KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Evaldas Užpalis

Lygiagrečiųjų simbolinių skaičiavimų programinė įranga

Magistro darbas

Darbo vadovas: doc. dr. Romas Marcinkevičius

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Evaldas Užpalis

Lygiagrečiųjų simbolinių skaičiavimų programinė įranga

Informatikos mokslo magistro baigiamasis darbas

Darbo vadovas
Programų inžinerijos katedros docentas
dr. Romas Marcinkevičius
2009-05-25

Recenzentas
Kompiuterių katedros docentas
dr. Stasys Maciulevičius
2009-05-25

Atliko
IFM-3/2 grupės studentas
Evaldas Užpalis
2009-05-22

Kaunas, 2009

Turinys

Paveikslėlių sąrašas

Lentelių sąrašas

Software for parallel symbolic computing

SUMMARY

There are two methods of mathematical problems solving: the digital, and symbolic. Symbolic solutions manipulate symbolic objects, such as logical or algebraic formulas, rules or programs. In contrast to the digital solution, the main purpose of the symbolic calculations is the symbolic simplification of mathematical expressions. In most cases, the final answer is rational number, or formula, and therefore symbolic calculations can be used: (1)

- to identify the precise solution of the mathematical problem,
- to simplify the mathematical model.

For calculation of small mathematical expression it is enough one computer. But there are expressions which need more then one computer memory capacity or processing power. In these cases best solution is parallel calculations in computer cluster.

The main problem of parallel calculations is the efficiency of distribution algorithm. This work presents experimental studies of one distribution algorithm and of several it‘s modifications.

1 Įvadas

1.1 Dokumento paskirtis

Šio dokumento paskirtis yra pateikti lygiagrečių simbolinių skaičiavimų programinės įrangos inžinerinių problemų sprendimų apžvalgą ir panaudotų algoritmų efektyvumo tyrimo rezultatus.

Antrajame yra skyriuje apžvelgiamos programinei įrangai sukurti reikalingos technologijos. Trečiajame skyriuje pateikiami realizacijos sprendimai ir ketvirtajame- tyrimų ir eksperimentų rezultatai.

1.2 Santrauka

Egzistuoja du matematinių problemų sprendimo būdai: skaitmeninis ir simbolinis. Simbolinis sprendimo būdas manipuliuoja simboliniais objektais, tokiais kaip loginės ar algebrinės formulės, taisyklės ar programos. Priešingai nei skaitmeninis būdas, pagrindinis simbolinių skaičiavimų tikslas yra matematinės išraiškos supaprastinimas. Dažniausiai galutinis atsakymas būna racionalusis skaičius arba formulė, todėl simboliniai skaičiavimai gali būti naudojami (1):

- surasti tikslų matematinės problemos sprendimą,
- supaprastinti matematinį modelį.

Nedidelės apimties matematinėms išraiškoms supaprastinti užtenka ir vieno kompiuterio, tačiau yra tokių išraiškų, kurioms supaprastinti nebeužtenka vieno kompiuterio atminties ar procesoriaus, todėl geriausias sprendimas šioje situacijoje yra lygiagretieji skaičiavimai kompiuterių klasteryje.

Pagrindinė problema lygiagrečiuose skaičiavimuose yra duomenų paskirstymo algoritmo efektyvumas. Šiame darbe yra pateikti vieno iš paskirstymo algoritmų ir kelių jo modifikacijų eksperimentiniai tyrimai.

2 Analizė

2.1 Darbo tikslas

Darbo tikslas yra sukurti matematinių išraiškų supaprastinimo programinę įrangą ir ištirti jos efektyvumą.

Matematinių išraiškų supaprastinimas yra reikalingas norint gauti trumpesnę ar aiškesnę išraišką, lengviau nustatyti matematinės funkcijos klasę.

Nedidelės apimties funkcijoms supaprastinti užtenka ir vieno kompiuterio, tačiau yra tokių funkcijų, kurioms supaprastinti nebeužtenka vieno kompiuterio atminties ar procesoriaus, todėl geriausias sprendimas šioje situacijoje yra skaičiavimai kompiuterių klasteryje.

