KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Aistė Vaškevičienė

Matematinių išraiškų pertvarkymo programinė įranga. Nuoseklieji algoritmai

Magistro darbas

Darbo vadovas: doc. dr. Romas Marcinkevičius

KAUNAS, 2004

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Matematinių išraiškų pertvarkymo programinė įranga. Nuoseklieji algoritmai

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių kalbos katedros lektorė
dr. J. Mikelionienė

Darbo vadovas
Programų inžinerijos katedros docentas
dr. R. Marcinkevičius

Recenzentas
Kompiuterių katedros docentas
dr. S. Maciulevičius

Atliko
IFM-8/2 grupės studentė
A. Vaškevičienė

KAUNAS, 2004

TURINYS

Santrauka

Šis projektas – tai matematinių išraiškų pertvarkymo programinės įrangos gyvavimo ciklas. Galutinis produktas – tai sistema, kuri funkcionuoja integruodamasi su Maple ir MPI sistemomis. Maple – tai universalus matematinis programinis produktas [2], [3]. Ši sistema reikalinga patogiai įvesti ir išvesti duomenis, panaudoti standartines funkcijas. MPI – tai paskirstytosios procesorių sistemos valdymo servisas [6], [7]. Šis servisas reikalingas reikalavimuose apibrėžtiems pertvarkymams lygiagrečiai vykdyti naudojant kelis vieno procesoriaus kompiuterius.

Projekto tikslas – suprojektuoti, realizuoti ir atiduoti užsakovui naudoti sukurtą produktą vietoj dabartinių lėtesnių sprendimų. Būtent greitis yra pagrindinis šio produkto nefunkcinis reikalavimas. Dabartiniai sprendimai egzistuoja, tačiau jų laikiniai parametrai netenkina užsakovų. Dauguma tokių sprendimų yra universali matematinė programinė įranga, kuri naudoja vartotojui nesunkiai prieinamą vieną kompiuterio procesorių. Daugiaprocesorinės sistemos yra labai brangios. Mūsų sukurtas produktas yra specializuotas, dėl to jame yra greitesnis pats pertvarkymo algoritmas. Be to, mūsų produktas naudoja už daugiaprocesorinę sistemą daug pigesnį variantą – vieno procesoriaus kompiuterių, sujungtų į vietinį tinklą, procesorius. Lygiagrečiai (naudojant visus procesorius) atliekamas matematinės išraiškos pertvarkymas ypač sumažina bendrą uždavinio sprendimo laiką.

Kadangi resursai buvo labai riboti, sistemos realizacija užtuko 16 mėnesių. Per šį laiką buvo atlikta poreikio analizė, surinkti ir specifikuoti reikalavimai, suprojektuota sistemos architektūra, detaliai suprojektuoti sistemos komponentai, sistema realizuota ir ištestuota, įdiegta pas užsakovą. Projektas buvo pilnai dokumentuotas ir turi tokius dokumentus kaip projekto paraiška, projekto planas, reikalavimų specifikacija, architektūros specifikacija, detalios architektūros specifikacija, testavimo planas, testavimo ataskaita, diegimo ataskaita. Produktas taip pat dokumentuotas ir turi vartotojo dokumentaciją, reklaminę medžiagą, interneto svetainę.

Šiame dokumente detaliai nagrinėjama taikomoji sritis, analizuojami esami ir pasirinkti sprendimai bei algoritmai, pateikiami pagrindiniai produkto projektavimo ir realizavimo etapų aspektai. Pagrindinę dokumento dalį sudaro atliktas tyrimas. Tyrimo metu mūsų sistemos matematinių išraiškų pertvarkymams sugaištas laikas buvo lyginamas su kitomis sistemomis. Taip pat buvo analizuojama kaip pertvarkymų laiką įtakoja sistemos charakteristikos, pvz. matematinės išraiškos ilgis ar naudojamų procesorių kiekis. Tyrimų rezultatai pateikti lentelėmis ir grafikais, pagal tai tendencingai padarytos išvados.

Summary

This project is a life cycle of the software for reformation of the mathematical expressions. A final product is a system, which operate integrated with Maple and MPI systems. Maple is universal mathematical software product [2], [3]. This system is required to handily input and output data, to have ability using standard functions. MPI is a service for controlling the distributed system [6], [7]. This service is required for parallel reformations of the mathematical expressions using several single-processor computers.

The project goal is to design, implement and deliver to customer for usage developed product instead of current slower solutions. Exactly speed is the key non-functional requirement for the product. Current solutions exist, but the speed is unacceptable for the customer. Most solutions are universal mathematical software, which uses one single-processor computer, which is obtainable for most users. Multi-processor systems are very expensive. The product we developed is specialized, so it has high-speed reformation algorithm. Besides, our product uses cheaper solution then multi-processor system – it uses processors of intranet single-processor computers. Parallel (using all processors) performing the reformation of mathematical expressions especially reduces the total time of decision of the problem.

We have been developing the system for 16 months, because the resources were limited. We have done demand analysis, collected and specified requirements, designed system architecture and components, implemented the system, tested it and installed it for the customer while this period. The all project was documented by writing such documents as project order, project schedule, requirement specification, architecture specification, detail architecture specification, test schedule, test report, install report. The product also was documented and has user guide, publicity and Internet site.

We have deeply researched environment, analyzed present and selected solutions and algorithms, reported key aspects of product design and implementation stages in this document. The main part of the document consists of accomplished research. The time taken for reformation of mathematical expressions in our system was compared with the taken time in other systems during the research. It was also analyzed how the system characteristics, like length of the mathematical expression or number of processors, influence the reformation time. Research results are presented in tables and diagrams, accordingly figured tendentious conclusions.

1. ĮVADAS

Šiame skyriuje trumpai apžvelgiamas visas magistro baigiamasis darbas: apibrėžiama taikymo sritis, nustatomi darbo tikslai, trumpai aprašomas temos aktualumas, nurodoma taikoma metodika, apibrėžiamas tiriamasis objektas, pristatoma teorinė literatūra, nustatoma analizės tvarka, nurodoma dokumento struktūra.

1.1. Taikymo sritis

Taikomoji sritis – labai didelių (ilgų) matematinių išraiškų pertvarkymas. Tai labai konkreti ir sudėtinga programinės įrangos taikymo sritis. Labai didelėms matematinėms išraiškoms pertvarkyti naudojami daugiaprocesoriniai kompiuteriai arba vieno procesoriaus kompiuterių tinklai. Tokiose sistemose skaičiavimai yra suskaidomi į atskirus procesus ir vykdomi lygiagrečiai. Didelės projektavimo ir programavimo išlaidos bei santykinai maža paklausa lemia ypatingai dideles tokių sistemų kainas. Dėl to šioje srityje neįmanoma suderinti pigumo ir efektyvumo, todėl paprastesniems skaičiavimams efektyvesne programine įranga galime laikyti universalias matematines programas, kurios pasižymi ypač plačiu funkcionalumu, vidutiniu greičiu ir vis dar aukšta kaina. Dažniausiai naudojami šie universalūs matematiniai produktai: Mathcad, Maple, MatLab.

Tokioje taikomojoje srityje dirba matematikos, fizikos, chemijos ir kt. mokslininkų grupės. Taigi ir produktas yra orientuotas į šių mokslinių sričių specialistus. Savo produktą Lietuvoje galėtume siūlyti aukštųjų mokyklų mokslininkams, institutų ir laboratorijų specialistams. Jį galėtume publikuoti ir internete, kad apie produktą sužinotų ir užsienio mokslininkai bei specialistai.

Praktikoje didelės matematinės išraiškos yra gaunamos sprendžiant didesnes diferencialinių lygčių sistemas, skaičiuojant aukštos eilės diferencialą.

1.2. Projekto tikslai

1.2.1. Projekto tikslas ir adresatas

Projektas tikslas – sukurti numatytą produktą ir platinti jį tarp vartotojų, dirbančių taikomojoje srityje. Šiuo atveju, taikomoji sritis – tai matematinių išraiškų pertvarkymas. Šioje srityje dirba matematikos, fizikos, chemijos ir kt. mokslininkų grupės. Todėl ir produktas yra skirtas šių mokslo sričių specialistams.

1.2.2. Produkto užsakovas

Produkto užsakovas – Kauno technologijos universiteto Informatikos fakulteto Programų inžinerijos katedros docentas dr. Romas Marcinkevičius. Užsakovas dažnai naudoja Maple sistemą ir yra suinteresuotas naudoti naująją specializuotą sistemą, kuri sukurta siekiant padidinti pertvarkymų greitį. Projekto eigoje visos autorinės teisės priklausė sistemos kūrėjams, o galutinio produkto autorinės teisės buvo perleistos ir priklauso užsakovui.

Kai produktas jau realizuotas, galimi ir kiti sistemos vartotojai, numatyti 1.2.1. skyrelyje.

1.3. Temos aktualumas

1.3.1. Problemos sprendimas pasaulyje

Matematinių išraiškų pertvarkymas – labai konkreti ir sudėtinga programinės įrangos taikymo sritis. Didelės projektavimo ir programavimo išlaidos bei santykinai maža paklausa lemia ypatingai dideles tokių sistemų kainas. Šioje srityje neįmanoma suderinti pigumo ir efektyvumo.

Išsiskiria tokios produktų klasės:

- universali matematinė programinė įranga: pritaikyta įvairioms platformoms, turi platų funkcionalumą, integralumą, patrauklią vartotojo sąsają;
- specializuota matematinė programinė įranga: pritaikyta konkrečiai taikymo sričiai, turi ribotą funkcionalumą, dažnai sudėtingą vartotojo sąsają.

Pasaulyje geriausiai žinomi matematinių produktų kūrėjai yra MathSoft, MathWorks, Waterloo Maple kompanijos. Jų produktai gerai žinomi visame pasaulyje. Tokių kompanijų produktai yra labai universalūs, turi patogias grafines sąsajas, puikias integracines priemones. Žinomiausi pasaulyje produktai yra Mathcad, MatLab, Mathematica, Maple, IDL, Gauss, LabView [1].

Nagrinėjamoje probleminėje srityje paskirstytos sistemos naudojamos gana retai, kadangi kiekvienas matematinio uždavinio tipas reikalauja vis kitokio procesorių valdymo mechanizmo. Kompanija Risc-Linz 1998 m. pradėjo kurti tokių mechanizmų aibę ir pavadino ją Distributed Maple [2].

1.3.2. Problemos sprendimas Lietuvoje

Peržiūrėjus prieinamus literatūros šaltinius, mums nepavyko rasti tokio Lietuvos programinės įrangos gamintojo, kuris būtų sukūręs ar kurtų universalią matematinę programinę įrangą. Tačiau Lietuvoje yra kuriami specializuotos paskirties užsakomieji projektai, kurie dažniausiai vis tiek susiveda į komplikuotos, daug resursų reikalaujančios programinės įrangos kūrimą.

1.4. Metodologija

1.4.1. Projekto vykdymo metodika

Visi projekto etapai buvo vykdomi pagal Programų inžinerijos standartus ir dėstytojų rekomendacijas. Projekto eigoje buvo išklausyti tokie naudingi moduliai kaip Programų inžinerijos procesai, Programų inžinerijos valdymas, Reikalavimų specifikavimas, Programų projektavimas, Duomenų projektavimas, Programų sistemos architektūros analizė, Programų testavimo metodai, Programinės įrangos įdiegimo tyrimas, Programinės įrangos kokybė, Programinės įrangos įrankiai, Informacinės technologijos projektavimo vadyboje. Vykdant projektą buvo griežtai atsižvelgta į šių modulių teorinę medžiagą, atsakingų dėstytojų rekomendacijas.

1.4.2. Produkto kūrimo technologijos

Projektavimo etapuose buvo naudojama UML projektavimo technologija. Realizavimo etape buvo naudojamas objektinis programavimas, C++ programavimo kalba ir šios kalbos kompiliatorius Linux operacinei sistemai. Testavimo vykdymui buvo naudojami specialiai pritaikyti automatinio testavimo įrankiai.

1.5. Analizės tvarka

Pirmiausia reikia detaliai aprašyti inžinerinę problemą. Problemai spręsti būtina išnagrinėti susijusią literatūrą ir esamus sprendimus. Tokiu būdu įsigilinus į problemą, galime išsikelti sau detalius darbo tikslus, parinkti ir pagrįsti tuos tikslus įgyvendinančius sprendimus. Sprendimams realizuoti reikia suprojektuoti ir realizuoti tinkamos architektūros bei greitų ir patikimų algoritmų visumos sistemą.

1.6. Tyrimo objektas ir tikslas

Teoriniuose ir eksperimentiniuose tyrimuose bus tiriamas sukurtas produktas. Tyrimo tikslas – ištirti matematinių išraiškų pertvarkymo greitį. Bus atlikti tyrimai, kokią įtaką matematinių išraiškų pertvarkymo greičiui daro įvairios sistemos ir duomenų charakteristikos, pvz. sistemos procesorių kiekis, lygiagrečių procesų kiekis, matematinės išraiškos ilgis ir sudėtingumas ir pan. Tuo pačiu bus atlikti pertvarkymų greičio palyginimai su kitais rinkoje esančiais produktais. Pagal tyrimų rezultatus bus padarytos pagrindinės išvados, nustatyta vizija sistemos plėtrai, pateiktos rekomendacijos naudotojams.

1.7. Dokumento struktūra

Magistro baigiamasis darbas susideda iš 6 esminių dalių: Įvadas, Analizė, Projektavimas, Tyrimas, Išvados ir priedai. Įvade bendrais bruožais apibendrinamas visas magistro darbas ir šio darbo dokumentas. Antroje dalyje analizuojama problema, nagrinėjama susijusi literatūra ir esami sprendimai, nustatomi detalūs darbo tikslai, parenkami ir pagrindžiami tikslus įgyvendinantys sprendimai ir algoritmai. Projektavimo dalyje išdėstomi visų produkto projektavimo etapų dokumentai: pirminis produkto apibūdinimas, reikalavimų specifikacija, architektūros specifikacija ir detalios architektūros specifikacija. Taip pat šioje dalyje detalizuojama produkto programinė realizacija. Tyrimo dalyje apibrėžiamos tyrimo aplinkybės, išdėstomi atlikti eksperimentiniai tyrimai, įvertinami rezultatai. Penktoje dalyje pagal tyrimų rezultatus nustatomos ir išdėstomos darbo išvados, apibrėžiamos galimos produkto plėtros kryptys, pateikiamos rekomendacijos naudotojams. Šeštoji dalis susideda iš kelių skyrių: literatūros, santrumpų ir terminų žodynų bei dokumento priedų.

Šis dokumentas parengtas pagal bendruosius nurodymus baigiamiesiems darbams, dokumento struktūra sudaryta pagal Programų inžinerijos katedros rekomendacijas.

2. ANALIZĖ

Šiame skyriuje detaliai aprašoma sprendžiama problema, nagrinėjama susijusi literatūra ir esami sprendimai, iškeliami detalūs darbo tikslai, parenkami ir pagrindžiami tikslus įgyvendinantys sprendimai bei algoritmai.

2.1. Problemos aprašymas

Užsakovas iki šiol matematinėms išraiškoms pertvarkyti naudojo įvairius programinius paketus, tačiau sprendimų laikiniai parametrai užsakovo netenkino. Dauguma tokių sprendimų yra universali matematinė programinė įranga, kuri naudoja vartotojui nesunkiai prieinamą vieną kompiuterio procesorių. Daugiaprocesorinės sistemos yra labai brangios. Turėjo būti sukurta specializuota sistema, dėl to joje turėjo būti greitesnis pats pertvarkymo algoritmas. Be to, sistema turėjo naudoti už daugiaprocesorinę sistemą daug pigesnį variantą: vieno procesoriaus kompiuterių, sujungtų į vietinį tinklą, procesorius. Lygiagrečiai atliekamas matematinės išraiškos pertvarkymas taip pat turėjo sumažinti bendrą uždavinio sprendimo laiką.

Reikia tokios sistemos, kuri pakeistų šiuo metu užsakovo naudojamus programinius paketus ir būtų greitesnė. Greitis yra pagrindinis šios sistemos nefunkcinis reikalavimas. Pagrindinis funkcinis reikalavimas – matematinių išraiškų pertvarkymas.Taigi, apibendrinant, mūsų galutinis tikslas buvo sukurti greitą ir patikimą matematinių išraiškų pertvarkymo sistemą. Todėl turėjome tokias pagrindines problemas: diferencijavimo laiko minimizavimas bei patikimumo užtikrinimas.

2.1.1. Nagrinėjama problema

Kuriamai sistemai reikėjo sukurti matematinių išraiškų pertvarkymo modulį, kuris greitai ir patikimai mokėtų diferencijuoti matematines išraiškas. Sistema turi turėti itin paprasta sąsają, kuria galima būtų paduoti pradinę matematinę išraišką ir kuria galima būti pasiimti rezultatinę matematinę išraišką.

Reikia detaliai išnagrinėti matematinių išraiškų analizės ir nagrinėjimo metodikas, esamus sprendimus ir priimti tinkamus sprendimus kuriamai sistemai. Analizuojant reikia neužmiršti ir kitų tos pačios klasės uždavinių, pvz. teksto analizavimas, gramatinis nagrinėjimas, leksikos ir sintaksės tikrinimas, kompiliatoriai.

2.2. Problemos analizė

Problemai spręsti pirmiausia reikia detaliai išanalizuoti susijusią literatūrą bei esamus sprendimus.

2.2.1. Įžanga

Nagrinėjama problema apima tokias koncepcijas kaip gramatinis nagrinėjimas, simbolių atpažinimas, leksikos analizė, mazgų medžiai ir lentelės.

Pirmasis matematinių išraiškų analizės algoritmą 1975 m. sumodeliavo Robert M. Graham. Matematinėse išraiškose buvo naudojami keturi aritmetiniai operandai: daugyba, dalyba, suma ir atimtis. Matematikos principais operandų simboliams buvo priskirti prioritetai: aukščiausi prioritetai daugybai bei dalybai, žemiausi sumai ir skirtumui. Matematinės išraiškos galėjo turėti ir skliaustelius. Iš esmės, algoritmo taisyklės atitiko algebros, kurią mokėmės dar mokykloje, taisykles, pavyzdžiui daugyba (arba dalyba) skaičiuojama prieš sumavimą ar atimtį; pradėti reikia nuo vidinių skliaustelių ir eiti išorės link; jeigu gretimi operandai turi vienodą prioritetą, tai pirmiau vykdomas pirmesnis (kairesnis).

Šio autoriaus algoritmas buvo pagrindas daugeliui kitų iki šiol sukurtų ir kuriamų algoritmų. Panagrinėkime Vajoningo profesoriaus Bill Murray algoritmą, kuris pirmiausia išplėtė algoritmą papildydamas jį eksponentės operandu bei keliomis funkcijomis (sin, cos, ir pan.). Taip pat jis įvedė konstantas ir kintamuosius, kuriuos galima apibrėžti vardais. Šie elementai matematinėje išraiškoje yra atpažįstami leksikos analizės fazėje. Dar vėliau algoritmas buvo papildytas ir vektorių bei matricų palaikymu, bet mes šito nenagrinėsime.

2.2.2. Gramatinis nagrinėjimas

Gramatinis nagrinėjimas (*angl*. Parser) – tai procesas, kai tekstas yra sudalomas į dalis, kiekvienai daliai priskiriant įvairius atributus ir, svarbiausia, nustatant ryšius su kitomis dalimis. Pavyzdžiui jeigu tai būtų šnekamosios kalbos sakinys, galėtume jį padalinti į žodžius, priskiriant jiems tokius atributus kaip kalbos dalis, sakinio dalis, laikas, giminė ir pan. Tokiu būdu mes detalizuojame, labiau suprantame gramatiškai išnagrinėtą sakinį. Iš esmės, gramatinis nagrinėjimas yra struktūros atpažinimas ir suvokimas.

