Diagrams.net*. Accessed: 2025-05-04. URL: https://www.drawio.com/features.

[19] JGraph Ltd. *Diagrams.net*. Accessed: 2025-06-04. URL: https://www.drawio.com/.

[20] Till Tantau. „Graph Drawing in TikZ“. Teoses: *Graph Drawing*. Toim. Walter Didimo ja Maurizio Patrignani. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, lk. 517–528. ISBN: 978-3-642-36763-2.

[21] Andrew Mertz ja William Slough. „A TikZ tutorial: Generating graphics in the spirit of TEX“. *TUGboat* 30.2 (2009), lk. 214–226.

[22] Benedek Dévényi. *Chlorophyll*. Accessed: 2025-05-05. URL: https://github.com/rdbende/chlorophyll.

[23] Stephen D Evans. *rotate() in Tkinter*. Accessed: 2025-05-13. URL: https://python-list.python.narkive.com/nhERybeW/rotate-in-tkinter#post2.

[24] Robin Lorenz ja Sean Tull. *Causal models in string diagrams*. 2023. arXiv: 2304.07638 [cs.LO]. URL: https://arxiv.org/abs/2304.07638.

[25] Paweł Sobociński. *6. Crema di Mascarpone and Diagrammatic Reasoning | Graphical Linear Algebra*. Accessed: 2025-05-15. URL: https://graphicallinearalgebra.net/2015/05/06/crema-di-mascarpone-rules-of-the-game-part-2-and-diagrammatic-reasoning/.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