2.2 Problemos sprendimas užsienyje

Egzistuoja bent keli komerciniai sprendimai. Vieni iš jų yra HPC-Grid for Maple ir Parallel Computing Toolkit.

2.2.1 HPC-Grid for Maple

Sukurtas internetinių sprendimų kompanijos- Nobilitas GmbH. HPC-Grid leidžia vartotojams paskirstyti skaičiavimus kompiuterių tinkle, superkompiuteryje arba kompiuteryje su daug procesorių. Veikia Windows ir Linux operacinėse sistemose .

Šios programinės įrangos savybės (2):

- naudoja Maple paketą,
- bendravimui tarp procesų naudoja žinutes panašias į MPI,
- palaiko tinklus sudarytus iš potinklių, kurie naudoja skirtingus protokolus.

2.2.2 Parallel Computing Toolkit

Sukurtas Wolfram Research kompanijos. Veikia Windows, MacOS ir Linux operacinėse sistemose. Šios programinės įrangos savybės (3):

- skaičiavimams naudoja Mathematica paketą,
- paralelizmas įgyvendintas Mathematica kalbos lygyje,
- komunikavimui naudoja MathLink protokolą,
- efektyvus ir prisitaikantis procesorių apkrovų valdymas,
- palaiko ad-hoc grupavimą,
- galima rašyti nuo operacinės sistemos nepriklausomą kodą,
- turi automatinį klaidų ištaisymo mechanizmą .

2.2.3 Matematinių paketų palyginimas

Palyginsime 2 matematinius paketus „Maple“ ir „Mathematica“(9).

1 lentelė. Matematinių paketų palyginimas

Parametras	Maple	Mathematica
Funkcijos		
Beselio funkcijos	+	+
Logaritmai	+	+
Trigonometrinės funkcijos	+	+
Matricos	+	+
Lygčių sprendimas	+	+
Integravimas	+	+
Diferencijavimas	+	+
Furjė transformacijos	+	+
Greičio palyginimas		
Ciklų testas 15.000 x 15.000	259 s	323 s
1000000 atsitiktinių reikšmių rikiavimas	236 s	38 s
Greitoji Furjė transformacija naudojant 1048576 atsitiktinių reikšmių	47 s	16 s
1500x1500 matricos determinanto skaičiavimas	43 s	24 s
Atvirkštinės 1500x1500 matricos radimas	150. s	74 s
Choleskio dekompozija 1500x1500 matricai	414 s	19 s
10000000 Fibonačio skaičių radimas	821 s	2 s
Gama funkcijos skaičiavimas 1500x1500 matricai	10081 s	252 s
Gauso klaidų funkcijos radimas 1500x1500 matricai	4815 s	272 s

2.3 Problemos sprendimas Lietuvoje

Lietuvos įmonės matematinės programinės įrangos nesiūlo. Vienintelis darbas, kurį pavyko surasti yra „Matematinių išraiškų pertvarkymo programinė įranga“ sukurtas magistrantų Aistės ir Vytauto Vaškevičių. Ši programinė įranga atlieka skaičiavimus kompiuterių klasteryje. Jos pagrindas yra „Maple“ matematinis paketas (8).

2.4 Funkcijų identiškumo problema

Tarkim, kad $f_1(x)$ yra pradinė funkcija, o $f_2(x)$- funkcijos $f_1(x)$ supaprastinimo rezultatas.

Teiginys $f_1(x) \equiv f_2(x)$ yra teisingas tik tuo atveju, jei $\forall x \in R: f_1(x) = f_2(x)$. Egzistuoja funkcijos, kurioms negalioja anksčiau minėtas teiginys. Pvz.: $f_1(x) = \frac{x^2-4}{x-2}$, tai supaprastinus $f_2(x) = x + 2$. Be papildomų funkcijos $f_2(x)$ apribojimų teiginys negalios, nes $f_1(2) = \infty$, o $f_2(2) = 4$.