Matematinių išraiškų gramatinis nagrinėjimas – tai algoritmas, kuris identifikuoja matematinio reiškinio struktūrą ir sudedamąsias dalis. Išraiška, susidedanti iš daugybės simbolių, tarp kurių yra skaičiai, raidės, žodžiai, specialūs simboliai ir t. t. Gramatinio nagrinėjimo tikslas yra atpažinti minėtus elementus ir transformuoti juos į priimtus žymėjimus, pvz. simbolį “+” transformuoti į sumos ženklą, simbolius “-4,03” transformuoti į realų neigiamą skaičių -4,03 ir pan. Be to, visi transformuoti elementai turi būti susieti tarpusavyje ir sudėti į tinkamą duomenų struktūrą.

2.2.3. Leksikos analizė

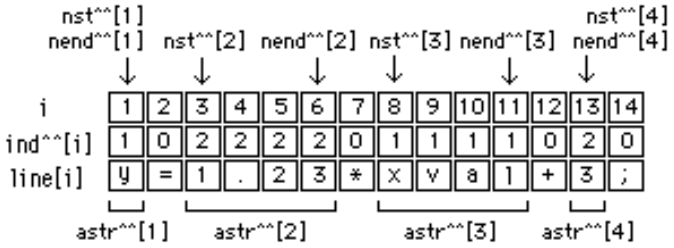
Leksikos analizės procedūra yra nagrinėjimo proceso pagrindas. Jis identifikuoja pagrindinius elementus matematinėje išraiškoje, pakeičia juos žymėjimais, nustato jų tipus, priskiria

prioritetus. Operandų ir jų dėmenų pozicijos atitinkamai nustatomos ir sintaksinėje struktūroje. Priklausomai nuo pozicijų, organizuojamas ir operandų hierarchiškumas, kas nusako operacijų vykdymo seką.

Nagrinėsime algoritmą, pritaikytą Pascal kalbai. Leksikos analizės įėjimu yra laikoma simbolių eilutė *line*. Algoritmo išėjime yra trys surikiuoti masyvai: žymėjimų masyvas *sy^^[i]*, žymėjimų tipų masyvas *tokentype^^[i]* ir prioritetų masyvas *pr^^[i]*, kur *i=1..ntot*.

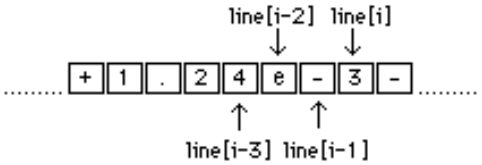
Kad simbolių eilutėje išskirti sudedamąsias dalis, naudojamas rodyklių masyvas *ind^^[i]*, kur *i=1..length(line)*. Šios rodyklės atitinka kiekvieną simbolių eilutės simbolį. Masyvas *astr^^[i]* yra skirtas simbolių grupėms (trumpesnėms simbolių eilutėms), t. y. kintamųjų, konstantų ir funkcijų vardams saugoti. Šių simbolių grupių pradžios ir pabaigos pozicijos pagrindinėje eilutėje yra saugomos atitinkamuose masyvuose *nst^^[j]* ir *nend^^[j]*.

Po to, kai apibrėžtas reikiamas kiekis atminties, visi kintamieji inicijuojami, iš išraiškos pašalinami visi tarpai. Pradedamas ciklas *for i := 1 to length(line)*. Kiekvienam eilutės simboliui, kuris priklauso alfabetui, *ind^^[i] := 1*., o kiekvienam skaitmeniniam simboliui (0..9) arba taškui *ind^^[i] := 2*. Dabar pagal masyve *ind* sudarytas 0, 1 ar 2 sekas galima nustatyti simbolių eilutės sudedamąsias dalis, greitai nustatant jų pradžios ir pabaigos pozicijas. Žemiau esanti schema iliustruoja simbolių eilutės *y = 1.23 * xval + 3* padalijimą į simbolių grupes.



pav. 1. Simbolių grupių nustatymas

Norint, kad algoritmas suprastu moksliniu formatu užrašytus skaičius, pvz. *1.24e-3*, reikia modifikuoti algoritmą. Kiekvienam *i>3* simboliui peržiūrime tris ankstesnius simbolius: *line[i-3]*, *line[i-2]* ir *line[i-1]*. Jeigu *(ind^^[i-3] = 2) and (line[i-2] in ['e', 'E']) and (line[i-1] in ['–', '+']) and (ind^^[i] = 2)*, tai *ind^^[i-2] := 2* ir *ind^^[i-1] := 2*.



pav. 2. Mokslinio formato skaičiaus interpretavimas

Toliau, simbolių grupės *astr^^[i]* yra sukabinami su operando simboliu, kad išgauti surikiuotą žymėjimų masyvą *sy^^[i]*.Nulinis žymėjimas *sy^^[0] := '@'*, kad vėliau jį galima būtų naudoti

kaip skyriklį. Šis surikiuotas žymėjimų masyvas dabar vaizduoja matematinę išraišką. Dar toliau reikia nustatyti žymėjimų tipus *tokentype^^[i]*, priklausomai nuo *sy^^[i]*. Viduje *for i :=0 to ntot do* kiekvienas žymėjimo tipas automatiškai inicijuojamas kaip *'string'*. Jeigu *(sy^^[i] in ['+', '–', '*', '/', '^", '=']) or (tokentype^^[i] = 'binary')*. Jeigu *(sy^^[i] = 'pi') or (sy^^[i][1] in ['0'..'9', '.'])*, tai *tokentype^^[i] := 'constant'*. Jeigu *sy^^[i]* yra viena iš funkcijų (pvz. sqrt, sin, cos, exp, ln ir pan.), tai *tokentype^^[i] := 'function'*. Jeigu *tokentype^^[i] = 'string'*, bet nelygi nei *'binary'*, nei *'constant'*, nei *'function'*, tada *tokentype^^[i] := 'variable'*. Galų gale, kiekvienam *i > 0*, jeigu *sy^^[i] in ['+', '–']*, ir *tokentype^^[i-1]* nėra nei kintamasis, nei skaičius, o *sy^^[i-1]* yra arba užsidarantys skliausteliai, arba kabutė, tada *tokentype^^[i] := 'unary'*.

Sekantis ciklas nuo *1* iki *ntot* patikrina kiekvieną konstantą, kuri nelygi *'pi'*, patikrinant netgi kiekvieną simbolį. Jeigu simbolis yra raidė, bet ne *'e'* ir ne *'E'*, tai generuojama klaida. Jeigu raidė vis dėl to yra *'e'* arba *'E'*, tai kairesnis simbolis būtinai turi būti skaičius, o dešinesnis – skaičius arba *'+'*, arba *'–'*. Priešingu atveju generuojama klaida. Be to, visi kiti simboliai taip pat turi būti skaičiai arba iš aibės *['e', 'E', '+', '–', '.']*. Priešingu atveju irgi generuojama klaida.

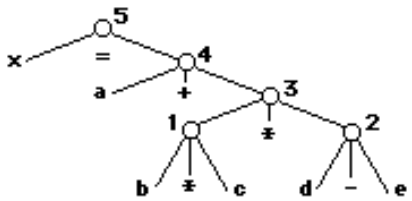
Galų gale, žymėjimams priskiriami prioritetai *pr^^[i]*.

lentelė 1. Žymėjimų prioritetai

Žymėjimas	Prioritetas
^	7
* /	6
+ –	5
=	4
) ;	3
(	2
@	1
kiti	0

2.2.4. Mazgų medžiai ir lentelės

Mazgų lentelė yra analizės algoritmų pagrindas. Medžio tipo struktūra sudaryta jungiamųjų taškų (mazgų) aibės pilnai apibrėžia matematinę išraišką. Kiekvienas įrašas, vadinamas mazgu, yra numeruojamas ir susideda iš visos būdingos informacijos. Kaip DNR molekulė, mazgų lentelė nusako kodą, atskleidžiantį matematinės išraiškos struktūrą ir sintaksę. Mazgų medis yra grafinis matematinių išraiškų atvaizdavimo būdas.



pav. 3. Mazgų medžio apibrėžta išanalizuota matematinė išraiška

Tokios duomenų struktūros, dar dažnai vadinamos operacijų medžiu, saugojimui yra struktūrą beveik atitinkanti mazgų lentelė, kuri yra numeruotas įrašų, kuriuose yra visa informacija apie mazgą ir jo ryšius, masyvas.

Po to kai matematinė išraiška buvo išanalizuota ir transformuota į mazgų lentelę, o vizualiai į mazgų medį, bet kokiems veiksmams atlikti yra naudojamas vienas bendras algoritmas – medžio perrinkimas nuo šaknies iki šakų.

2.2.5. Išvados

Išanalizavę literatūrą, vis dėl to pasinaudojome 1975 metais Robert M. Graham sumodeliuoto matematinių išraiškų analizės algoritmo idėja ir jo modifikacijomis naudojant patobulintas duomenų struktūras.

2.3. Darbo tikslai

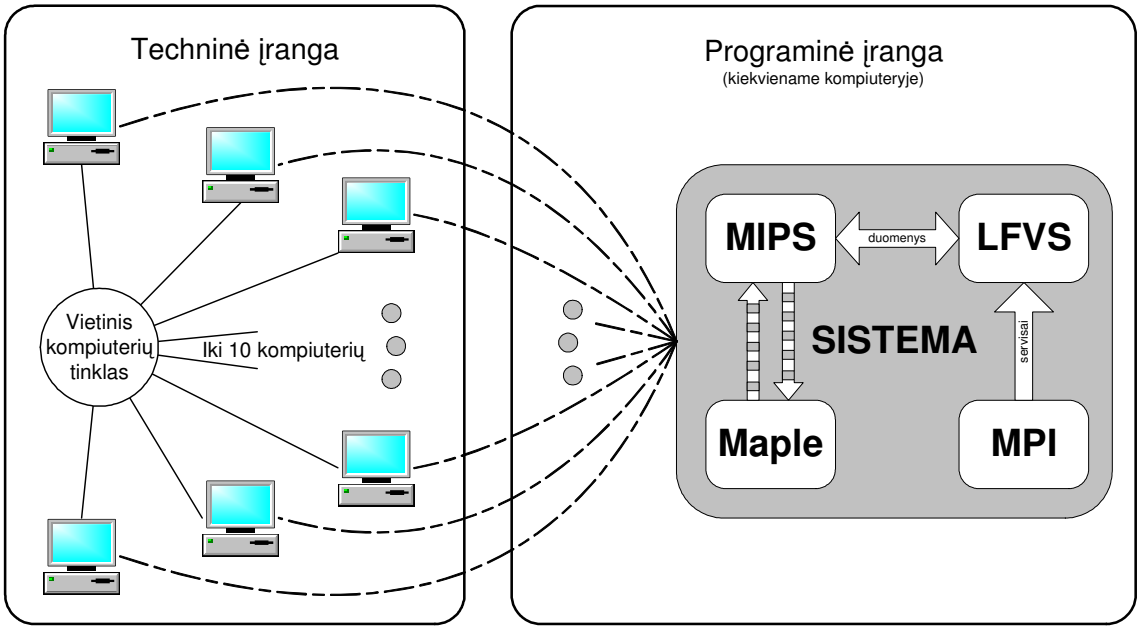
Identifikavus problemą ir išanalizavus esamus sprendimus, sistemos kūrėjų komanda išsikėlė sau darbo tikslus, kurių dalis, susijusi su nagrinėjama problema, yra išvardijama žemiau:

- suprojektuoti detalią sistemos architektūrą išskiriant reikiamas valdymo ir objektų klases;
- suprojektuoti ir realizuoti matematinių išraiškų pertvarkymo algoritmą;
- ištestuoti matematinių išraiškų pertvarkymo algoritmo patikimumą;
- integruoti matematinių išraiškų pertvarkymo algoritmą į sistemą;
- atlikti matematinių išraiškų pertvarkymo sistemos tyrimus, palyginti su kitomis sistemomis;
- įvertinti tyrimų rezultatų, įžvelgti tendencijas, padaryti išvadas, pateikti rekomendacijas.

2.4. Priimti sprendimai

Darbo tikslams pasiekti sistemos kūrėjų komanda pirmiausia turėjo priimti įvairius strateginius sprendimus, kurie numatytų sistemos veikimo principą, sistemos aplinką, programų inžinerijos metodikas, technologijas, programavimo kalbas ir pan.

2.4.1. Sistemos veikimo principas



pav. 4. Sistemos veikimo principas

Aptarsime aukščiau pavaizduotos sistemos veikimo principą. Pirmiausia, sistema turėjo mokėti tiesiog diferencijuoti matematinę išraišką. Tokios funkcijos yra paskirtos matematinių išraiškų pertvarkymo sistemai MIPS. Dar didesniam diferencijavimo greičiui užtikrinti buvo pasirinkta lygiagrečių skaičiavimų logika. Tokia logika realizuojama lygiagretaus funkcionavimo valdymo sistemoje LFVS, kuri atsakinga už matematinės išraiškos suskaldymą į smulkesnes ir paprastesnes išraiškas, jų perdavimą atskiriems MIPS skaičiavimų procesams ir, galiausiai, apskaičiuotų išraiškų atgalinį sujungimą. Procesams lygiagretinti yra naudojamas MPI servisas. Efektyvumui užtikrinti reikalingi keli procesoriai. Visa minėta programinė įranga įdiegiama į kiekvieną naudojamą kompiuterį. Vartotojas bet kuriame kompiuteryje inicijuoja matematinės išraiškos diferencijavimą. LFVS suskaido išraišką į smulkesnes ir per MPI servisą inicijuoja atskirus procesus kituose kompiuteriuose. Procesuose veikianti MIPS pagal reikalavimus pertvarko matematinę išraišką ir grąžina iniciatoriui. Grąžinti rezultatai yra sujungiami ir išvedami vartotojui.

2.4.2. Matematinių išraiškų pertvarkymo sistema

Matematinių išraiškų pertvarkymo sistema veikia kaip visai atskira posistemė, t. y. gauna duomenis, atlieka pertvarkymus ir grąžina rezultatus. Detaliau, MIPS veikia tokiais etapais:

- perteklinių simbolių pašalinimas;
- tekstinės matematinės išraiškos transformavimas į operacijų medį;
- operacijų medžio supaprastinimas;
- diferencijavimas atliekant operacijų medžio šakų pakeitimus;

- pakartotinis operacijų medžio supaprastinimas;
- operacijų medžio transformavimas į tekstinę matematinę išraišką.

Prieš atliekant bet kokius veiksmus pirmiausia yra pašalinami visi tarpo simboliai, kadangi jie nedaro jokios įtakos išraiškai. Taip pat prieš atliekant pertvarkymus tekstinė matematinė išraiška yra transformuojama į matematinės išraiškos operacijų medį. Dabar jau gali būti atliekamas diferencijavimas. Tiesa, ištyrėme, kad prieš ir po diferencijavimo naudinga atlikti operacijų medžio supaprastinimą, ekvivalentų matematinės išraiškos supaprastinimui. Galų gale gautasis operacijų medis yra vėl transformuojamas į rezultatinę tekstinę matematinę išraišką.

2.4.3. Naudojami produktai

Projekto metu dažnai buvo naudojamas Maple programinis paketas. Jis buvo pasirinktas dėl to, kad jį naudoja užsakovas. Prie šio produkto yra priderintos matematinės išraiškos vaizdavimo taisyklės. Be to, Maple paketas yra patikimas, greitas, lankstus, pritaikytas papildomų funkcijų integracijai, veikiantis Linux operacinėje sistemoje. Produkto gamintojas – Waterloo Maple, Inc.. Buvo naudojama Maple 7 versija.

3. PROJEKTAVIMAS

Šiame skyriuje išdėstomi visų produkto projektavimo etapų dokumentai: pirminis produkto apibūdinimas (projekto paraiška), reikalavimų specifikacija, architektūros specifikacija ir detalios architektūros specifikacija. Čia yra įkelti tik su magistro darbu susiję minėtų dokumentų skyriai. Taip pat šioje dalyje detalizuojama produkto programinė realizacija.

3.1. Produkto apibūdinimas

3.1.1. Programų sistemos funkcijos

Produkto esmė – matematinių išraiškų simbolinis diferencijavimas. Diferencijavimo rezultatas – nauja matematinė išraiška. Diferencijavimas yra riboto funkcionalumo, t. y. sistema turi atpažinti ir diferencijuoti tokias funkcijas:

- $f_1(x) + f_2(x)$
- $f_1(x) – f_2(x)$
- $f_1(x) \times f_2(x)$
- $(f(x))^n$
- ir jų junginius,

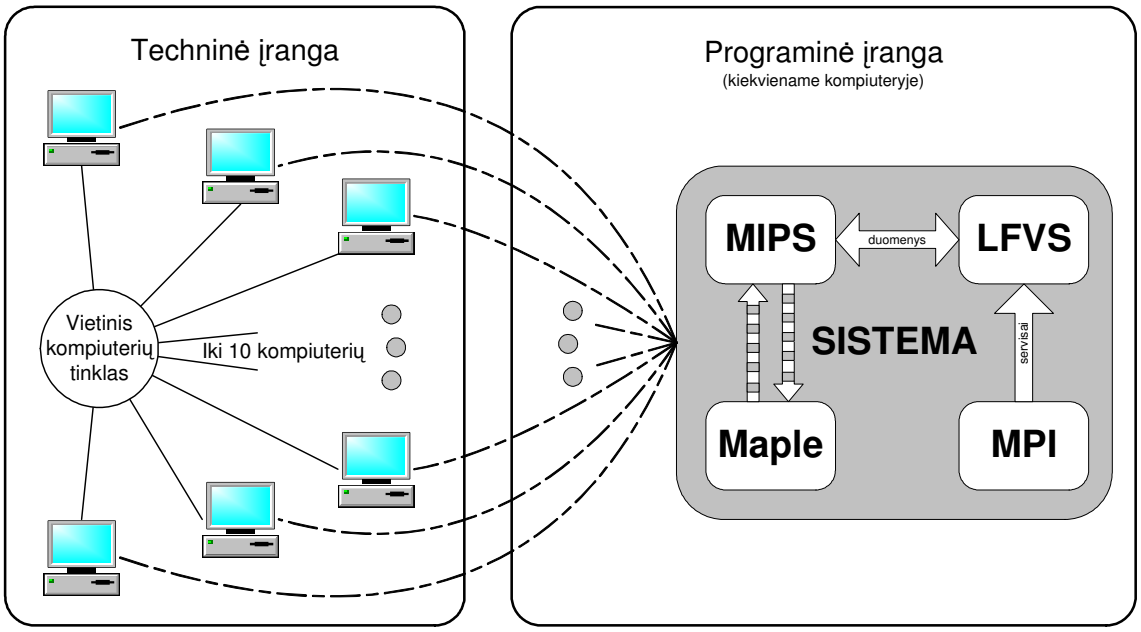
  kai $f(x)$, $f_1(x)$, $f_2(x)$ gali būti:
- $a \cdot x^n + b$
- $\sin x$
- $\cos x$
- ir jų junginiai.

Kitos diferencijavimo formulės paliekamos kitoms produkto versijoms. Taip pat ir kitokie funkcionalumai (pvz., integravimas) paliekami kitoms produkto versijoms. Taigi sistema turi būti atvira funkcionalumų papildymams.

Atliekant matematinės išraiškos diferencijavimą, išraiška turi būti kuo optimaliau skaidoma į nepriklausomas išraiškas, kurias galėtų lygiagrečiai apdoroti atskiri procesoriai. Reikia įvertinti tokios paskirstytos sistemos apribojimus ir ypatybes:

- sistemos procesoriai neturi bendros atminties;
- duomenims perduoti gaištamas papildomas laikas.

Apibendrintas sistemos veikimo principas pavaizduotas pav. 5.



pav. 5. Sistemos veikimo principas

Sistema sudaryta iš dviejų posistemių:

- MIPS – matematinių išraiškų pertvarkymo sistema;
- LFVS – lygiagretaus funkcionavimo valdymo sistema.