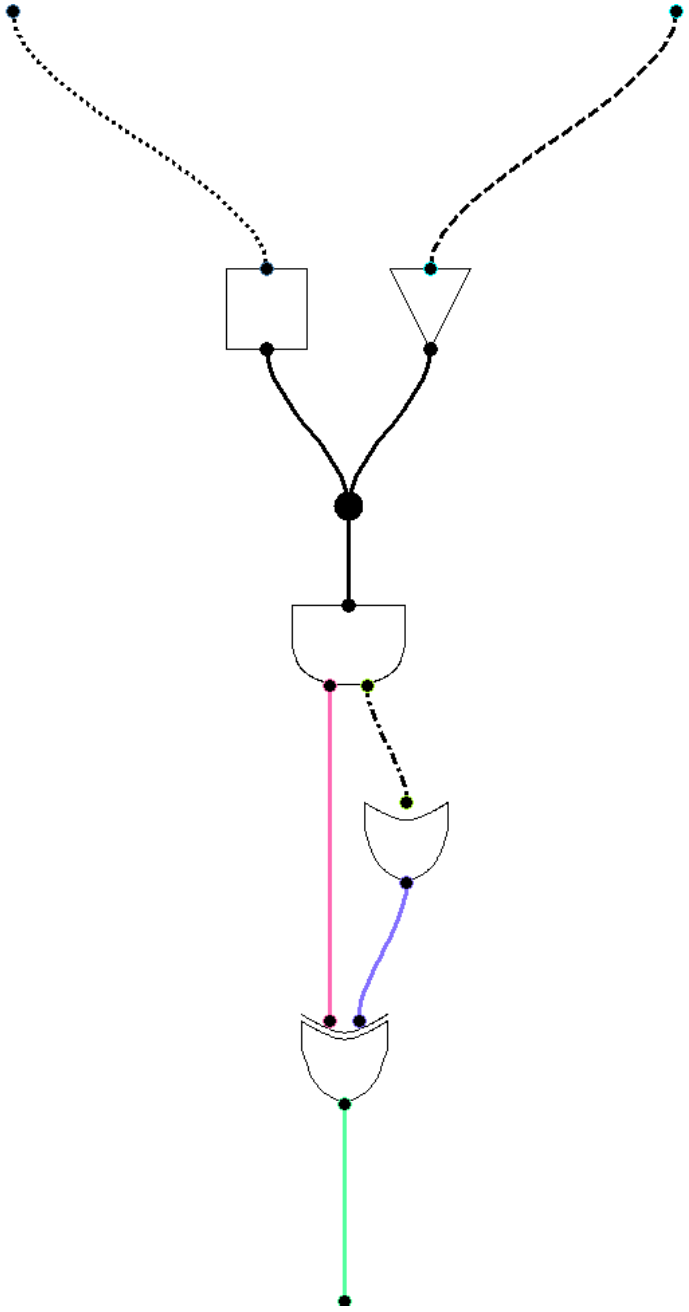
Meie, Mark Dzotsenidze ja Markus Vrager

1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Stringdiagrammi redaktori kasutajaliidese arendamine domeeni spetsiifiliste keelte kasutuseks", mille juhendajad on Paweł Maria Sobociński ja Anton Osvald Kuusk
    1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
    1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Oleme teadlikud, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autoritele.
3. Kinnitame, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

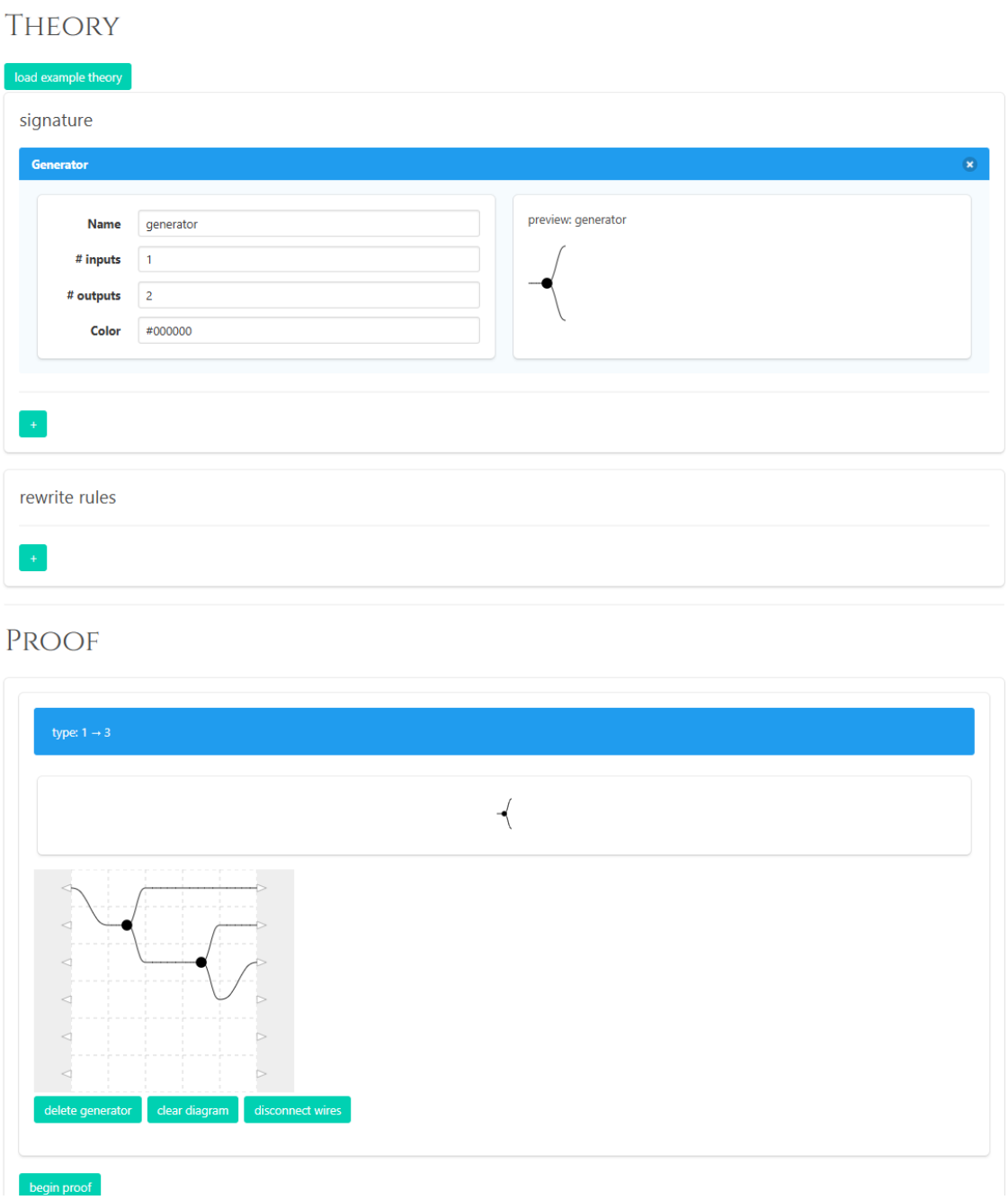
06.06.2025

[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Kastide ja juhtme tüüpide näide