2.5 Kintamųjų ir matematinių operacijų atpažinimas

2.5.1 Gramatika

Formali gramatika yra taisyklių rinkinys, kuris apibrėžia kokios simbolių eilutės (žodžiai) yra tinkamos. Gramatika tik nustato žodžių vietą sakinyje.

Gramatika susideda iš žodžių pakeitimo taisyklių rinkinio ir pradinio simbolio.

Dažniausiai gramatika yra apibrėžiama kaip kalbos generatorius, bet ji gali būti panaudota patikrinti ar duotas žodis priklauso nustatytai gramatikai.

Klasikinė generuojančių gramatika, suformuluota Naom Chomsky, susideda iš:

- Baigtinės neterminalinių simbolių aibės *N*.
- Baigtinės terminalinių simbolių aibės Σ, $\Sigma \cap N = \emptyset$.
- Baigtinės formavimo taisyklių aibės *P,* kur kiekviena taisyklė yra tokios formos: $(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$.
- Pradinio simbolio $S \in N$.

Yra kelių rūšių gramatikos:

- Nuo konteksto priklausanti gramatika

  Tai gramatika, kurioje kiekviena taisyklė yra tokios formos:

  $\alpha A \beta \rightarrow \alpha \gamma \beta$

  Kur $A \in N$, $\alpha, \beta \in (\Sigma \cup N)^*$, $\gamma \in (\Sigma \cup N)^+$

- Nuo konteksto nepriklausanti gramatika

  Tai gramatika, kurioje kiekviena taisyklė yra tokios formos:

  $V \rightarrow w$

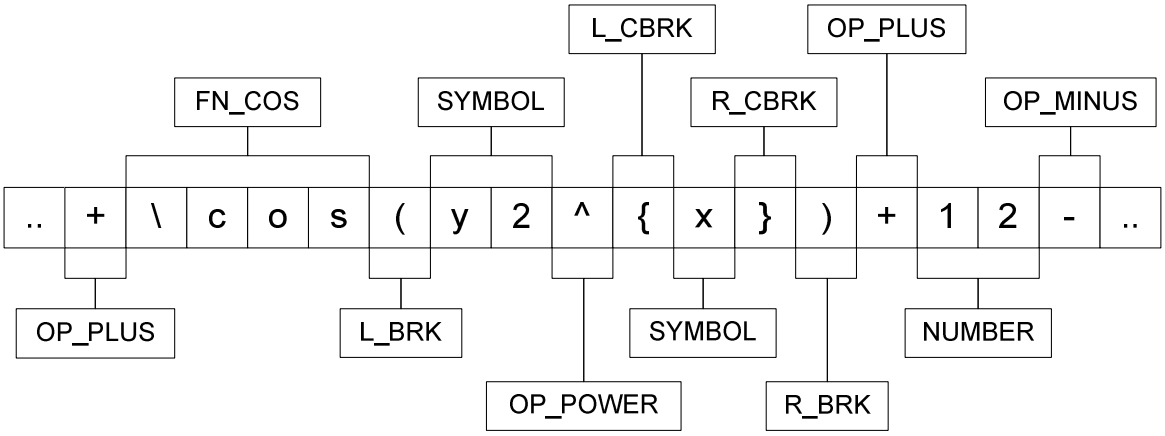
V - yra vienas neterminalinis simbolis ($V \in N$).

w- terminalinių ir/arba neterminalinių simbolių eilutė ($w \in (\Sigma \cup N)^*$).

Terminas “nuo konteksto nepriklausanti” reiškia, kad neterminaliniai simboliai gali būti perrašyti neatsižvelgiant į jų kontekstą (7).

2.5.2 Leksemų analizė

Leksemų analizės metu tekstas yra sudalinamas į teksto blokus vadinamus leksemomis. Tam, kad atliekant sintaksinę analizę nereikėtų nagrinėti visų galimų leksemų reikšmių, kiekvienai leksemai yra priskiriamas tipas (7).