Matematinių išraiškų pertvarkymo sistema atsakinga už matematinių išraiškų reikalavimuose nurodytų pertvarkymų atlikimą. Ši posistemė gali naudotis Maple paslaugomis. Lygiagretų matematinių išraiškų pertvarkymo vykdymą organizuoja lygiagretaus funkcionavimo valdymo sistema. Ši posistemė, naudodamasi paskirstytosios sistemos valdymo terpės MPI servisais, manipuliuoja vietiniame tinkle sujungtų kompiuterių procesoriais, t. y. naudoja juos lygiagretiems skaičiavimams.

Produktas turi turėti vartotojo sąsają, kuri:

- būtų patogi;
- būtų lengvai suprantama;
- nereikalautų papildomų programavimo žinių;
- mandagiai praneštų apie galinčius kilti nesklandumus.

3.1.2. Sistemos kontekstas

Projekte realizuojamas produktas yra integruojamas kartu su Maple ir MPI sistemomis. Produktas naudosis Maple sistema duomenims įvesti ir išvesti (lentelės pavidalu, grafiku) bei kai kurioms funkcijoms (pvz. matematinės išraiškos supaprastinimui) vykdyti. Todėl kuriamas produktas turi atitikti Maple sistemoje taikomus duomenų struktūrų apibrėžimus. Produktas naudos MPI sistemos servisus, lygiagrečiam matematinių išraiškų pertvarkymui valdyti. Todėl kuriamas produktas turi atitikti MPI sistemos teikiamų servisų reikalavimus.

3.1.3. Produkto vartotojas

Potencialūs produkto vartotojai – dabartiniai Maple sistemos vartotojai. Todėl produkto vartotojas gali neblogai žinoti Maple sistemą, bet gali nieko nežinoti apie MPI. Todėl sistema turi būti suprantama ir nežinantiems MPI.

3.1.4. Vartotojo problemos

Dabartinė vartotojo problema: nepatenkinamas didelių matematinių išraiškų diferencijavimo greitis. Sukurtame produkte turi būti pasiektas vartotoją tenkinantis matematinių išraiškų diferencijavimo greitis.

3.1.5. Vartotojo tikslai

Vartojimo požiūriu, kuriamas produktas neturi būti sudėtingesnis nei Maple sistema. Vartotojas neturi papildomai mokytis, kad galėtų inicijuoti matematinių išraiškų diferencijavimą.

Vartotojas naudodamas sukurtą produktą nori greičiau nei su Maple sistema diferencijuoti matematines išraiškas. Vartotojas negali sau leisti panaudoti efektyvias bet brangias daugiaprocesorines sistemas, tačiau gali sau leisti įsigyti kelis kompiuterius, sujungti juos į vietinį tinklą ir diferencijuoti matematines išraiškas naudodamas kelis paskirstytus procesorius.

3.1.6. Bendri apribojimai

Produktas turi tokius nefunkcinius reikalavimus:

- C++ programavimo kalba;
- Linux operacinė sistema;
- atvira funkcionalumų papildymams.

Projektas turi tokius organizacinius apribojimus:

- resursų trūkumas (techniniai resursai testavimui prieinami tik 2 val. per savaitę, kūrėjų grupės vadovas projektui gali skirti tik 2 val. per savaitę, kūrėjų grupės nariai – po 6 val. per savaitę);
- personalo patirties stoka.

3.2. Reikalavimų specifikacija

Reikalavimų specifikacija skirta sistemos reikalavimams tarp sistemos užsakovo ir sistemą realizuojančios komandos suderinti. Dokumentas parengtas pagal programų inžinerijos standartus ir pagal dėstytojų rekomendacijas.

3.2.1. Diegimo aplinka

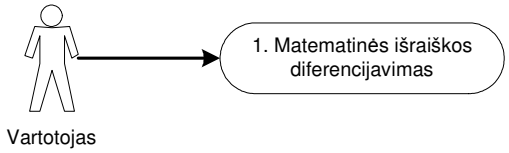
Sistema turi būti sukurta funkcionuoti PC tipo kompiuteriuose Linux operacinėje sistemoje. Sistemos greičiui užtikrinti yra reikalaujamas C++ kompiliatorius. Pilnaverčiam sistemos funkcionavimui reikalingas vietinis kompiuterinis tinklas (iki 10 kompiuterių), su įdiegtomis lygiagretaus valdymo MPI ir matematinio programinio paketo Maple sistemomis.

3.2.2. Bendradarbiaujančios sistemos

Sistema turi bendradarbiauti su matematiniu programiniu paketu Maple. Tiksli sąsajos specifikacija bus rengiama sistemos realizacijos etape. Šis paketas turi būti naudojamas kai kuriems standartiniams matematinių išraiškų pertvarkymams.

Sistema taip pat turi naudotis lygiagretaus funkcionavimo valdymo sistema MPI.

3.2.3. Veiklos kontekstas



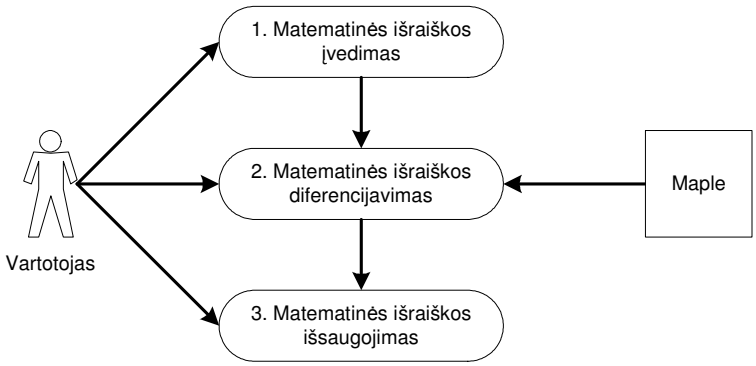
pav. 6. Veiklos konteksto diagrama

3.2.4. Veiklos padalijimas

lentelė 2. Veiklos padalijimas

Įvykio Nr.	1
Įvykis	**Matematinės išraiškos diferencijavimas**
Įeinantys informacijos srautai	Pradinė matematinė išraiška
Išeinantys informacijos srautai	Išdiferencijuota matematinė išraiška Diferencijavimui sugaištas laikas

3.2.5. Sistemos ribos



pav. 7. Panaudojimo atvejų diagrama

3.2.6. Panaudojimo atvejų sąrašas

lentelė 3. Matematinės išraiškos diferencijavimo panaudojimo atvejis

Panaudojimo atvejo Nr.	2
Panaudojimo atvejis	**Matematinės išraiškos diferencijavimas**
Tikslas	Išdiferencijuoti vartotojo įvestą matematinę išraišką
Aktoriai	Vartotojas, Maple
Ryšiai su kitais PA	1, 3
Nefunkciniai reikalavimai	Diferencijavimas turi būti greitesnis nei užsakovo anksčiau naudoti produktai
Pradinės sąlygos	Sistema yra aktyvi Matematinė išraiška yra įvesta Maple sistema yra pasiekiama
Sužadinimo sąlygos	Vartotojas sužadina matematinės išraiškos diferencijavimo funkciją
Pabaigos sąlygos	Ekrane parodoma išdiferencijuota matematinė išraiška arba jos dalis (kai išraiška yra labai ilga)
Pagrindinis scenarijus	Sistema integruota su Maple sistema išdiferencijuoja įvestą matematinę išraišką Sistema ekrane atvaizduoja išdiferencijuotą išraišką arba jos dalį (kai išraiška yra labai ilga) ir diferencijavimui sugaištą laiką
Alternatyvūs scenarijai	Matematinė išraiška neįvesta Nekorektiška matematinė išraiška Maple sistema yra nepasiekiama Kitais atvejais, sistema praneša “Nenumatyta klaida”

3.2.7. Funkciniai reikalavimai

lentelė 4. Funkcinis reikalavimas Nr. 1

Reikalavimo Nr.	1
Reikalavimo tipas	Funkcinis reikalavimas
Panaudojimo atvejis	1. Matematinės išraiškos diferencijavimas
Aprašymas	Sistema turi teisingai išdiferencijuoti vartotojo įvestą matematinę išraišką
Pagrindimas	Vartotojui reikalingas tik teisingas matematinės išraiškos išdiferencijavimas
Šaltinis	Vartotojas
Tikimo kriterijus	Įvesta matematinė išraiška yra teisingai išdiferencijuota
Užsakovo tenkinimas	3

Užsakovo netenkinimas	5
Priklausomybės	2, 3, 4, 5
Konfliktai	–
Papildoma medžiaga	–
Istorija	Reikalavimas užregistruotas 2003 03 14

3.2.8. Reikalavimai duomenims

Matematinė išraiška – tai simbolinė išraiška, galinti susidėti iš lentelėje Nr. nurodytų simbolių.

lentelė 5. Leistini matematinės išraiškos simboliai

Simbolis	Apibūdinimas	Naudojimas	Pavyzdžiai
0-9	Visi skaitmenys	Skaičiai, indeksai	805, t2
,	Kablelis	Trupmenoms	2,05
.	Taškas	Trupmenoms	0,92
A-Z	Didžiosios lotyniškos raidės	Funkcijos, nežinomieji	PI, X, Y
a-z	Mažosios lotyniškos raidės	Funkcijos, nežinomieji	sin, a, h
+	Pliusas	Sudėtis	a + 5, t + t1
–	Minusas	Atimtis, neigimas	h – 1, -3
*	Žvaigždutė	Sandauga	7 * h, 2 * PI
/	Pavirtęs brūkšniukas	Dalyba	1 / 2, PI / 2
( )	Skliausteliai	Operacijų tvarka	2 * (b + 3)
^	Stogelis	Laipsnis	c^2, x^e

Matematinėse išraiškose gali būti naudojamos žemiau pateiktoje lentelėje nurodytos funkcijos.

lentelė 6. Matematinėse išraiškose leistinos funkcijos

Junginys	Apibūdinimas	Pavyzdžiai
sin, SIN	Trigonometrinė funkcija sinusas	sin(x), sin(pi)
cos, COS	Trigonometrinė funkcija kosinusas	cos(a – b), cos(1)
x, X	Nežinomasis x	x ^ 2, x / 5

Matematinės išraiškos ilgis nėra apribotas. Ekrane yra vaizduojama visa matematinė išraiška jei jos ilgis neviršija 1024 simbolių. Priešingu atveju yra vaizduojami pirmi 1024 simboliai.

3.2.9. Reikalavimai vykdymo charakteristikoms

Diferencijavimo greitis – pagrindinis sistemos reikalavimas. Skaitinės šio reikalavimo vertės nėra numatytos, nes sistema kuriama tiriamojo darbo pagrindu. Yra numatyta, kad diferencijavimo laikas turi būti mažesnis nei iki šiol užsakovo naudojamų matematinių programiniu paketų.

3.3. Architektūros specifikacija

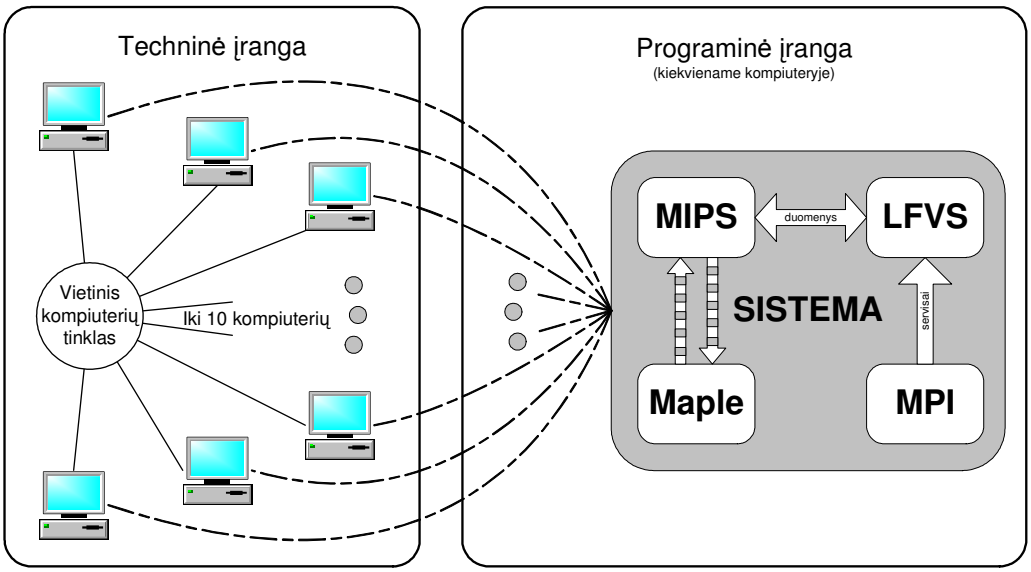
Architektūros specifikacija yra skirta sistemos architektūrai specifikuoti. Dokumentas yra parengtas Rational Unified Process (RUP) pagrindu naudojant Unified Modeling Language (UML).

Dokumente pateikiamas apibendrintas architektūros modelis, sistemos aplinka ir sistemos panaudojimo scenarijai. Vėliau apžvelgiami sistemos projektavimo kriterijai, detalizuojama procesų architektūra, veikimo aplinka, komponentų architektūra, pagrindinių sistemos nefunkcinių reikalavimų įvykdymo priemonės ir sprendimai.

Kuriamos programų sistemos architektūros parinkimas – svarbus projektinis sprendimas – atliekamas vadovaujantis analizės etape pateikta reikalavimų specifikacija.

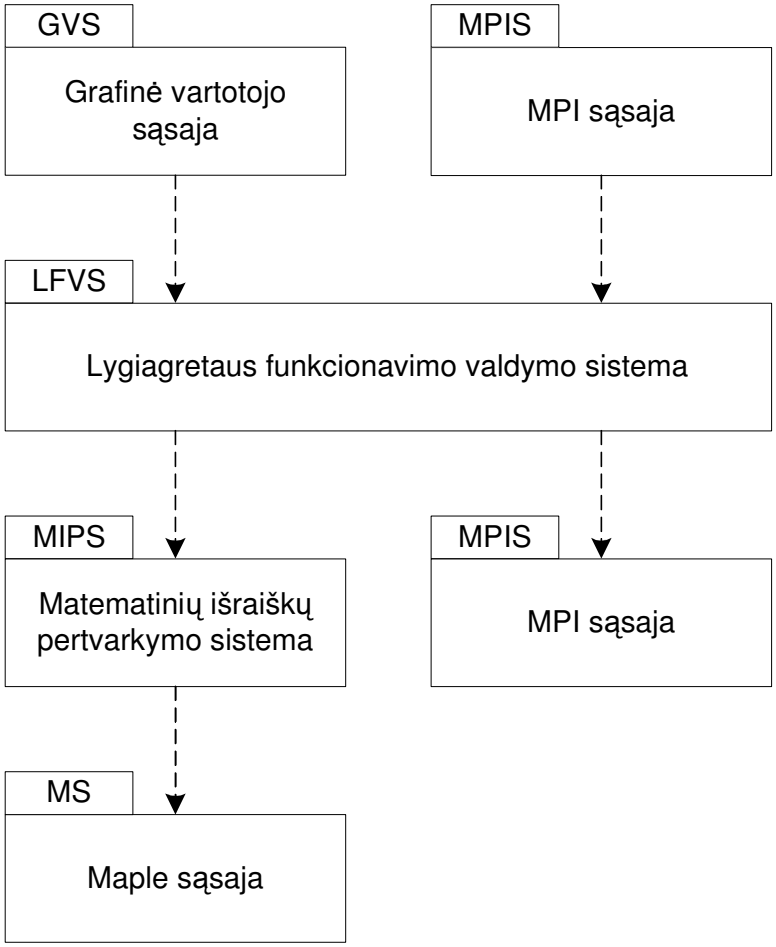
3.3.1. Apibendrintas architektūros modelis

Žemiau pateiktas apibendrintas sistemos veikimo principas.



pav. 8. Sistemos veikimo principas

Toks sistemos veikimo principas buvo numatytas dar projekto paraiškoje. Šis architektūros modelis yra tinkamas ir bus plėtojamas.

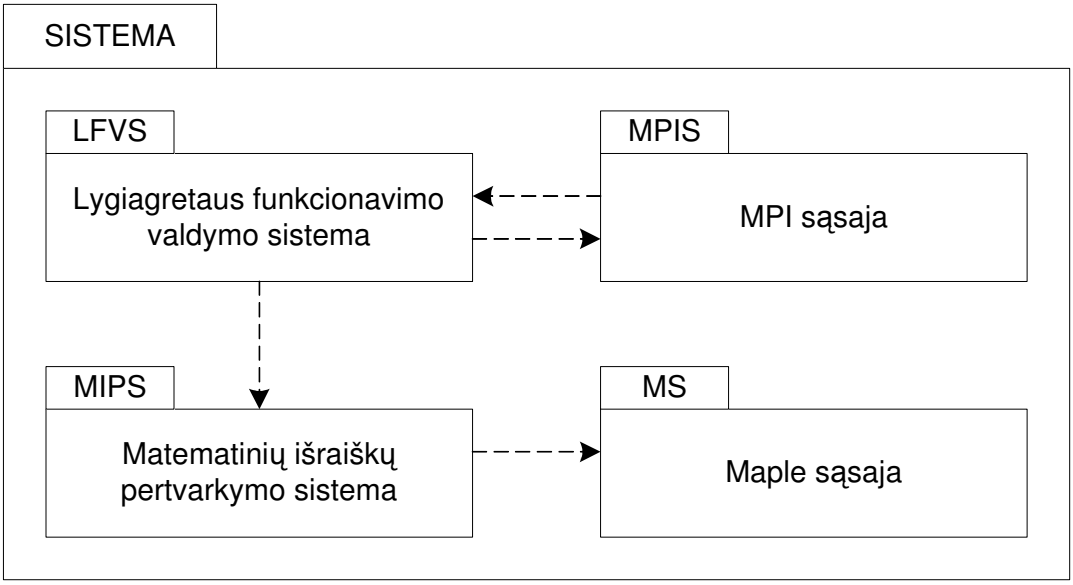


pav. 9. Architektūros modelis

Tokios architektūros sistema turi būti įdiegta kiekviename skaičiavimams naudojamame vietinio tinklo kompiuteryje. Vartotojo sąsaja yra naudojama pagrindiniame kompiuteryje, t. y. tame kompiuteryje, kuriame vartotojas inicijuoja skaičiavimus. Kituose kompiuteriuose skaičiavimus inicijuoja MPI servisas. Lygiagretaus funkcionavimo valdymo sistema yra atsakinga už matematinių išraiškų pertvarkymo veiksmų išlygiagretinimą į kelis procesus (jei tai yra naudinga). MPI sąsaja naudojama išlygiagretintiems procesams inicijuoti naudojant MPI servisą. Matematinių išraiškų pertvarkymo sistema geba pagal reikalavimus pertvarkyti reikalavimus atitinkančią matematinę išraišką. MIPS naudoja Maple sąsają, kad galėtų įvykdyti kai kurias standartines Maple funkcijas (pvz. reiškinio supaprastinimas).

3.3.2. Sistemos struktūra

Kuriama sistema yra paskirstyta ir yra projektuojama veikti vietiniame (lokaliame) kompiuterių tinkle, kai sistema yra atskirai įdiegiama kiekviename kompiuteryje.



pav. 10. Sistemos modulių dekompozicija

Kiekviena modulis aptariamas žemiau.

3.3.3. Matematinių išraiškų pertvarkymo sistema

Matematinių išraiškų pertvarkymo sistema (MIPS) yra atsakinga už galutinį, reikalavimus atitinkantį matematinių išraiškų pertvarkymą.

Modulio įėjimo duomenimis yra reikalavimus atitinkanti ir korektiška matematinė išraiška. Gautą išraišką MIPS dar perduoda Maple sistemai, kad ši atliktų išraiškos supaprastinimą. Duomenims į Maple sistemą perduoti ir rezultatams grąžinti iš jos yra naudojama Maple sąsaja. Supaprastinta ar ne, išraiška toliau analizuojama kaip tekstinė eilutė ir yra atliekami reikalavimuose apibrėžti pertvarkymai. Pertvarkyta išraiška yra laikoma modulio išėjimo duomenimis (rezultatais).