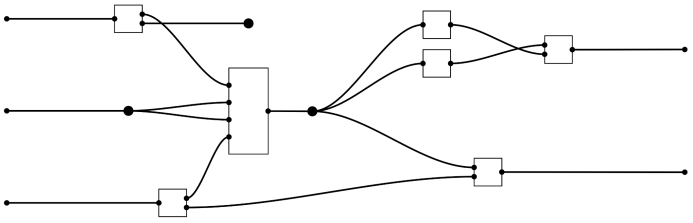


Lisa 3 – Cartographer'i kasutajaliides



Lisa 4 – Kasutaja testülesanded

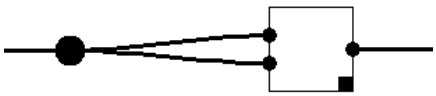
1. Looge all oleval pildil diagramm rakenduses.



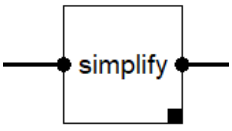
   Salvestage loodud diagram pildi (PNG) vormis.

2. Importige rakendusse diagramm nimega “test_case_2” ning minge alam-diagrammi nimega “implicate” ja kustutage kastid nimega “remove”, samuti eemaldage alam-diagrammi väljundid (output) ning salvestage projekt nimega “test_case_2_done”

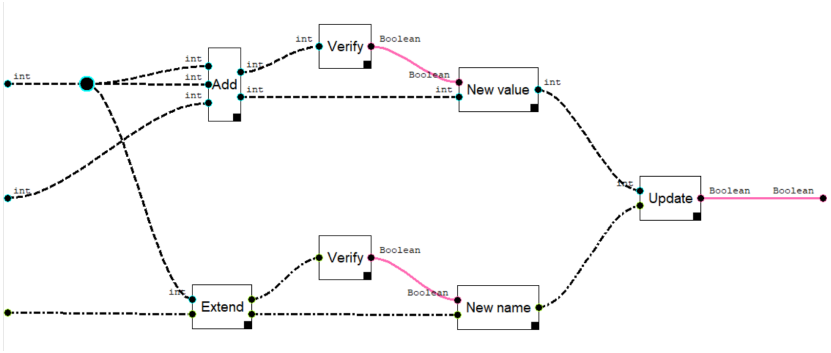
3. Importige rakendusse diagram nimega “test_case_3” ning otsige diagrammist üles kõik sellised kohad:



   Teisisõnu siis kohad, kus on üks ring (spider) ühendatud kastiga, kus on kaks sisendit ja üks väljund, mõlemad kasti sisendid peavad olema ühendatud ringiga. Juhtmete kuju ei pea sama olema. Seejärel leitud osad asendage allolevaga.

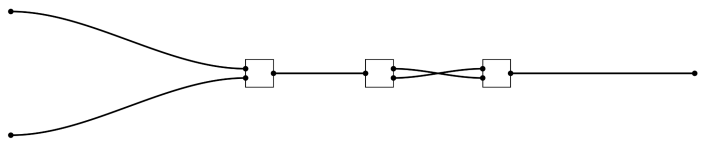


4. Looge alloleval pildil diagramm rakenduses



Juhtme stiilid ei pea vastama täpselt pildid toodutele näidetele, küll aga peavad olema erinevat tüüpi ning hoidma samasid nimesid.
Genereerige valmis diagram TikZ formaadis.

5. Looge pildil olev diagramm.



Lisage vasakult paremale minnes kastidesse järgmised funktsioonid ning geneerige kood.

```
def addition(x, y):
    return x + y
```

```
def copy(z):
    return z, z
```

```
def subtraction(x, y):
    return x - y
```