1 pav. Leksemų analizės pavyzdys

2.5.3 Sintaksinė analizė

Sintaksinė analizė analizuoja leksemas ir nustato jų struktūrą pagal duotą gramatiką. Šio proceso rezultatas yra sintaksinės analizės medis.

Yra dviejų rūšių analizatoriai (7):

- Iš viršaus apačion (angl. Top-down parsing)

Iš viršaus apačion analizė- tai nežinomų duomenų ryšių analizė, kuriant galimas analizės medžio struktūras ir tikrinant ar duomenys atitinka jas. Analizė pradedama nuo startinio simbolio ir einama iki leksemų. Šios rūšies analizatorių darbo trukmė yra eksponentiškai priklausoma nuo įėjimo duomenų kiekio.

Pvz.:

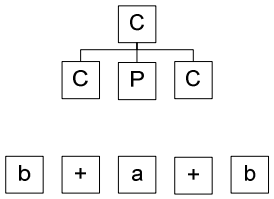
Turime gramatiką:

1. <P>::=’+’
2. <C>::=’a’
3. <C>::=’b’
4. <C>::=<C><P><C>

Startinis simbolis <C>.

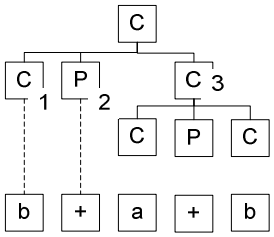
Reikia sudaryti eilutės „b+a+b“ analizės medį.

Startinį simbolį galime išskleisti pasinaudoję 2, 3 arba 4 taisykles. Tačiau tinkama yra tik 4-oji, nes išskleidę pagal antrąją gausime, kad reikalingas simbolis ‘a’, o turime ‘b’, išskleidę pagal trečiąją, užbaigsim analizę neišanalizavę visos eilutės.



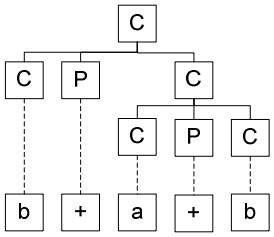
2 pav. „Iš viršaus-žemyn" sintaksinė analizė. 1 žingsnis

Pirmasis simbolis išsiskleidžia pagal 3 taisyklę, antrasis pagal 1. Trečiąjį simbolį išskleidžiame taip pat kaip ir pirmajame žingsnyje.



3 pav. „Iš viršaus-žemyn" sintaksinė analizė. 2 žingsnis

Ir gauname galutinį analizės medį.



4 pav. „Iš viršaus-žemyn" sintaksinė analizė. 3 žingsnis

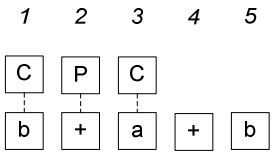
- Iš apačios į viršų (angl. Bottom-up parsing)

Iš apačios į viršų analizė- tai nežinomų duomenų ryšių analizė, kurios metu pirmiausia identifikuojami patys fundamentaliausi vienetai ir tada bandoma išvesti aukštesnio lygio struktūras. Jos metu sintaksinės analizės medį bandoma sudaryti nuo šaknų iki viršūnės. Šios rūšies analizatorių darbo trukmė yra polinomiškai priklausoma nuo įėjimo duomenų kiekio.

Pvz.:

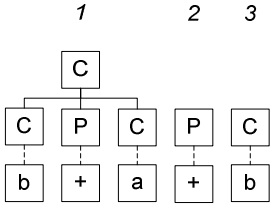
Panaudosime užduotį aprašytą „iš viršaus žemyn“ tipo analizatoriaus pavyzdyje.

Pirmąjį simbolį ‘b’ galima pakeisti pagal 3 taisyklę, simbolį ‘+‘ pagal 1, simbolį ‘a‘, pagal 2 taisyklę. Pirmuosius 3 simbolius galima pakeisti pagal 4 taisyklę.