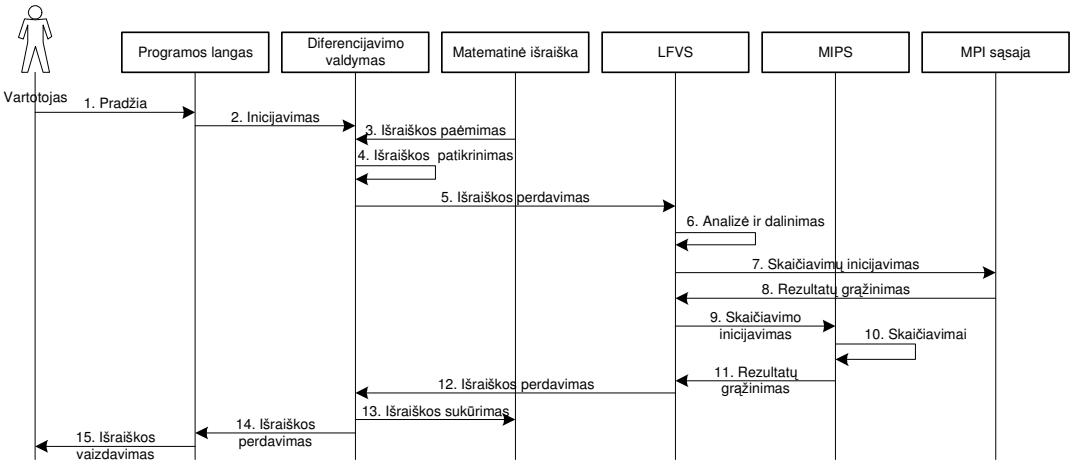
3.3.4. Maple sąsaja

Maple sąsaja (MS) yra atsakinga už matematinių išraiškų perdavimą tarp MIPS ir Maple bei Maple funkcijų inicijavimą.

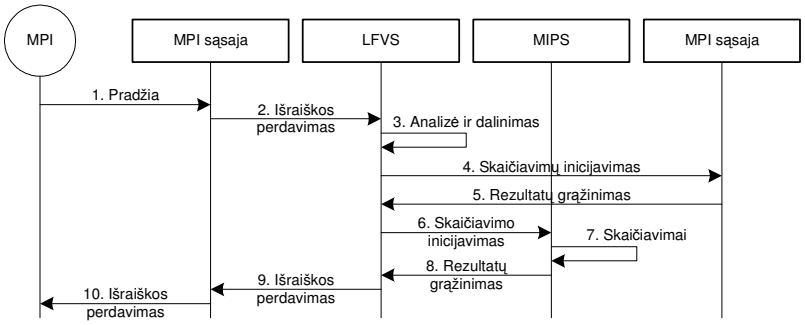
Modulio įėjimo duomenimis yra reikalavimus atitinkanti ir korektiška matematinė išraiška bei Maple funkcijos identifikatorius kartu su papildomais argumentais. Modulis startuoja Maple sistemą, perduoda jai gautą matematinę išraišką, inicijuoja reikalaujamą funkciją ir pasiima iš Maple suskaičiuotą išraišką. Ši išraiška yra laikoma modulio išėjimo duomenimis (rezultatais).

3.3.5. Panaudojimo atvejų sekų diagramos

Sekų diagramos naudojamos atvaizduoti sistemos objektų sąveikai, ryšiams tarp objektų ir pranešimams, kuriais keičiasi objektai.



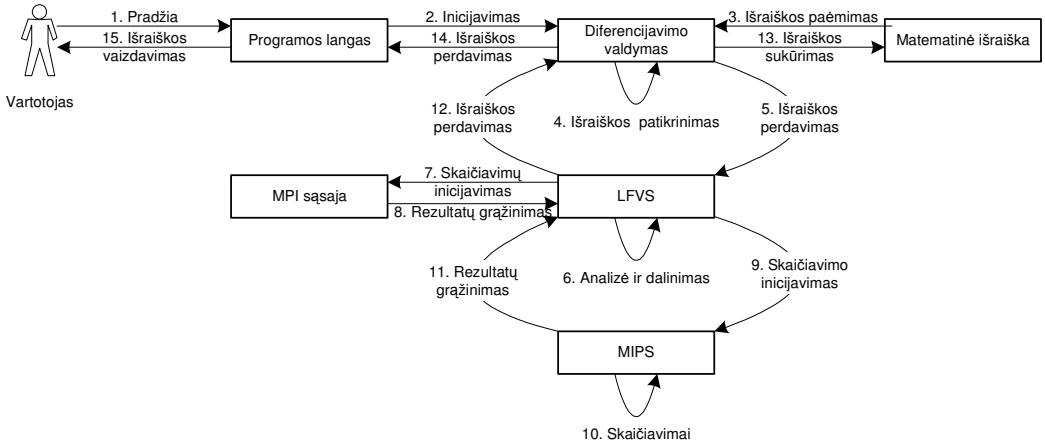
pav. 11. Matematinės išraiškos diferencijavimo (kai inicijuoja vartotojas) sekų diagrama



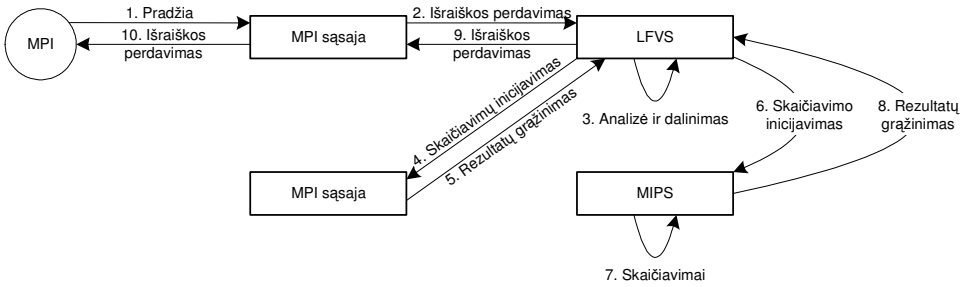
pav. 12. Matematinės išraiškos diferencijavimo (kai inicijuoja sistema) sekų diagrama

3.3.6. Objektų bendradarbiavimo diagramos

Objektų bendradarbiavimo diagramos vaizduoja didelį, bet detalų objektų sąveikos paveikslą, nes jos pateikia tiek objektų ryšius, tiek pranešimus.



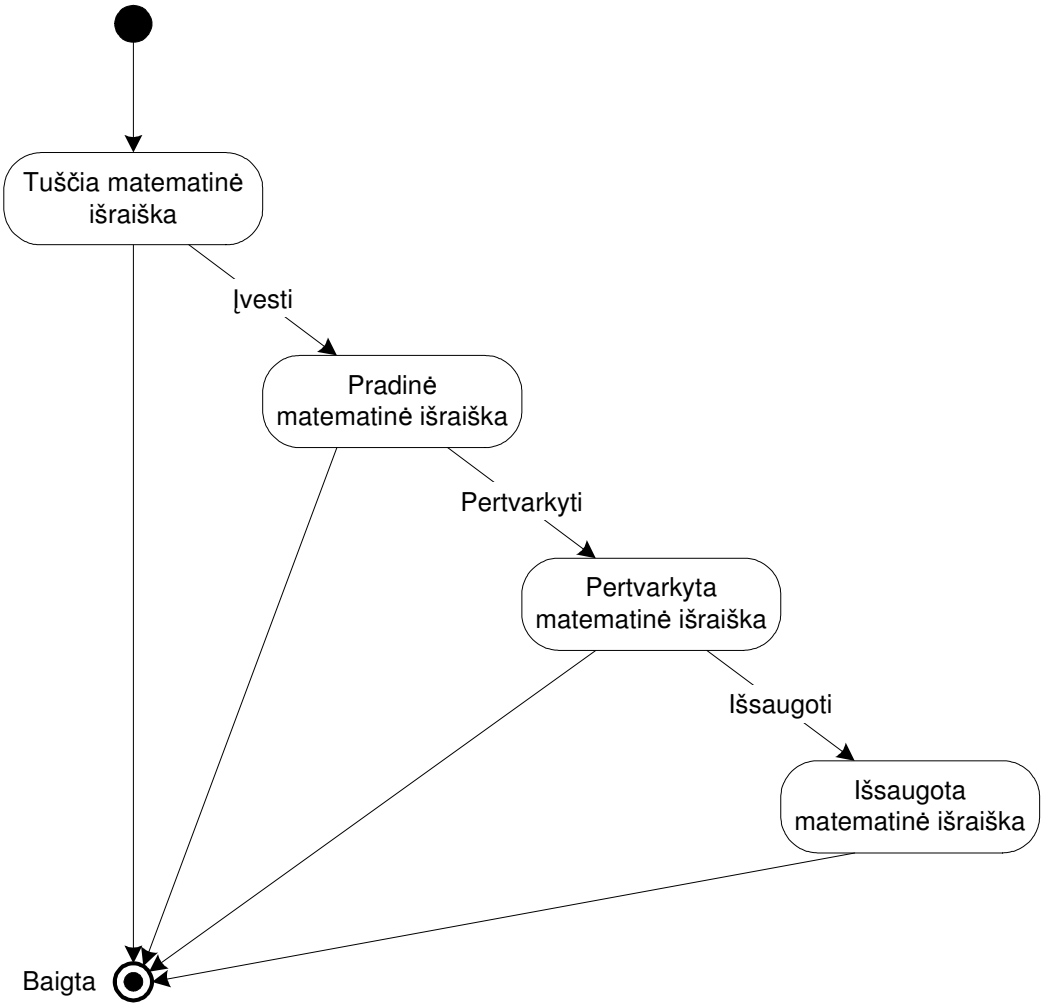
pav. 13. Matematinės išraiškos diferencijavimo (kai inicijuoja vartotojas) bendradarbiavimo diagrama



pav. 14. Matematinės išraiškos diferencijavimo (kai inicijuoja sistema) bendradarbiavimo diagrama

3.3.7. Objektų būsenų kaitos diagramos

Būsenų diagramos leidžia aprašyti modeliuojamų objektų elgesį. Paprastai jos naudojamos klasių elgesiui aprašyti.



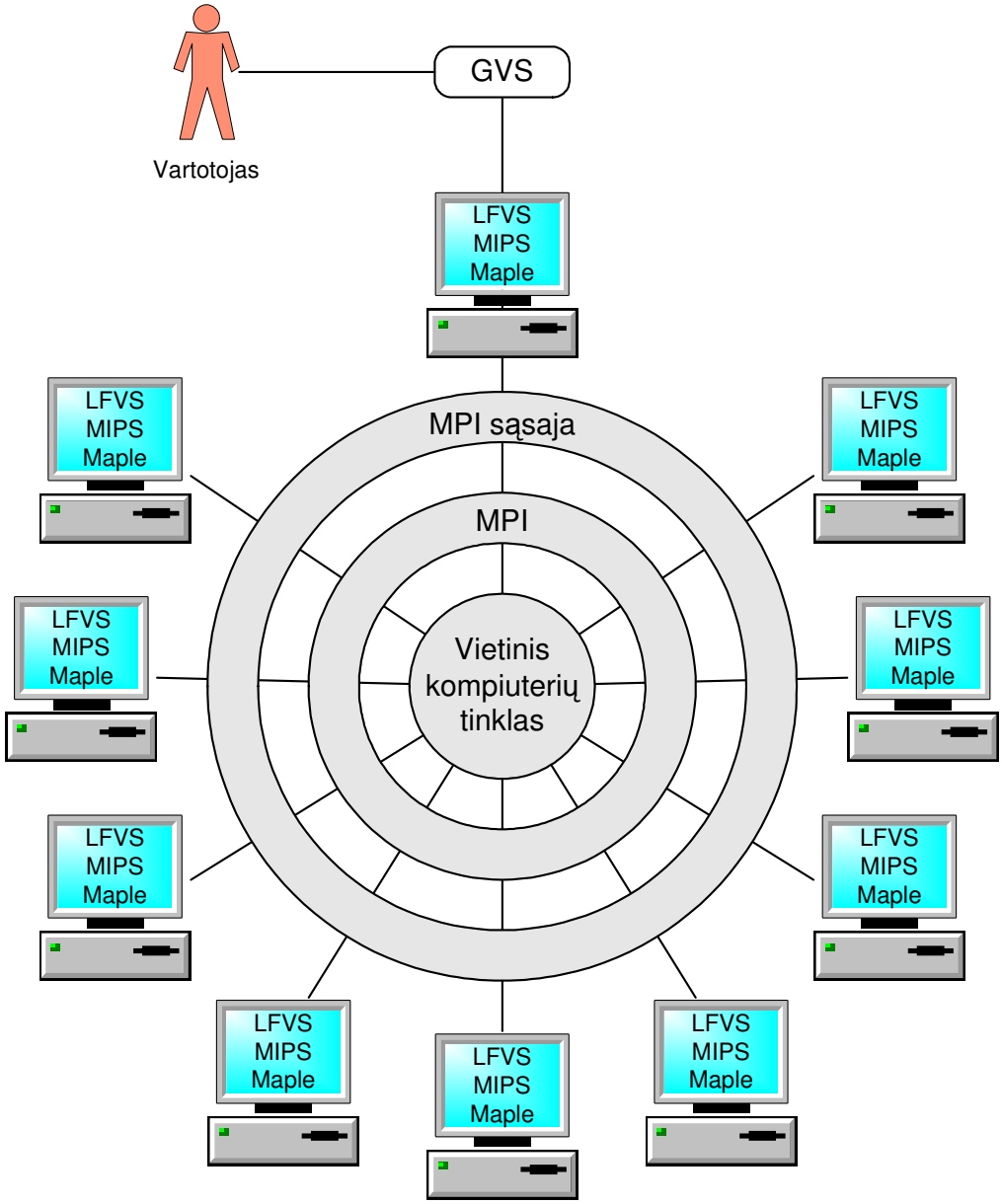
pav. 15. Matematinės išraiškos būsenų kaitos diagrama

3.3.8. Veikimo aplinka

Sistema gali funkcionuoti ir atskiroje darbo vietoje. Pilnam sistemos išnaudojimui reikalingi iki 10-ies kompiuterių, sujungtų į vietinį kompiuterių tinklą. Kiekviename kompiuteryje turi būti

įdiegta Linux operacinė sistema, MPI servisas, Maple programinis paketas bei kuriama sistema. Tokios sistemos schema buvo pateikta produkto apibūdinimo skyrelyje.

Žemiau esančioje schemoje pavaizduotas architektūrinių komponentų išsidėstymas minėtoje veikimo aplinkoje.

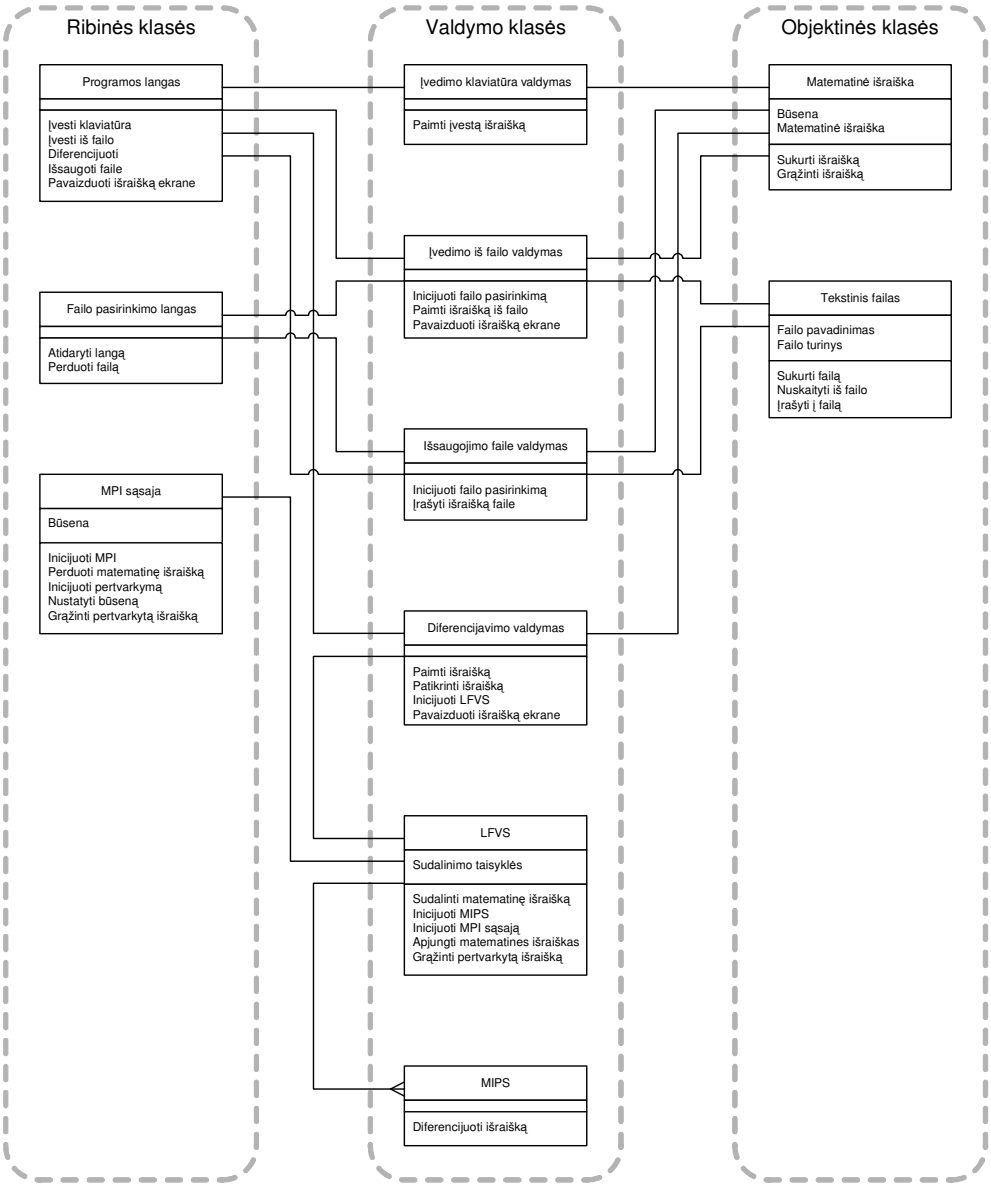


pav. 16. Architektūrinių komponentų išsidėstymas

3.3.9. Klasių diagrama

Komponentų architektūra – tai sistemos struktūra sudaryta iš susietų komponentų. Sistemos komponentų projektavimas apima komponentų identifikavimą ir jų sąryšių pavaizdavimą. Čia atspindimi statiniai sistemos aspektai.

Kaip pavaizduota pav. 10., sistema sudaryta iš keturių modulių: LFVS, MIPS, MPI sąsaja ir Maple sąsaja. Žemiau pateikiama klasių diagrama, kurioje matyti kiekvienos klasės pavadinimas, atributai ir operacijos bei ryšiai tarp klasių.



pav. 17. Klasių diagrama

3.3.10. Vykdymo charakteristikos

Pagrindinis sistemos reikalavimas – simbolinio diferencijavimo greitis. Dėl to turi būti priimti ir atitinkami architektūriniai sprendimai, kuriems esant būtų užtikrintas minimalus sprendimų laikas.

Jau analizės stadijoje buvo parinkta tinkamiausia veikimo aplinka (Linux operacinė sistema) ir tinkamiausias kompiliatorius (funkcionali bei greita C++ programavimo kalba). Greitis taip pat

turi būti pagrindinis rodiklis modeliuojant ir realizuojant sąsajos, valdymo ir, svarbiausia, skaičiavimų algoritmus. Algoritmų modeliavimas atliekamas vėlyvosiose projektavimo ir ankstyvosiose realizacijos stadijose, dėl to šiame dokumente greičio uždavinio sprendimai dar neapibrėžti.