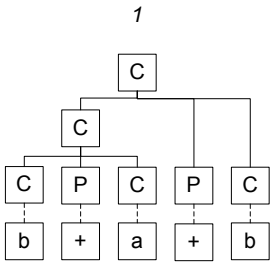
5 pav. „Iš apačios-aukštyn" sintaksinė analizė. 1 žingsnis

Antras ir trečias simboliai pakeičiami atitinkamai pagal 1 ir 3 taisykles. Pirmuosius 3 simbolius galima pakeisti pagal 4 taisyklę.



6 pav. „Iš apačios-aukštyn" sintaksinė analizė. 2 žingsnis

Gaunamas galutinis analizės medis.

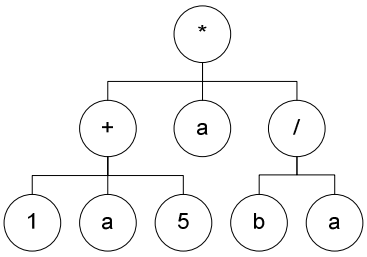


7 pav. „Iš apačios-aukštyn" sintaksinė analizė. 3 žingsnis

2.6 Matematinių funkcijų medis

Matematinė išraiška gali būti aprašoma matematinių funkcijų medžiu (operacijų medžiu). Pvz.:

$$(1 + a + 5) * a * \frac{b}{a} =$$



8 pav. Funkcijų medis

Kiekvienas medžio mazgas atvaizduoja matematinę funkciją, o iš mazgo išeinančios šakos- tos funkcijos argumentus. Toks funkcijų medis yra formuojamas iš sintaksinės analizės medžio. Skaičiavimai atliekami nuo šaknų iki viršūnės.

2.7 MPI

MPI yra nuo programavimo kalbos nepriklausantis komunikacijų standartas, naudojamas kurti lygiagrečių skaičiavimų programas.

MPI tikslai yra didelė sparta, išplečiamumas ir pernešamumas. MPI yra dominuojantis modelis didelės spartos skaičiavimuose. MPI nėra patvirtinta jokios stambios standartų įstaigos, bet vis tiek jis tapo de facto standartu komunikacijoms tarp procesų, kurie vykdo lygiagrečią programą su paskirstyta atmintimi.

MPI pranašumai prieš senesnes duomenų perdavimo bibliotekas yra pernešamumas (kadangi MPI yra realizuotas beveik visoms paskirstytos atminties architektūroms) ir greitis (nes kiekviena realizacija iš principo yra optimizuota techninei įrangai, kurioje ji veikia).

MPI sąsaja yra skirta pateikti esmines virtualios topologijos, sinchronizacijos ir komunikavimo funkcijas tarp procesų, nuo kalbos nepriklausomu būdu. Dažniausiai didžiausiai spartai pasiekti kiekvienam procesoriui (ar branduoliui) priskiriamas tik vienas procesas.

Nors MPI standartas apibrėžia daugiau nei 300 funkcijų, tačiau didžiajai daliai programų pakanka ir 10 paprasčiausių funkcijų (4).

Galima paminėti ir kelias MPI standarto realizacijas:

- OpenMPI- realizuotas MPI-2 standartas, skirtas Linux, OS X ir Solaris operacinėms sistemoms. Galima naudoti su C, C++ ir Fortran kalbomis. Ši realizacija atsirado sujungus FT-MPI, LA-MPI ir LAM/MPI (5).
- MPICH2- realizuotas MPI-2 standartas, skirtas Linux, Windows ir operacinėms sistemoms. Galima naudoti su C, C++, Fortran 77 ir Fortran 90 kalbomis (10).

2.8 Priimti sprendimai

2.8.1 Gramatinė analizė

Matematinei išraiškai užrašyti panaudotas LaTeX sintaksės poaibis. LaTeX buvo pasirinktas, kadangi juo naudojasi daugelis matematikų, inžinierių ir kitų techninės pakraipos specialistų (11). Detalus naudojamos gramatikos aprašas yra pateiktas skyriuje „8.1 Gramatika“.

Išraiškos gramatinei analizei panaudotas „iš apačios- aukštyn“ analizatorius, kadangi jo sudėtingumas yra polinominis.

2.8.2 Sistemos veikimo principas

Programa veikia kompiuterių klasteryje. Lygiagrečiųjų procesų valdymui panaudota MPI standarto realizacija OpenMPI.