3.3.11. Kokybė

Sistemos architektūrai specifikuoti ir tolimesniems projektavimo darbams buvo nustatyti sistemos kokybės kriterijai ir jų prioritetai. Projektuojant ir parenkant sprendimus, bus pirmiausia atsižvelgiama į aukščiausio prioriteto kriterijų patenkinimą. Kriterijų ir prioritetų lentelė yra nustatyta pagal reikalavimų specifikacijoje nurodytus nefunkcinius reikalavimus.

lentelė 7. Sistemos kokybės kriterijai

Prioritetas / Kriterijus	l. aukštas	aukštas	vidutinis	žemas	l. žemas
Panaudojamumas				●	
Saugumas					●
Efektyvumas	●				
Teisingumas	●				
Patikimumas	●				
Palaikomumas		●			
Testuojamumas		●			
Lankstumas				●	
Suprantamumas			●		
Pakartotinis panaudojimas		●			
Pernešamumas				●	
Įsiliejimas			●		

Sistema skirta greitam konkrečių uždavinių sprendimui. Dėl to yra reikalaujamas didelis greitis, teisingumas, patikimumas, tačiau nereikalaujama aukšto panaudojamumo, lankstumo, suprantamumo. Dėl labai aukštų efektyvumo, teisingumo ir patikimumo kriterijų prioritetų, sistemai papildomai reikalingas aukšto prioriteto testuojamumo kriterijus, kad sistema būtų sėkmingai ištestuota. Reikalavimų specifikacijoje apibrėžta nedidelė sistemos pernešimo tikimybė, tačiau aukšta sistemos plėtojimo tikimybė, kas lemia aukštus palaikomumo ir pakartotinio panaudojimo kriterijų prioritetus.

3.4. Detalios architektūros specifikacija

Detalios architektūros specifikacija yra skirta sistemos komponentų detaliai architektūrai specifikuoti. Dokumentas parengtas pagal programų inžinerijos standartus ir pagal dėstytojų rekomendacijas. Komponentų projektavimas – vėlyva sistemos projektavimo proceso stadija. Dokumentas skirtas sistemos kūrėjams, kurie naudos dokumentą visose sistemos realizacijos proceso stadijose.

3.4.1. Diferencijavimo valdymas

- Klasifikacija

Pavadinimas	Diferencijavimo valdymas
Tipas	Valdymo klasė
Posistemė	Grafinė vartotojo sąsaja
Versija	1.0, 2003.05.17

- Apibrėžimas

  Komponentas skirtas matematinės išraiškos simboliniam diferencijavimui valdyti.

- Atsakomybės

  Komponentas atsakingas už matematinės išraiškos simbolinio diferencijavimo valdymą, kas apima matematinės išraiškos korektiškumo patikrinimo inicijavimą, lygiagretaus funkcionavimo valdymo sistemos sužadinimą bei rezultatinės matematinės išraiškos pavaizdavimą pagrindiniame programos lange.

- Apribojimai

  Apribojimų nėra.

- Struktūra

  Komponentas neturi viešų subkomponentų.

- Sąveikavimas

  Komponentas yra naudojamas programos lango komponento egzemplioriaus, kuris atitinkamai sužadina šią valdymo klasę.

  Komponentas savo ruožtu naudoja du matematinės išraiškos egzempliorius bei lygiagretaus funkcionavimo valdymo sistemos egzempliorių. Vienas matematinės išraiškos egzempliorius saugo pradinę matematinę išraišką, o kitas – rezultatinę matematinę išraišką. Lygiagretaus funkcionavimo valdymo sistemos komponento egzempliorius naudojamas lygiagrečiam simboliniam diferencijavimui.

- Resursai

  Komponentas naudojasi operacinės sistemos funkcijomis grafinei vartotojo sąsajai organizuoti.

- Skaičiavimai

Komponentas turi vienintelį valdantį metodą, kuris inicijuoja matematinės išraiškos korektiškumo patikrinimą, paleidžia lygiagretaus funkcionavimo valdymo sistemą bei pavaizduoja rezultatinę matematinę išraišką pagrindiniame programos lange.

- Sąsaja/eksportas

  Valdantis metodas parametrų neturi.

3.4.2. Matematinių išraiškų pertvarkymo sistema

- Klasifikacija

Pavadinimas	Matematinių išraiškų pertvarkymo sistema
Tipas	Valdymo klasė
Posistemė	Matematinių išraiškų pertvarkymo sistema
Versija	1.0, 2003.05.16

- Apibrėžimas

  Komponentas skirtas matematinei išraiškai simboliškai diferencijuoti.

- Atsakomybės

  Komponentas atsakingas už simbolinį matematinės išraiškos diferencijavimą.

- Apribojimai

  Simbolinis diferencijavimas – tai diferencijavimas tam tikro kintamojo atžvilgiu, rezultate neįstatinėjant reikšmių, t. y. rezultate paliekant gautą naują matematinę išraišką. Pati diferencijavimo funkcija taip pat yra apribota funkcionalumu. Tai buvo nurodyta dar reikalavimų specifikacijoje.

- Struktūra

  Komponentas neturi viešų subkomponentų.

- Sąveikavimas

  Diferencijuojant ypač aktyviai naudojami du matematinės išraiškos egzemplioriai: vienas su pradine matematine išraiška, kitas su rezultatine matematine išraiška.

  Pats komponentas yra naudojamas lygiagretaus funkcionavimo valdymo sistemoje.

- Resursai

  Komponentui svarbiausias resursas yra procesorius.

- Skaičiavimai

  Komponento konstruktorius skirtas komponentui inicijuoti. Valdantis metodas skirtas matematinei išraiškai simboliškai diferencijuoti.

  Matematinės išraiškos simbolinio diferencijavimo algoritmas dar nėra numatytas.

- Sąsaja/eksportas

  Komponentui inicijuoti reikalinga iškviesti konstruktorių, kurio parametras yra matematinė išraiška. Valdantis metodas parametrų neturi, tačiau grąžina rezultatinę matematinę išraišką.

3.4.3. Matematinė išraiška

- Klasifikacija

Pavadinimas	Matematinė išraiška
Tipas	Objektinė klasė
Posistemė	-
Versija	1.0, 2003.05.15

- Apibrėžimas

  Komponentas skirtas matematinei išraiškai saugoti, korektiškumui tikrinti. Realizacijos procese klasė gali būti papildoma kitais funkcionalumais, pvz. nedideliems pertvarkymams vykdyti, supaprastinti ir pan.

- Atsakomybės

  Komponentas atsakingas už vienos matematinės išraiškos saugojimą. Taip pat komponentas geba pasitvarkyti išraišką, t. y. pašalinti ir taip ignoruojamus simbolius, bei patikrinti išraiškos korektiškumą.

- Apribojimai

  Veiksmai su komponentu yra neleidžiami, kol matematinei išraiškai nėra patikrintas korektiškumas. Jei matematinė išraiška nekorektiška, veiksmai ir toliau draudžiami.

  Matematinės išraiškos formatas buvo numatytas dar reikalavimų specifikacijoje. Išraiškoje gali būti sakančioje lentelėje nurodyti simboliai.

lentelė 8. Leistini matematinės išraiškos simboliai

Simbolis	Apibūdinimas	Naudojimas	Pavyzdžiai
0-9	Visi skaitmenys	Skaičiai, indeksai	805, t2
,	Kablelis	Trupmenoms	2,05
.	Taškas	Trupmenoms	0,92
A-Z	Didžiosios lotyniškos raidės	Funkcijos, nežinomieji	PI, X, Y
a-z	Mažosios lotyniškos raidės	Funkcijos, nežinomieji	sin, a, h
+	Pliusas	Sudėtis	a + 5, t + t1
–	Minusas	Atimtis, neigimas	h – 1, -3
*	Žvaigždutė	Sandauga	7 * h, 2 * PI
/	Pavirtęs brūkšniukas	Dalyba	1 / 2, PI / 2
( )	Skliausteliai	Operacijų tvarka	2 * (b + 3)
^	Stogelis	Laipsnis	c^2, x^e

Matematinėse išraiškose gali būti naudojamos žemiau esančioje lentelėje nurodytos funkcijos.

lentelė 9. Matematinėse išraiškose leistinos funkcijos

Junginys	Apibūdinimas	Pavyzdžiai
sin, SIN	Trigonometrinė funkcija sinusas	sin(x), sin(pi)
cos, COS	Trigonometrinė funkcija kosinusas	cos(a – b), cos(1)
x, X	Nežinomasis x	x^2, x / 5

Matematinės išraiškos ilgis nėra apribotas.

- Struktūra

  Pagrindinė komponento dalis – matematinė išraiška. Kad išraiškos dydis nebūtų labai apribotas, jai saugoti išskiriama vieta yra dinaminėje atminties dalyje. Taip pat komponentas turi išraiškos korektiškumo požymį.

- Sąveikavimas

  Komponento egzempliorius yra naudojamas visoje sistemoje. Komponento egzemplioriaus naudojimas yra prasmingas tik esant korektiškai matematinei išraiškai.

- Resursai

  Matematinė išraiška yra saugoma dinaminėje operatyvinės atminties dalyje. Ši atmintis, priklausomai nuo naudojamo kompiuterio, yra ribota. Tačiau, pagal reikalavimų specifikaciją, matematinės išraiškos ilgis nėra apribotas, todėl naudojant komponentą, būtina, kad operatyvinėje atmintyje būtų pakankamai laisvos vietos.

- Skaičiavimai

  Komponento konstruktorius skirtas komponentui inicijuoti. Papildymo metodas geba papildyti matematinę išraiška atitinkamai išskirdamas papildomos atminties. Korektiškumo patikrinimo metodas geba patikrinti matematinės išraiškos korektiškumą. Komponento destruktorius skirtas išskirtai atminčiai atlaisvinti.

  Matematinės išraiškos korektiškumo tikrinimas vykdomas rekurentiškai tikrinant tarp skliaustų esančią matematinę išraišką. Matematinė išraiška, kurioje nėra skliaustų, tikrinama kaip išraiškos elementų (kintamųjų, konstantų, operacijų) seka, žinant kokios elementų poros gali būti viena paskui kitą ir kokios negali.

  Komponentas turi savo išimčių gaudyklę, t. y. įvykus klaidai, geba tvarkingai nutraukti tolimesnius veiksmus ir apie tai informuoti vartotoją. Specialiai yra apdorojamos darbo su operatyvine atmintimi klaidos.

- Sąsaja/eksportas

  Komponentui inicijuoti reikalinga iškviesti konstruktorių, kuris neturi parametrų. Papildymo metodas kaip parametro reikalauja simbolių eilutės, kuria yra papildoma matematinė išraiška. Korektiškumo patikrinimo metodas bei destruktorius parametrų neturi.

3.5. Programinė realizacija

Analizės dalyje aprašytiems sprendimams įgyvendinti reikėjo detaliai suprojektuoti ir realizuoti atitinkamus algoritmus bei tinkamas duomenų struktūras. Projektavimo etape nebuvo atliekami algoritmų projektavimai, nes dar projekto plane buvo numatyta, kad algoritmai bus vystomi evoliuciniu būdu programavimo stadijoje. Šiame skyriuje išdėstomi ir pagrindžiami suprojektuoti algoritmai, detalizuojamos naudojamos duomenų struktūros.

3.5.1. Matematinė išraiška

Pagrindinis sistemos objektas – matematinė išraiška – tai simbolinė eilutė, matematikoje reiškianti reiškinį, kuris gali būti sudarytas iš sveikųjų ir realiųjų, teigiamų ir neigiamų skaičių, konstantų, kintamųjų, aritmetinių operandų (neigimas, suma, atimtis, daugyba, dalyba, kėlimas laipsniu, šaknies traukimas) bei įvairių funkcijų (trigonometrinių, logaritminių ir pan.). Matematinės išraiškos ilgis nėra apribotas. Matematinė išraiška gali susidėti iš žemiau esančioje lentelėje nurodytų simbolių.

lentelė 10. Leistini matematinės išraiškos simboliai

Simbolis	Apibūdinimas	Naudojimas	Pavyzdžiai
0-9	Visi skaitmenys	Skaičiai, indeksai	805, t2
,	Kablelis	Trupmenoms	2,05
.	Taškas	Trupmenoms	0,92
A-Z	Didžiosios lotyniškos raidės	Funkcijos, nežinomieji	PI, X, Y
a-z	Mažosios lotyniškos raidės	Funkcijos, nežinomieji	sin, a, h
+	Pliusas	Sudėtis	a + 5, t + t1
–	Minusas	Atimtis, neigimas	h – 1, -3
*	Žvaigždutė	Sandauga	7 * h, 2 * PI
/	Pavirtęs brūkšniukas	Dalyba	1 / 2, PI / 2
( )	Skliausteliai	Operacijų tvarka	2 * (b + 3)
^	Stogelis	Laipsnis	c^2, x^e

Matematinėse išraiškose gali būti naudojamos žemiau pateiktoje lentelėje nurodytos funkcijos.

lentelė 11. Matematinėse išraiškose leistinos funkcijos

Junginys	Apibūdinimas	Pavyzdžiai
sin, SIN	Trigonometrinė funkcija sinusas	sin(x), sin(pi)
cos, COS	Trigonometrinė funkcija kosinusas	cos(a – b), cos(1)
x, X	Nežinomasis x	x ^ 2, x / 5

Žemiau pateikti keli matematinių išraiškų pavyzdžiai:

- `3 * x ^ 4 – 5 * x ^ 3 + 7 * x ^ 2 – 9 * x + 11`
- `(5 * x – 9,2) ^ 3 * (7 * x + 4) ^ 2`
- `2 * sin(x) – cos(2 * x)`

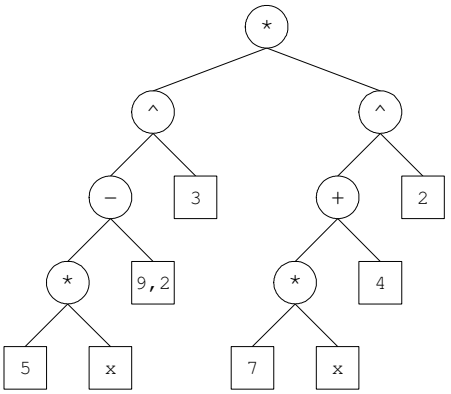
- `(6 * x * sin(5 * x ^ 2 - x)) ^ 3,5`
- `(sin(3 * x)) ^ 2 * (-cos(x ^ 1/2)) * (4 * x ^ 1/3) ^ 4`

Produktas pritaikytas labai didelių (ilgų) matematinių išraiškų pertvarkymams. Labai didelės matematinės išraiškos gali būti sudarytos pavyzdžiui iš 5 milijonų simbolių.

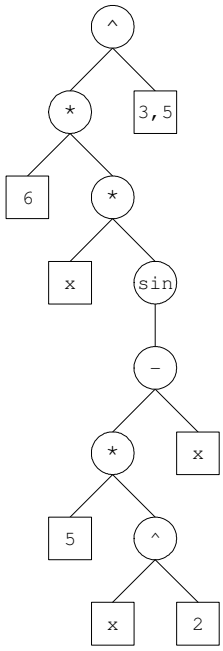
Matematinėms išraiškoms saugoti išorinėse laikmenose yra naudojama simbolinės eilutės duomenų struktūra, atitinkanti matematinį reiškinį, kurių pavyzdžiai pateikti aukščiau.

Matematinėms išraiškoms saugoti programos vykdymo metu yra naudojama operacijų medžio struktūra, kur medžio viršūnes atitinka reiškinio dėmenys (skaičiai, konstantos, kintamieji), o medžio mazgus atitinka matematiniai operandai (aritmetiniai, trigonometriniai, logaritminiai ir pan.). Žemiau pateikti keli matematinių išraiškų pavyzdžiai, transformuoti į operacijų medžio struktūrą:

`(5 * x - 9,2) ^ 3 * (7 * x + 4) ^ 2`



`(6 * x * sin(5 * x ^ 2 - x)) ^ 3,5`



pav. 18. Matematinių išraiškų operacijų medžių pavyzdžiai

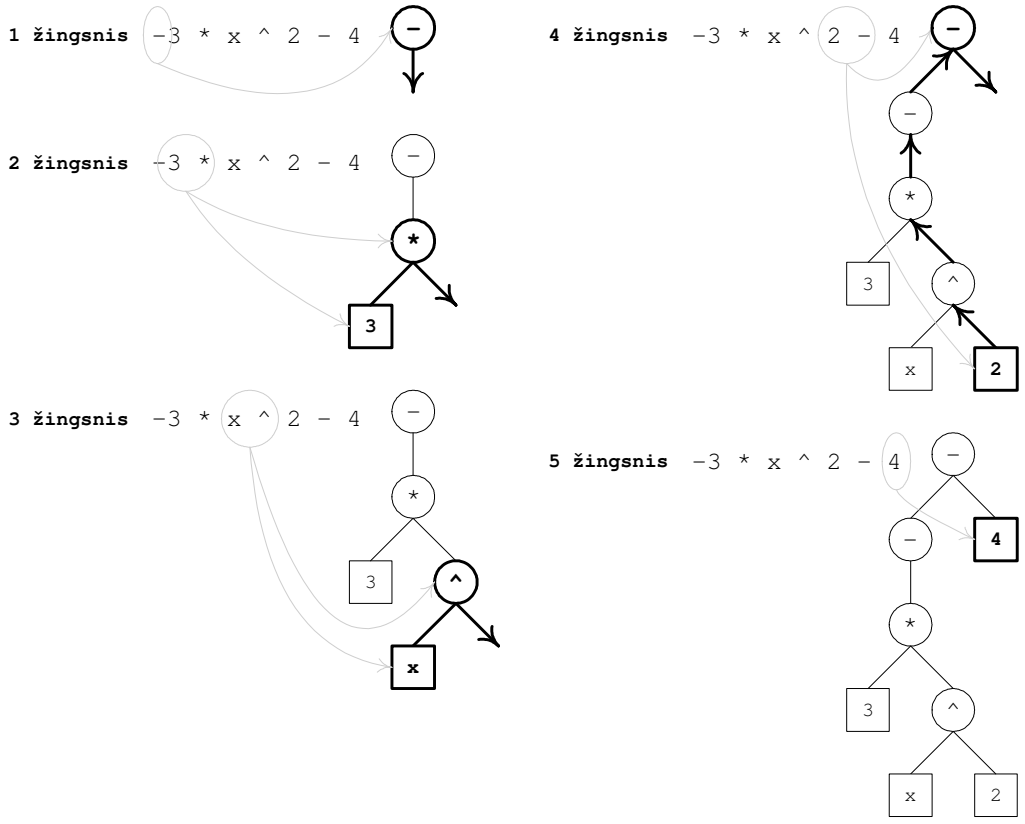
3.5.2. Matematinės išraiškos transformavimas į operacijų medį

Matematinėms išraiškoms transformuoti iš simbolinės eilutės duomenų struktūros į operacijų medžio duomenų struktūrą yra naudojamas išraiškos skaidymo pagal operandus algoritmas. Visiems operandams (neigimas, suma, dalyba, sin, log ir pan.) yra priskirti skaidymo prioritetai, kurių didėjimo tvarka matematinė išraiška yra transformuojama į operacijų medžio struktūrą. Matematinėms išraiškoms su skliausteliais transformavimas rekursiškai vykdomas kiekvienai atskiruose skliauteliuose esančiai subišraiškai. Žemiau esančioje lentelėje didėjimo tvarka nurodyti operandų prioritetai.

lentelė 12. Operandų prioritetai

Operandas	Ženklas	Prioritetas	Pavyzdžiai
Funkcijos		0	sin(x), arctg(2 / y), ln(z)
Teigimas	+	5	+5, +x, +(7,1 * y ^ 3)
Neigimas	–	5	–1, –y, –sin(g / 2)
Suma	+	5	y + 5, sin(x) + cos(x), x ^ 2 + 2 * x
Atimtis	–	5	4 – z, ln(y) – e, (b – a) – (d – c)
Daugyba	*	6	4 * x, (3 + h) * arccos(w), 2 * x * y * z
Dalyba	/	6	r / 2, tg(x) / ctg(x), (–5 * x ^ 2) / (6 * y ^ 3)
Kėlimas laipsniu	^	7	y ^ 3, cos(x) ^ 2, (sin(x) – 1) ^ (n – 1)

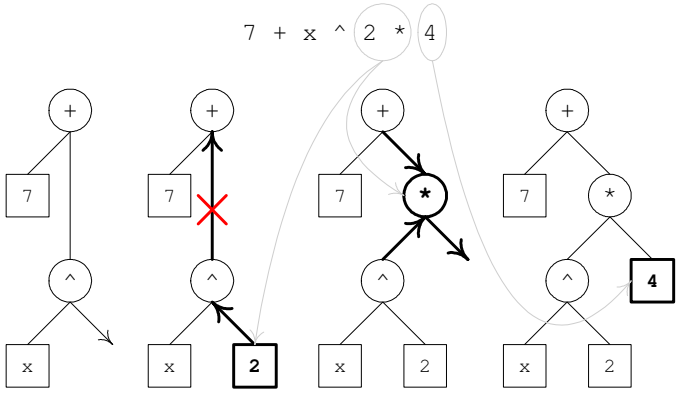
Žemiau esančiame pavyzdyje gerai matyti, kuria kryptimi priklausomai nuo operando prioriteto formuojasi operacijų medis.



pav. 19. Matematinės išraiškos transformavimo į operacijų medį pavyzdys

Jeigu operandas yra aukštesniu arba tokiu pačiu prioritetu nei ankstesnysis operandas, tada operacijų medis iš karto piešiamas platyn į šakas (aukščiau esančiame pavyzdyje 2 ir 3 žingsniai). Jeigu operandas yra žemesniu prioritetu nei ankstesnysis operandas, tada operacijų medžio šaknies link kiekviename mazge yra ieškomas žemesnio arba tokio pačio prioriteto operandas. Pagal operando prioritetą pasirenkama grandis: jeigu operandų prioritetai sutampa, pasirenkama žemesnio lygio grandis (esanti arčiau operacijų medžio šaknies), jeigu nesutampa – aukštesnio lygio grandis (esanti toliau nuo operacijų medžio šaknies). Pasirinkta grandis yra pertraukiama ir

jos viduryje patalpinamas operandas. Operando įterpimas tarp dviejų mazgų yra pavaizduotas žemiau esančiame pavyzdyje.