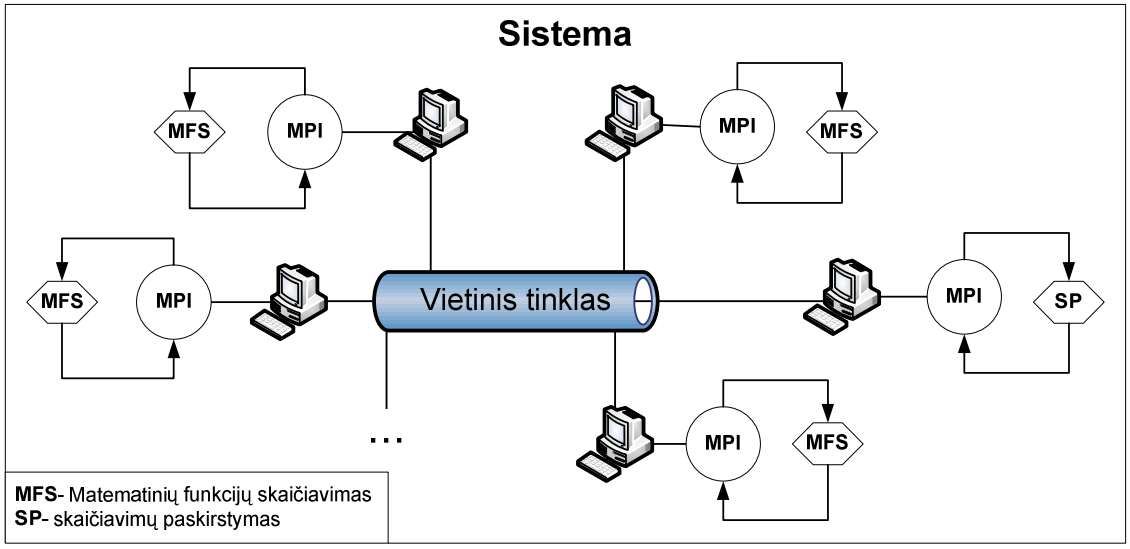
Yra dviejų rūšių procesai: valdantysis ir skaičiuojantys. Valdantysis procesas yra tik vienas, o visi likę procesai- skaičiuojantieji.

Valdančiojo proceso funkcijos:

- Matematinės išraiškos įkėlimas.
- Struktūros patikrinimas ir funkcijų medžio sudarymas.
- Skaičiavimų paskirstymas.

Skaičiuojančiojo proceso funkcijos:

- Matematinių funkcijų apskaičiavimas .



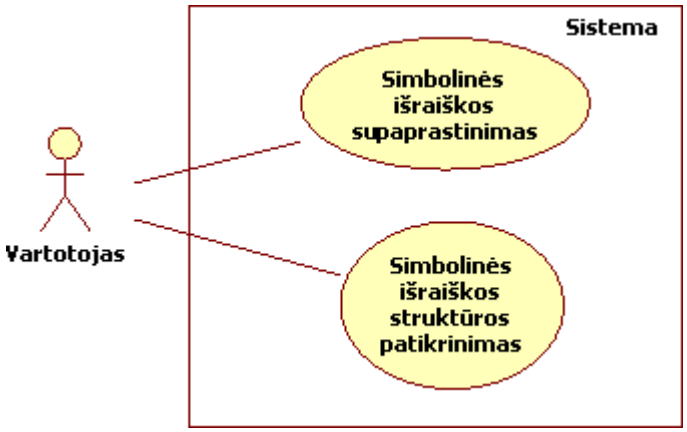
9 pav. Sistemos veikimo principas

3 Projektavimas

3.1 Apribojimai sprendimui

- Sistema turi veikti Linux operacinėje sistemoje;
- Skaičiavimai turi būti vykdomi lygiagrečiai;
- Sistemos kodas turi būti rašomas C++ programavimo kalba.