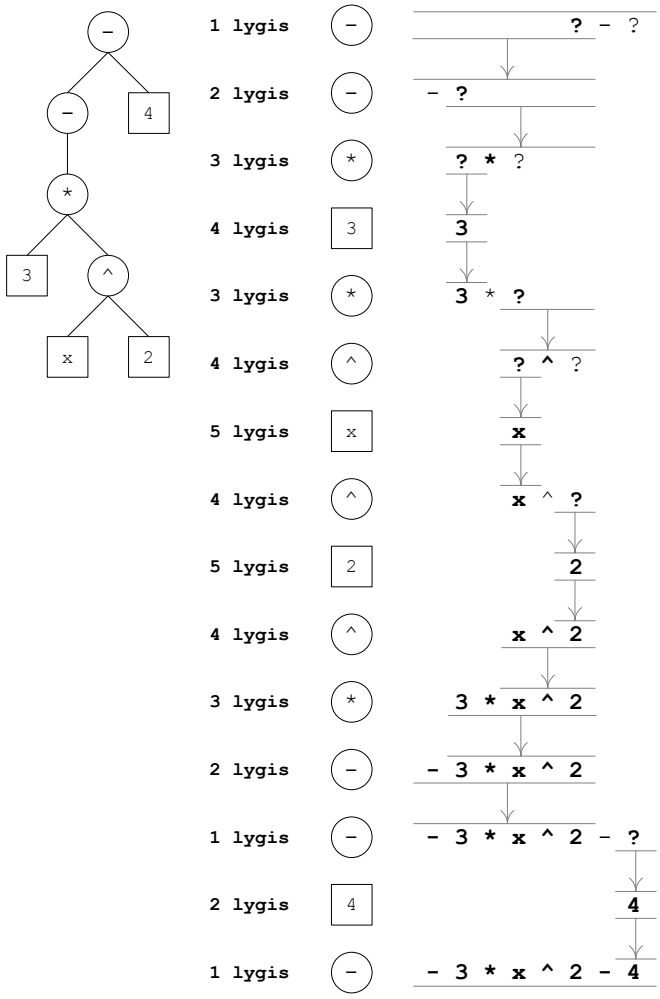


pav. 20. Operando įterpimas tarp dviejų mazgų

Kai iki pat operacijų medžio šaknies nėra randamas žemesnio arba tokio paties prioriteto operandas arba pačioje šaknyje randamas tokio paties prioriteto operandas, tada naujasis operandas tampa operacijų medžio šaknimi (pirmame pavyzdyje 4 žingsnis).

3.5.3. Operacijų medžio transformavimas į matematinę išraišką

Matematinėms išraiškoms transformuoti iš operacijų medžio duomenų struktūros į simbolinės eilutės duomenų struktūrą naudojamas adaptuotas medžio perėjimo nuo šaknies iki viršūnių (arba lapų, nesišakojančių šakų) algoritmas, kiekvieno žingsnio pabaigoje sudedant visus operando dėmenis į simbolinę matematinę išraišką. Žemiau pateiktas transformavimo pavyzdys.

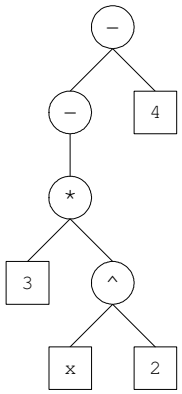


pav. 21. Operacijų medžio transformavimas į matematinę išraišką

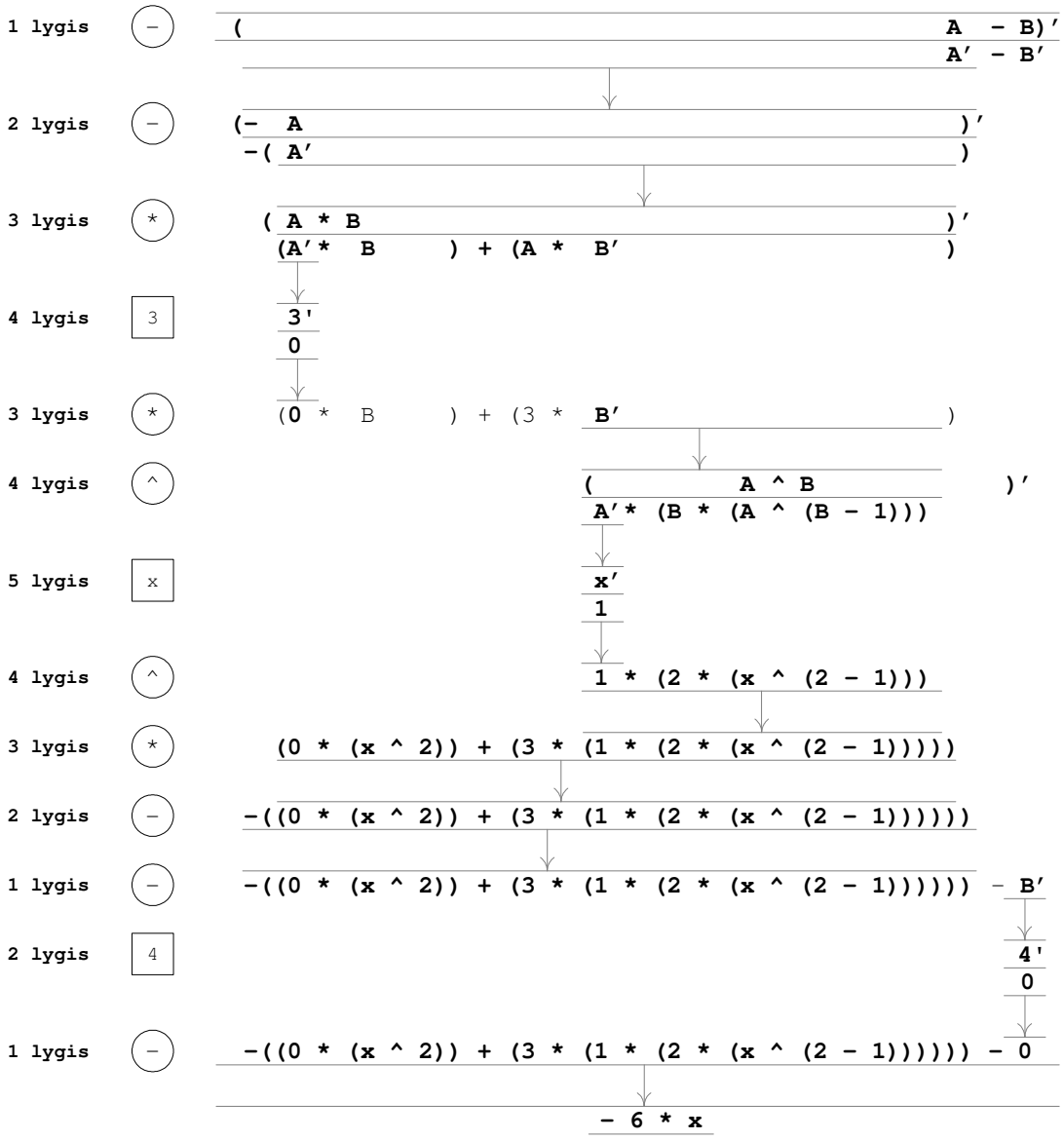
3.5.4. Matematinių išraiškų diferencijavimas

Matematinių išraiškų pertvarkymo sistemai (MIPS) įėjimo ir išėjimo duomenys yra matematinė išraiška. Algoritmo pradžioje matematinė išraiška yra transformuojama į operacijų medžio duomenų struktūrą, tada operacijų medis pertvarkomas ir transformuojamas atgal į matematinę išraišką.

Matematinei išraiškai pertvarkyti naudojamas adaptuotas medžio perėjimo nuo šaknies iki viršūnių (arba lapų, nesišakojančių šakų) algoritmas, kiekvieno žingsnio pabaigoje atliekant operando ir jo dėmenų pakeitimą sprendiniu. Žemiau pateiktas matematinės išraiškos pavyzdys, jo pagrindu detalizuotas pertvarkymo algoritmas ir rezultato matematinė išraiška.

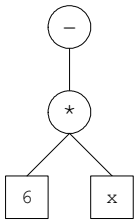


pav. 22. Matematinės išraiškos pavyzdys



pav. 23. Matematinės išraiškos diferencijavimas

Pakomentuosime diferencijavimo veikimą. Šakninis operandas yra atimtis. Atimties išvestinė yra lygi dėmenų išvestinių skirtumui. Leidžiamės į pirmojo dėmens išvestinės skaičiavimą, t. y. į 2-ą lygį. Čia operandas yra neigimas. Neigimo išvestinė lygi neigiamai dėmens išvestinei. Leidžiamės į to dėmens išvestinės skaičiavimą, t. y. į 3-ią lygį. Čia operandas yra daugyba. Daugybos išvestinė yra lygi pirmo dėmens išvestinės ir antro dėmens sandaugos bei antro dėmens išvestinės ir pirmo dėmens sandaugos sumai. Leidžiamės į pirmojo dėmens išvestinės skaičiavimą, t. y. į 4-ą lygį. Čia yra operacijų medžio viršūnė (arba lapas, nesišakojanti šaka), kuriame įrašyta konstanta 3. Konstantos išvestinė lygi 0 (nuliui). Grįžtame į 3-ią lygį. Leidžiamės į antrojo daugybos dėmens išvestinės skaičiavimą, t.y. į 4-ą lygį. Čia operandas yra kėlimas laipsniu. Kėlimo laipsniu, kai laipsnyje yra konstanta, išvestinė yra lygi pagrindo išvestinei padaugintai iš laipsnio ir dar padaugintai pagrindo pakelto vienetu sumažintu laipsniu. Reikės skaičiuoti tik pagrindo išvestinę, todėl ir leidžiamės į 5-ą lygį. Čia yra operacijų medžio viršūnė, kuriame įrašytas kintamasis x. Kintamojo, kurio pagrindu vyksta diferencijavimas, išvestinė yra lygi 1. Grįžtame į 4-ą lygį ir įstatome visas reikšmes į kėlimo laipsniu išvestinės formulę. Įstatymas šiuo atveju reiškia operacijų medžio šakos pakeitimą kita, dažniausiai šakotesne, šaka. Turime ir šį rezultatą, tad grįžtame į 3-ią lygį, kur taip pat įstatome visas reikšmes į daugybos išvestinės formulę. Lygiai taip grįžtame į 2-ą lygį ir vienintelę reikšmę įstatome į neigimo išvestinės formulę. Grįžtame į 1-ą lygį, kur turime suskaičiuoti antrojo atimties dėmens išvestinę. Tai atliekame patekdami į 2-ą lygį, kur apskaičiuojame konstantos išvestinę. Grįžę atgal į 1-ą lygį, įstatome abiejų atimties dėmenų išvestines į atimties išvestinės formulę ir štai turime rezultatą.



pav. 24. Diferencijavimo rezultatas

Diferencijavimo rezultatas yra pertvarkytas matematinės išraiškos operacijų medis. Tačiau, medis po diferencijavimo yra labai išsikerojęs, todėl tiesiog būtina jį supaprastinti, kaip ir parodyta pavyzdyje. Matematinės išraiškos operacijų medis taip pat supaprastinamas ir prieš diferencijavimą.

4. TYRIMAS

Šiame skyriuje apibrėžiamos tyrimo aplinkybės, įvairiais pjūviais lentelėmis ir grafikais išdėstomi atlikti eksperimentiniai tyrimai, įvertinami rezultatai. Šio skyriaus pagrindu yra daromos pagrindinės darbo išvados.

4.1. Tyrimo aplinkybės

4.1.1. Tyrimo objektas ir tikslas

Teoriniuose ir eksperimentiniuose tyrimuose tiriamas sukurtas produktas matematinėms išraiškoms pertvarkyti.

Tyrimo tikslas – ištirti matematinių išraiškų pertvarkymo, o konkrečiai diferencijavimo, greitį, palyginti jį su kitų sistemų greičiu, nustatyti sistemos ir duomenų charakteristikų kitimo įtaką skaičiavimų greičiui.

4.1.2. Tyrimo kryptys

Tyrimo metu buvo lyginamas sistemos greitis esant besikeičiančioms sąlygoms. Sąlygomis čia vadiname įvairių sistemos ir duomenų charakteristikų rinkinius. Išskyrėme tokias charakteristikas, kurių kitimo pagrindu galėtume atlikti tyrimą:

- matematinės išraiškos ilgis;
- matematinės išraiškos sudėtingumas;
- kompiuterinės technikos našumas;
- papildomi sistemos apkrovimai.

Pagrindine tyrimų kryptimi pasirinkome matematinės išraiškos ilgio įtakos diferencijavimo greičiui tyrimus.

Tyrime įtraukėme ir matematinių išraiškų pertvarkymo greičio palyginimą su kitais produktais.

4.1.3. Tyrimo įrankiai

Tyrimams atlikti buvo naudojama testavimo etape sukurto automatinio testavimo įrankio atskira modifikacija. Įrankio pagalba galima vykdyti matematinių išraiškų pertvarkymą ir greta rezultatų gauti pertvarkymams sugaištą laiką sekundėmis.

Taip pat buvo naudojami atskirai sukurti testinių atvejų generavimo ir testavimo rezultatų apibendrintos suvestinės sudarymo įrankiai.

4.1.4. Tyrimo duomenys

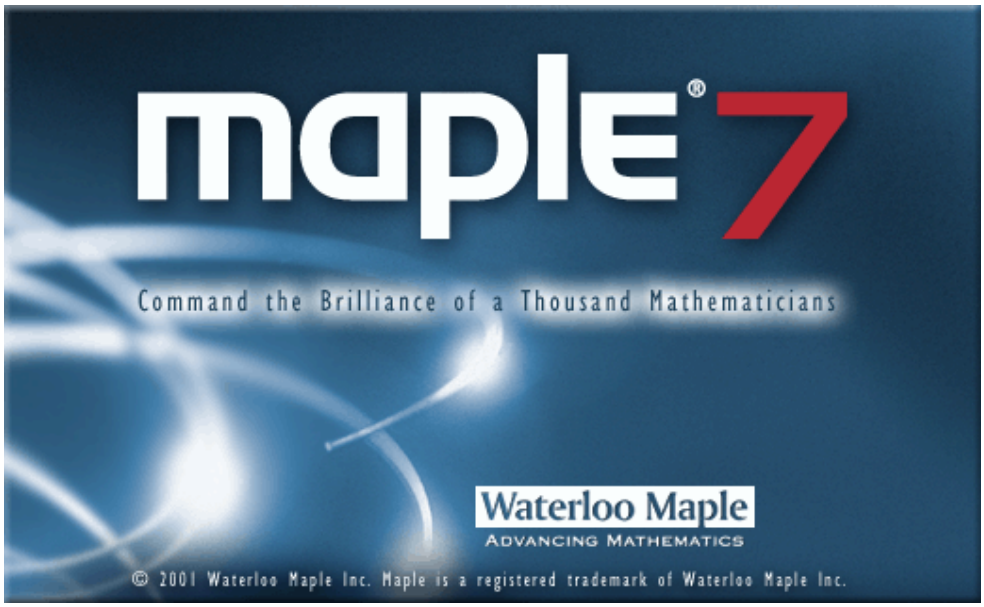
Tyrimams atlikti buvo naudojama testavimo etape sukauptų testinių atvejų biblioteka. Be šiai bibliotekai priklausančių matematinių išraiškų, papildomai buvo sugeneruotos dar kelios labai ilgos išraiškos.

Visose tyrimuose naudotose matematinėse išraiškose figūruoja įvairūs reikalavimuose numatyti operandai: neigimas, suma, atimtis, daugyba, dalyba, kėlimas laipsniu, sinusas ir kosinusas. Taip pat intensyviai naudojami ir skliausteliai. Tyrimo metu duomenims naudotos matematinės išraiškos buvo nuo 28.567 iki 28.851.913 simbolių ilgio, o pertvarkytos matematinės išraiškos buvo nuo 61.432 iki 62.372.342 simbolių ilgio.

4.1.5. Programinė įranga

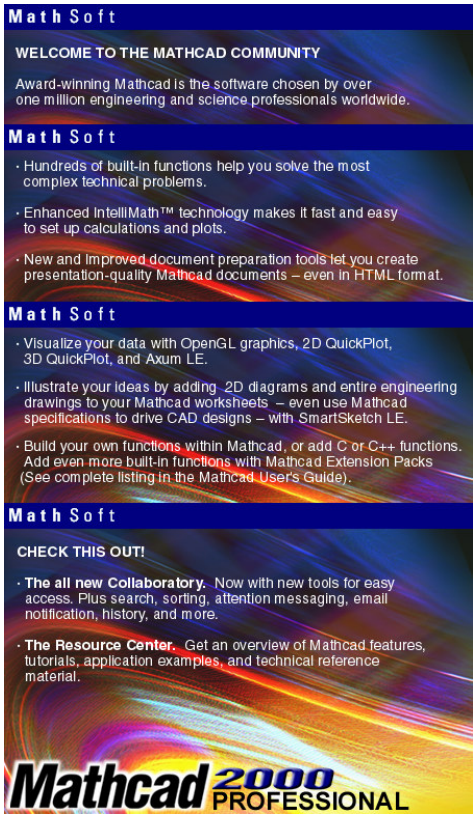
Sukurto produkto matematinių išraiškų pertvarkymo greitis gali būti lyginamas su įvairiais universaliais matematiniais produktais: Maple, Mathcad, MatLab, Mathematica ir pan.

Pirmiausia pasirinkome Maple, nes jį naudoja užsakovas. Mūsų sukurtoje sistemoje prie šio produkto yra priderintos matematinės išraiškos vaizdavimo taisyklės. Maple sukūrė kompanija Waterloo Maple, Inc. Buvo pasirinkta Maple 7 versija (naujausia versija – Maple 9.5).



pav. 25. Universalus matematinis produktas Maple 7

Taip pat keli testai buvo vykdomi ir su Mathcad. Tai populiariausias universalus matematinis produktas pasaulyje. Mathcad kitais tikslais sistemos kūrėjai yra naudoję ir anksčiau. Mathcad sukūrė kompanija MathSoft. Tyrimams buvo pasirinkta Mathcad 2000 Professional versija (naujausia versija – Mathcad 11.2).



pav. 26. Universalus matematinis produktas Mathcad 2000 Professional

Tačiau Mathcad priemonėmis nepavyko sukurti automatizuoto tyrimo įrankių, dėl to tyrimas šiuo produktu buvo nutrauktas. Tai neturėtų labai pakenkti tyrimų patikimumui, nes daugelis šaltinių teigia, kad tyrimams naudojama Maple sistema yra pati profesionaliausia.

4.1.6. Kompiuterinė technika

Tyrimai buvo atliekami naudojant Kauno Technologijos Universitetui priklausančio Informacinių technologijų plėtros instituto 320 auditorijos kompiuterių sistemą. Čia yra sukonfigūruotas 8 kompiuterių, kurių charakteristikos pateiktos žemiau esančioje lentelėje, klasteris.

lentelė 13. Klasterio kompiuterių charakteristikos

Eil. Nr.	Kompiuterio vardas	Archi-tektūra	Operacinė sistema	Procesorių kiekis	Procesorių dažnis	Atminties kiekis
1.	tulpe.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
2.	kopustas.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
3.	alyva.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
4.	zibute.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
5.	roze.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
6.	kietas.elen.ktu.lt	sun4u	SunOS 5.9	1 / 1	167 MHz	498.160 MB
7.	geras.elen.ktu.lt	sun4u	SunOS 5.9	1 / 1	270 MHz	498.184 MB
8.	astra.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.472 MB

Sukurtos sistemos tyrimų metu nebuvo naudojami 1-as, 6-as ir 7-as kompiuteriai. 1-asis buvo sugedęs, o 6-as ir 7-as nebuvo naudojami dėl jų našumo trūkumo bei operacinių sistemų nesuderinamumo.

4.2. Eksperimentai

4.2.1. Matematinių išraiškų ilgio įtaka

Matematinių išraiškų ilgio įtaka pertvarkymų greičiui buvo tiriama sistemoje naudojant nuo 2 iki 5 procesorių ir tiek pat lygiagrečių procesų. Buvo pasirinktos nuo 28.694 iki 8.621.540 simbolių ilgio matematinės išraiškos.

lentelė 14. Matematinių išraiškų ilgio įtaka

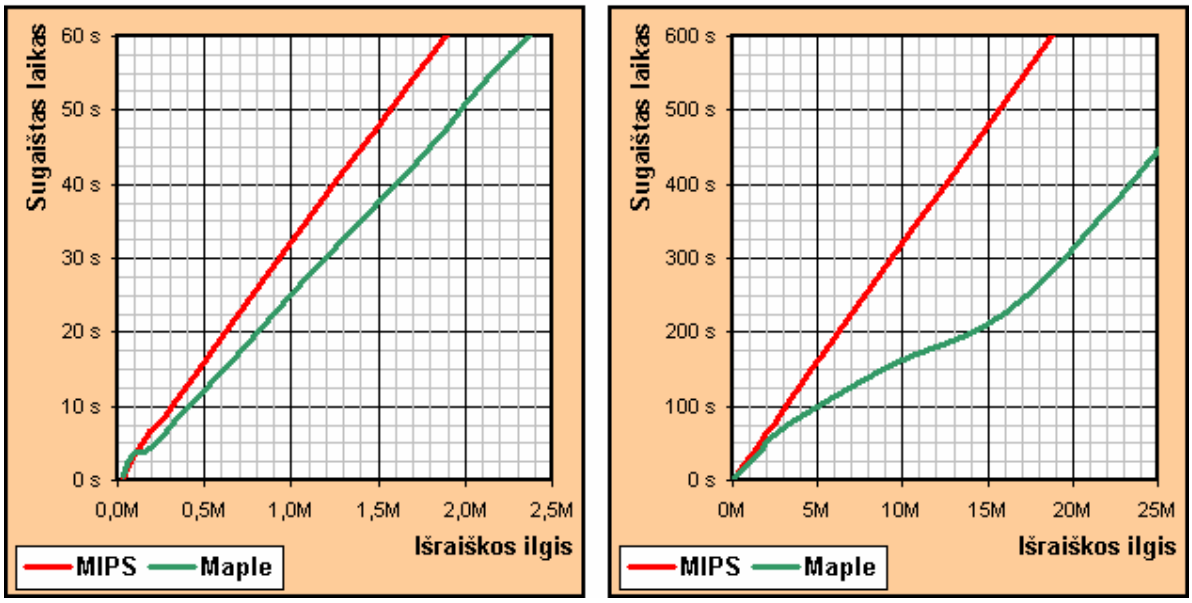
Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
2	2	expr9100.txt	86.160	expr9100.txt	185.720	3 s	4 s
2	2	expr9200.txt	171.825	expr9200.txt	372.238	6 s	4 s
2	2	expr9300.txt	285.527	expr9300.txt	619.136	9 s	7 s
2	2	expr9400.txt	865.470	expr9400.txt	1.876.867	28 s	22 s
2	2	expr9500.txt	1.721.990	expr9500.txt	3.726.800	55 s	43 s
2	2	expr9600.txt	2.878.752	expr9600.txt	6.240.910	92 s	69 s
2	2	expr9700.txt	8.621.540	expr9700.txt	18.678.390	278 s	146 s
2	2	expr9800.txt	17.305.880	expr9800.txt	37.409.525	551 s	249 s
2	2	expr9900.txt	28.845.371	expr9900.txt	62.350.917	926 s	550 s
3	3	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
3	3	expr9100.txt	86.160	expr9100.txt	185.720	2 s	4 s
3	3	expr9200.txt	171.825	expr9200.txt	372.238	3 s	4 s
3	3	expr9300.txt	285.527	expr9300.txt	619.136	4 s	7 s
3	3	expr9400.txt	865.470	expr9400.txt	1.876.867	14 s	22 s
3	3	expr9500.txt	1.721.990	expr9500.txt	3.726.800	29 s	43 s
3	3	expr9600.txt	2.878.752	expr9600.txt	6.240.910	49 s	69 s
3	3	expr9700.txt	8.621.540	expr9700.txt	18.678.390	146 s	146 s
3	3	expr9800.txt	17.305.880	expr9800.txt	37.409.525	294 s	249 s
3	3	expr9900.txt	28.845.371	expr9900.txt	62.350.917	489 s	550 s
4	4	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
4	4	expr9100.txt	86.160	expr9100.txt	185.720	1 s	4 s
4	4	expr9200.txt	171.825	expr9200.txt	372.238	2 s	4 s
4	4	expr9300.txt	285.527	expr9300.txt	619.136	3 s	7 s
4	4	expr9400.txt	865.470	expr9400.txt	1.876.867	10 s	22 s
4	4	expr9500.txt	1.721.990	expr9500.txt	3.726.800	20 s	43 s
4	4	expr9600.txt	2.878.752	expr9600.txt	6.240.910	34 s	69 s
4	4	expr9700.txt	8.621.540	expr9700.txt	18.678.390	100 s	146 s
4	4	expr9800.txt	17.305.880	expr9800.txt	37.409.525	202 s	249 s
4	4	expr9900.txt	28.845.371	expr9900.txt	62.350.917	335 s	550 s
5	5	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
5	5	expr9100.txt	86.160	expr9100.txt	185.720	1 s	4 s
5	5	expr9200.txt	171.825	expr9200.txt	372.238	2 s	4 s
5	5	expr9300.txt	285.527	expr9300.txt	619.136	2 s	7 s
5	5	expr9400.txt	865.470	expr9400.txt	1.876.867	8 s	22 s
5	5	expr9500.txt	1.721.990	expr9500.txt	3.726.800	16 s	43 s
5	5	expr9600.txt	2.878.752	expr9600.txt	6.240.910	26 s	69 s
5	5	expr9700.txt	8.621.540	expr9700.txt	18.678.390	74 s	146 s
5	5	expr9800.txt	17.305.880	expr9800.txt	37.409.525	156 s	249 s
5	5	expr9900.txt	28.845.371	expr9900.txt	62.350.917	262 s	550 s

Sekančiuose skyreliuose matematinių išraiškų ilgio įtaka pertvarkymų greičiui yra šiek tiek detalizuojama.

4.2.2. Matematinių išraiškų ilgio įtaka (2 naudojami procesoriai)

lentelė 15. Matematinių išraiškų ilgio įtaka (2 naudojami procesoriai)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
2	2	expr9100.txt	86.160	expr9100.txt	185.720	3 s	4 s
2	2	expr9200.txt	171.825	expr9200.txt	372.238	6 s	4 s
2	2	expr9300.txt	285.527	expr9300.txt	619.136	9 s	7 s
2	2	expr9400.txt	865.470	expr9400.txt	1.876.867	28 s	22 s
2	2	expr9500.txt	1.721.990	expr9500.txt	3.726.800	55 s	43 s
2	2	expr9600.txt	2.878.752	expr9600.txt	6.240.910	92 s	69 s
2	2	expr9700.txt	8.621.540	expr9700.txt	18.678.390	278 s	146 s
2	2	expr9800.txt	17.305.880	expr9800.txt	37.409.525	551 s	249 s
2	2	expr9900.txt	28.845.371	expr9900.txt	62.350.917	926 s	550 s

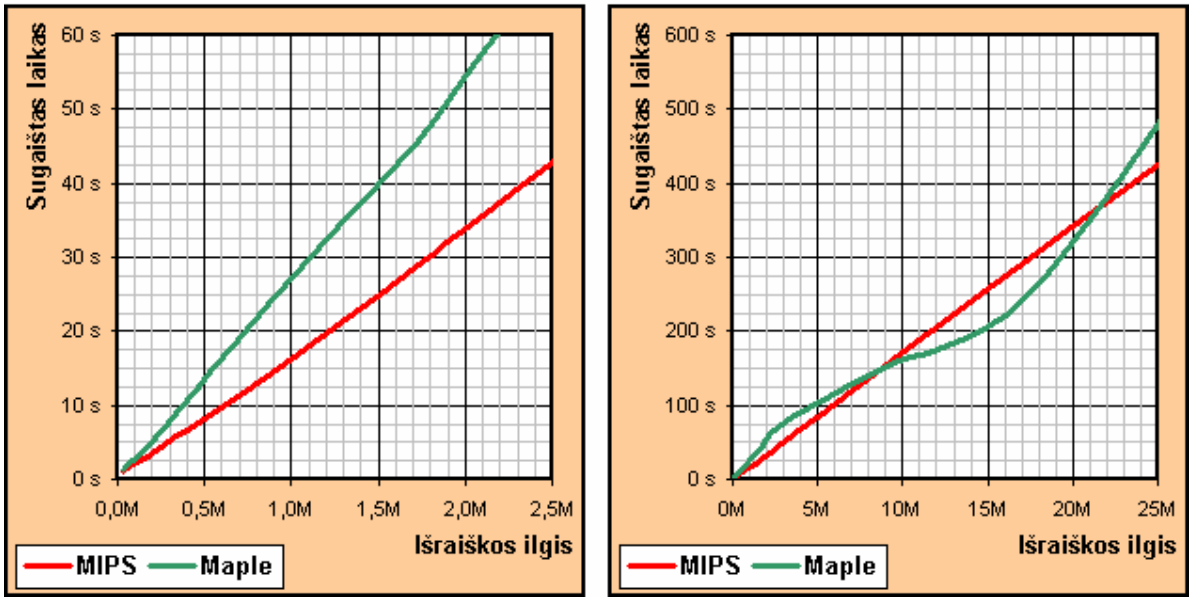


pav. 27. Matematinių išraiškų ilgio įtaka (2 naudojami procesoriai)

4.2.3. Matematinių išraiškų ilgio įtaka (3 naudojami procesoriai)

lentelė 16. Matematinių išraiškų ilgio įtaka (3 naudojami procesoriai)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
3	3	expr9001.txt	28.922	expr9001.txt	63.297	1 s	1 s
3	3	expr9101.txt	86.215	expr9101.txt	186.497	2 s	3 s
3	3	expr9201.txt	173.777	expr9201.txt	378.010	3 s	4 s
3	3	expr9301.txt	287.920	expr9301.txt	623.400	5 s	8 s
3	3	expr9401.txt	866.648	expr9401.txt	1.883.594	14 s	24 s
3	3	expr9501.txt	1.727.643	expr9501.txt	3.742.278	29 s	46 s
3	3	expr9601.txt	2.872.551	expr9601.txt	6.222.326	49 s	74 s
3	3	expr9701.txt	8.647.693	expr9701.txt	18.753.983	147 s	146 s
3	3	expr9801.txt	17.308.035	expr9801.txt	37.413.622	294 s	247 s
3	3	expr9901.txt	28.851.913	expr9901.txt	62.372.342	491 s	604 s

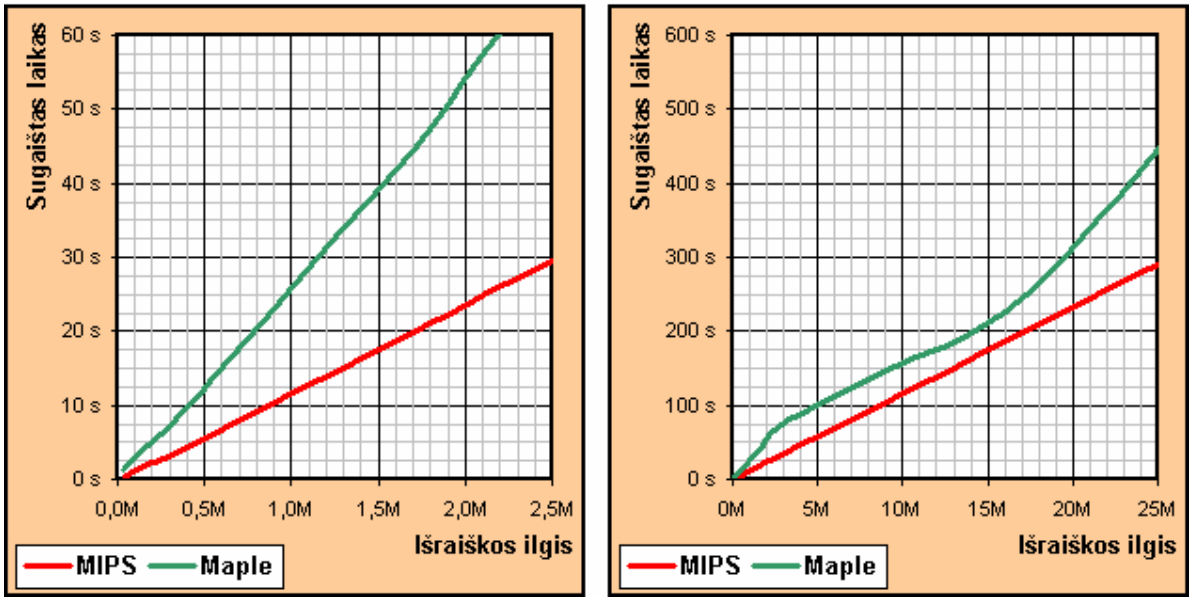


pav. 28. Matematinių išraiškų ilgio įtaka (3 naudojami procesoriai)

4.2.4. Matematinių išraiškų ilgio įtaka (4 naudojami procesoriai)

lentelė 17. Matematinių išraiškų ilgio įtaka (4 naudojami procesoriai)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
4	4	expr9002.txt	28.751	expr9002.txt	62.039	0 s	1 s
4	4	expr9102.txt	85.419	expr9102.txt	183.488	1 s	3 s
4	4	expr9202.txt	172.909	expr9202.txt	374.853	2 s	5 s
4	4	expr9302.txt	286.894	expr9302.txt	620.521	3 s	7 s
4	4	expr9402.txt	864.761	expr9402.txt	1.876.381	10 s	22 s
4	4	expr9502.txt	1.725.880	expr9502.txt	3.737.053	20 s	45 s
4	4	expr9602.txt	2.878.669	expr9602.txt	6.237.786	34 s	74 s
4	4	expr9702.txt	8.626.377	expr9702.txt	18.692.958	100 s	143 s
4	4	expr9800.txt	17.305.880	expr9800.txt	37.409.525	202 s	249 s
4	4	expr9900.txt	28.845.371	expr9900.txt	62.350.917	335 s	550 s

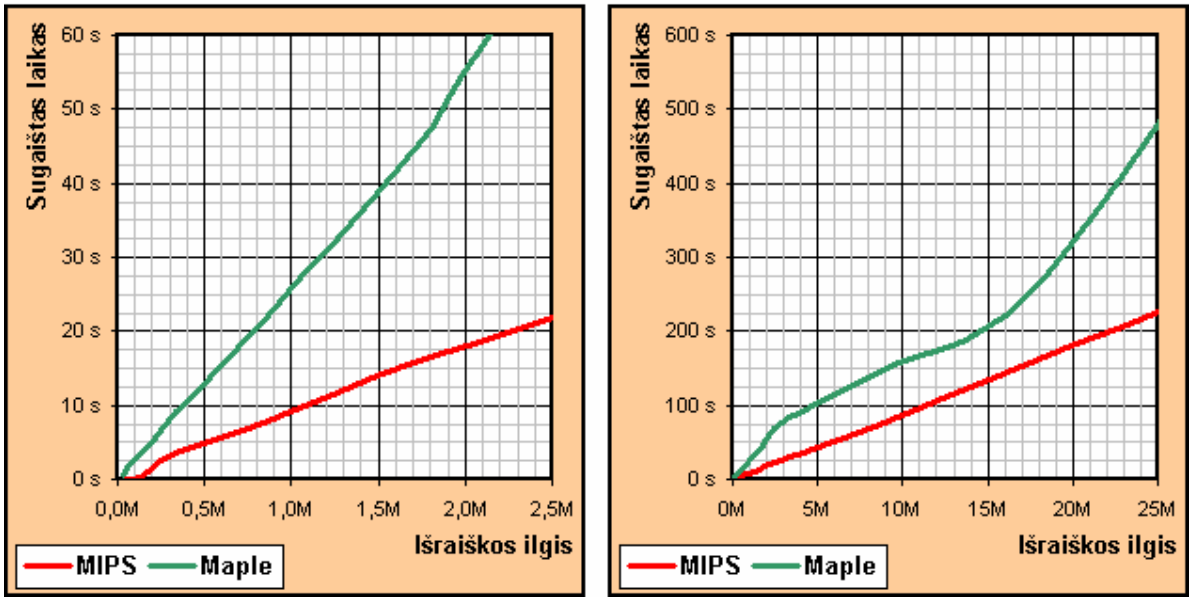


pav. 29. Matematinių išraiškų ilgio įtaka (4 naudojami procesoriai)

4.2.5. Matematinių išraiškų ilgio įtaka (5 naudojami procesoriai)

lentelė 18. Matematinių išraiškų ilgio įtaka (5 naudojami procesoriai)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
5	5	expr9003.txt	28.859	expr9003.txt	62.834	0 s	0 s
5	5	expr9103.txt	86.147	expr9103.txt	185.661	0 s	3 s
5	5	expr9203.txt	173.741	expr9203.txt	377.280	1 s	4 s
5	5	expr9303.txt	288.259	expr9303.txt	625.574	3 s	8 s
5	5	expr9403.txt	860.550	expr9403.txt	1.866.003	8 s	22 s
5	5	expr9503.txt	1.727.841	expr9503.txt	3.744.006	16 s	45 s
5	5	expr9603.txt	2.876.643	expr9603.txt	6.230.294	25 s	77 s
5	5	expr9703.txt	8.631.403	expr9703.txt	18.705.667	75 s	144 s
5	5	expr9801.txt	17.308.035	expr9801.txt	37.413.622	156 s	247 s
5	5	expr9901.txt	28.851.913	expr9901.txt	62.372.342	260 s	604 s

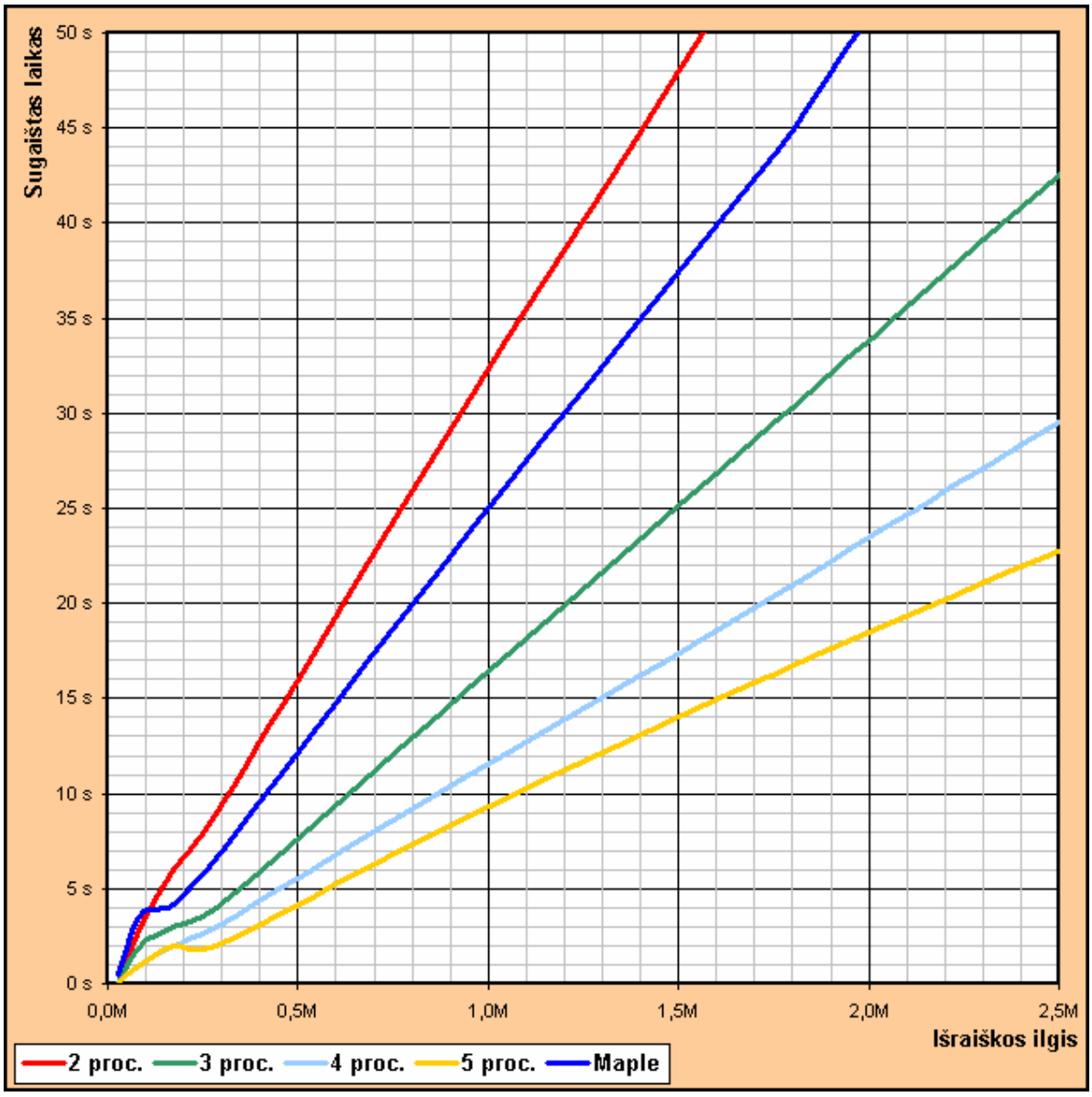


pav. 30. Matematinių išraiškų ilgio įtaka (5 naudojami procesoriai)

4.3. Rezultatų įvertinimas

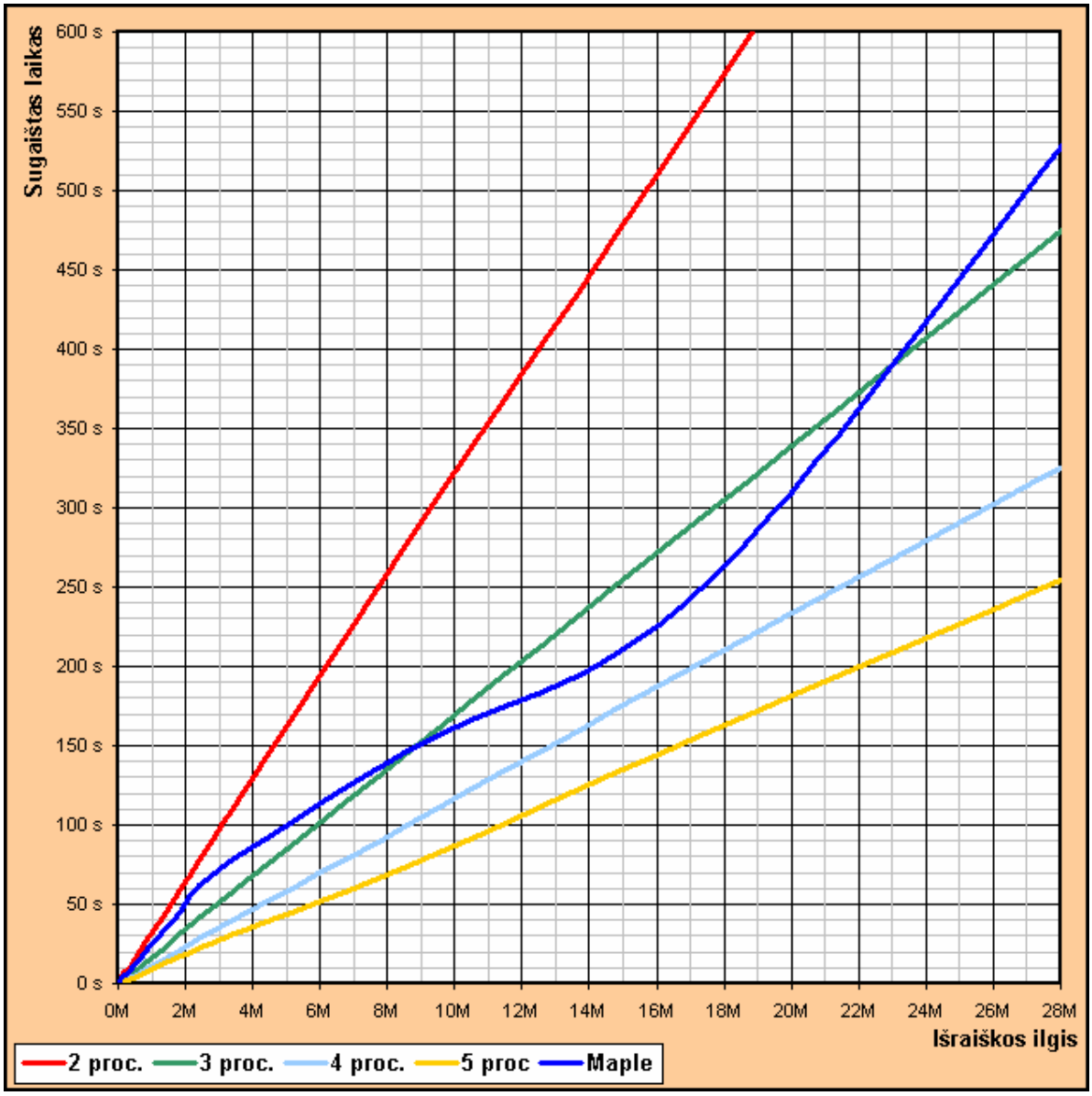
4.3.1. Apibendrinti rezultatai

Žemiau pateikiame apibendrintą matematinių išraiškų pertvarkymo greičio priklausomybę nuo matematinių išraiškų ilgio. Pirmasis grafikas rodo priklausomybę, kai matematinių išraiškų ilgis yra iki 2.500.000, antrasis grafikas – iki 28.000.000 simbolių.



pav. 31. Matematinių išraiškų ilgio (iki 2,5 mln. simbolių) įtaka

Atskiros grafikų kreivės rodo priklausomybę naudojant skirtingus kiekis procesorių. Mėlynoji kreivė rodo Maple sistemos greitį pertvarkant tas pačias matematines išraiškas naudojant vieną procesorių.



pav. 32. Matematinių išraiškų ilgio (iki 28 mln. simbolių) įtaka

4.3.2. Tendencijos

Iš abiejų grafikų matosi tokios tendencijos:

- matematinėms išraiškoms iki 100.000 simbolių sistemoje užtenka naudoti 2 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema;
- matematinėms išraiškos nuo 100.000 iki 8.500.000 simbolių sistemoje jau reikia naudoti 3 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema;
- matematinėms išraiškos nuo 8.500.000 iki 23.000.000 simbolių sistemoje jau reikia naudoti 4 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema;
- matematinėms išraiškos nuo 23.000.000 simbolių sistemoje užtenka naudoti 3 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema.

Pagal tendencijas matosi, kad Maple sistemą galime maždaug sulyginti su sukurta sistema, kai joje naudojami 3 procesoriai.

4.3.3. Rezultatų įvertinimas

Atlikus tyrimą išryškėjo sukurtos sistemos privalumai ir trūkumai. Pagrindinis trūkumas yra tai, kad vis dėl to nepavyko sukurti greitesnio matematinių išraiškų pertvarkymo algoritmo nei jis yra profesionalioje sistemoje Maple. Pagrindinis privalumas yra tai, kad vis dėl to pavyko sukurti tokią sistemą, kurioje naudojant 4 ir daugiau procesorių, būtų galima efektyviau nei iki šiol spręsti taikymo srities uždavinius.

4.3.4. Rezultatų gerinimas

Žinant, kad buvo siekiama padaryti specializuotą matematinių išraiškų pertvarkymo algoritmą, dar galima tikėtis jo pagreitėjimo. Pirmiausia reikia atlikti dar išsamesnius atskirų algoritmo dalių tyrimus. Tokių tyrimų tikslas – išryškinti tas algoritmo vietas, kurios užtrunka nepagrįstai ilgai laiko. Tyrimų rezultatų pagrindu turi būti atliekamos algoritmo modifikavimas. Po modifikacijų tyrimai būtinai turi būti pakartoti.

5. IŠVADOS

Šiame skyriuje užfiksuojama sukurta sistema, pakartojami pagrindiniai tyrimų rezultatai, jų pagrindu padaromos ir išdėstomos darbo išvados. Skyriaus gale apibrėžiamos galimos produkto plėtros kryptys, pateikiamos rekomendacijos naudotojams.

5.1. Sukurta sistema

Per 16 mėnesių buvo sukurta ir pilnai dokumentuota matematinių išraiškų pertvarkymo programinė įranga. Sistema skirta matematinių išraiškų pertvarkymui lygiagrečiai naudojant kelis vieno procesoriaus kompiuterius, sujungtus į vietinį tinklą. Šiuo metu sistema pritaikyta matematinių išraiškų diferencijavimui.

5.2. Pagrindiniai tyrimų rezultatai

Tyrimų metu nustatėme, kad sukurta sistema yra itin efektyvi, kai sistemoje naudojami bent 3 procesoriai. Sistemos efektyvumas yra mažesnis esant 2 procesoriams, nes naudojams tik vienas vykdantysis procesas antrajame procesoriuje. Matematinių išraiškų ilgis pertvarkymų greitį įtakoja beveik tiesiškai proporcingai.

5.3. Plėtros kryptys

Darbo tiksluose buvo numatytas susiaurintas funkcionalumas. Dėl to lieka nemenka erdvė funkcionalumui plėsti, t. y. diferencijavimo funkcijų papildymas ir kitokie pertvarkymai, pvz. integravimas. Kadangi sistema yra komplikuota ir sudėtinga, liko erdvės algoritmams optimizuoti, greičiui bei patikimumui didinti.

Produktas neturi patogios grafinės vartotojo sąsajos, todėl sekančiose versijose galima būtų ją sukurti pagal sistemos naudotojų rekomendacijas.

5.4. Algoritmų tobulinimas

Kalbant apie matematinių išraiškų pertvarkymo algoritmo tobulinimą, pirmiausia sistemos kūrėjams reikia detaliai ištirti atskiras algoritmo dalis, kad nustatyti nepagrįstai ilgai veikiančias vietas. Po kiekvienos algoritmo modifikacijos būtina kartoti tyrimus, kad pastebėti modifikacijų įtaką ir tendencijas.

Projektuojant naujus algoritmus, galima bandyti keisti ne tik algoritmo pagrindą, bet ir duomenų struktūras, galbūt net sąsajos tipą bei valdymą. Naujus algoritmus reiktų lyginti su esamais algoritmais taikant tą pačią detalaus tyrimo procedūrą.

5.5. Rekomendacijos

Žemiau lentelės forma pateiktos rekomendacijos sistemos naudotojams kada kurią sistemą naudoti.

lentelė 19. Kada kurią sistemą naudoti

Matematinės išraiškos ilgis		Naudojamų procesorių kiekis				
nuo	**iki**	**1**	**2**	**3**	**4**	**5**
0	100.000	Maple	MIPS	MIPS	MIPS	MIPS
100.000	8.500.000	Maple	Maple	MIPS	MIPS	MIPS
8.500.000	23.000.000	Maple	Maple	Maple	MIPS	MIPS
23.000.000	∞	Maple	Maple	MIPS	MIPS	MIPS

6. LITERATŪRA

Šiame skyriuje pateikiamas literatūros sąrašas, nurodomi papildomai naudoti dokumentai.

6.1. Literatūra

[1] Problem Solving Environments [interaktyvus]. – [žiūrėta 2002 10 24]. Prieiga per internetą: <www-cgi.cs.purdue.edu/cgi-bin/acc/pses.cgi>

[2] Distributed Maple [interaktyvus]. – [žiūrėta 2002-10-17]. Prieiga per internetą: <http://www.risc.uni-linz.ac.at/software/distmaple>

[3] Maple [interaktyvus]. – [žiūrėta 2002-10-10]. Prieiga per internetą: <http://www.maplesoft.com/products/maple7>

[4] The Maple Reporter. 2001.

[5] Ross S. S. Maple 6: The Analytical Engine. 2000.

[6] Message Passing Interface Forum: Document for a Standard Message-Passing Paradigm. – 1993.

[7] MPI forum [interaktyvus]. – [žiūrėta 2002-10-10]. Prieiga per internetą: <http://www.mpi-forum.org>

[8] Grossman F., Klerer R. J., Klerer M. A language for highlevel programming of mathematical applications: In Proceedings of the International Conference on Computer Languages. – Miami Beach, FL, USA: 1988.

[9] Chang S. K. A method for the structural analysis of 2-D mathematical expressions: Information Sciences 2. – 1970.

[10] Lee H. J., Lee M. C. Understanding mathematical expressions using procedure-oriented transformation: Pattern Recognition. – 1994.

[11] Earley J. An efficient context-free parsing algorithm. – Communications of ACM, 6, 8: 1970.

[12] Aho A. V., Peterson T. G. A minimum-distance error-correcting parser for context-free languages. – Journal on Computing, 1, 4: SIAM, 1972.

[13] Stolcke A. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. – Computational linguistics, 21, 2: 1995.

[14] Okamoto M., Miao B. Recognition of mathematical expressions by using the layout structures of symbols. – Proceedings of the First International Conference on Document Analysis and Recognition. – Saint-Malo, France: 1991.

6.2. Naudoti dokumentai

lentelė 20. Naudoti dokumentai

Dokumentas	Data	Dokumento autorius
Projekto paraiška	2002.11.27	A. Vaškevičienė, V. Vaškevičius
Reikalavimų specifikacija	2003.03.17	A. Vaškevičienė, V. Vaškevičius
Architektūros specifikacija	2003.04.15	A. Vaškevičienė, V. Vaškevičius
Detalios architektūros specifikacija	2003.05.23	A. Vaškevičienė, V. Vaškevičius
Testavimo planas	2003.09.28	A. Vaškevičienė, V. Vaškevičius
Vartotojo dokumentacija	2003.12.11	A. Vaškevičienė, V. Vaškevičius
Dokumentacija	2004.01.05	A. Vaškevičienė, V. Vaškevičius

7. SANTRUMPŲ IR TERMINŲ ŽODYNAI

Šiame skyriuje atskirai pateikiami santrumpų ir terminų žodynai.

7.1. Santrumpos

lentelė 21. Santrumpos

Santrumpa	Apibūdinimas
MIPS	Matematinių išraiškų pertvarkymo sistema
LFVS	Lygiagretaus funkcionavimo valdymo sistema
MPI	Message Passing Interface – pranešimų perdavimo sąsaja
Maple	Universalus simbolinės matematikos produktas (gamintojas – Waterloo Maple, Inc). Naudojama Maple 7 versija.
Mathcad	Universalus simbolinės matematikos produktas (gamintojas – MathSoft). Naudojama Mathcad 2000 Professional.

7.2. Terminai

lentelė 22. Terminai

Terminas	Terminas anglų kalba (English)	Apibūdinimas
Matematinė išraiška	Mathematical expression	Matematinis reiškinys sudarytas iš skaičių, kintamųjų, aritmetinių, trigonometrinių, logaritminių funkcijų
Diferencijavimas	Differentiation	Simbolinio diferencialo (išvestinės) nustatymas
Nuoseklieji algoritmas	Sequential algorithm	Standartinis algoritmas, vykdomas viename procese
Lygiagretieji algoritmas	Parallel algorithm	Algoritmas, kurio atskiros dalys lygiagrečiai vienu metu vykdomos atskiruose procesuose
Lygiagretusis funkcionavimas	Parallel operation	Kelių procesų veikimas vienu metu

PRIEDAI

Priedas A. Magistro darbo dokumentas

Dokumento paskirtis

Dokumentas skirtas magistro darbui ir tyrimui aprašyti. Taip pat šis dokumentas yra Kauno technologijos universiteto Informatikos fakulteto Programų inžinerijos katedros modulio "Magistro baigiamasis darbas" atsiskaitymo ataskaita.

Dokumento autoriai

Šį magistro darbo dokumentą ruošė Kauno technologijos universiteto Informatikos fakulteto Programų inžinerijos katedros IFM-8/2 grupės magistrantė Aistė Vaškevičienė ir docentas dr. Romas Marcinkevičius.

Dokumentą maketavo Aistė Vaškevičienė.

Dokumento istorija

Magistro darbas pradėtas rengti 2004 m. kovo 1 d.

lentelė 23. Dokumento istorija

Data	Versija	Aprašymas	Redaktorius
2004.04.18	1.0	Pirminės dokumento struktūros sudarymas	A. Vaškevičienė
2004.05.05	1.1	Pradinis dokumento parengimas	A. Vaškevičienė
2004.05.07	1.2	Lietuvių kalbos konsultantės redakcija	J. Mikelionienė
2004.05.10	1.3	Įvairūs pataisymai ir papildymai	A. Vaškevičienė
2004.05.12	1.4	Darbo vadovo redakcija	R. Marcinkevičius
2004.05.13	1.5	Įvairūs pataisymai ir papildymai	A. Vaškevičienė
2004.05.16	1.6	Tyrimų skyrius	A. Vaškevičienė
2004.05.19	1.7	Analizės skyrius	A. Vaškevičienė
2004.05.20	1.8	Lietuvių kalbos konsultantės redakcija	J. Mikelionienė
2004.05.21	1.9	Darbo vadovo redakcija	R. Marcinkevičius
2004.05.24	2.0	Galutinė redakcija	A. Vaškevičienė