3.2 Panaudos atvejai



10 pav. Apibendrinti panaudos atvejai

1. PANAUDOJIMO ATVEJIS: Simbolinės išraiškos supaprastinimas

Vartotojas/Aktorius:	Vartotojas
Aprašas:	Supaprastinama simbolinė išraiška.
Prieš sąlyga:	Yra įvesta į programą simbolinė išraiška.
Sužadinimo sąlyga:	Vartotojas nuspaudė mygtuką “Supaprastinti išraišką”.
Po-sąlyga:	Programos lange pateiktas skaičiavimų rezultatas.

2. PANAUDOJIMO ATVEJIS: Simbolinės išraiškos struktūros patikrinimas

Vartotojas/Aktorius:	Vartotojas
Aprašas:	Patikrinama simbolinės išraiškos struktūra.
Prieš sąlyga:	Yra įvesta į programą simbolinė išraiška.
Sužadinimo sąlyga:	Vartotojas nuspaudė mygtuką “Patikrinti išraišką”.
Po-sąlyga:	Programos lange pateiktas patikrinimo rezultatas.

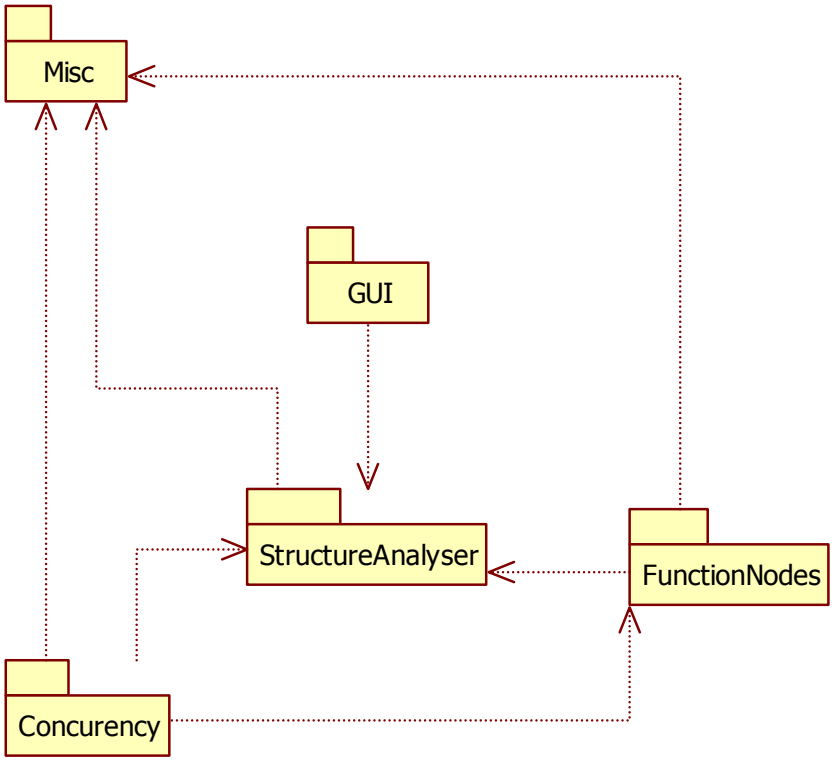
3.3 Funkciniai reikalavimai

Reikalavimas #:	1	Reikalavimo tipas:	10	Įvykis/panaudojimo atvejis #:	1
Aprašymas:	Programinė įranga turi teisingai supaprastinti matematinę išraišką				
Pagrindimas:	Vartotojui yra reikalinga tik teisingai supaprastinta išraiška.				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	Matematinė išraiška yra teisingai supaprastinta.				
Užsakovo tenkinimas:	5		Užsakovo netenkinimas:		5
Priklausomybės:	Nėra		Konfliktai:		Nėra

Reikalavimas #:	2	Reikalavimo tipas:	10	Įvykis/panaudojimo atvejis #:	2
Aprašymas:	Programinė įranga turi patikrinti matematinę išraiškos struktūrą				
Pagrindimas:	Vartotojui yra reikalinga tik strukūriškai taisyklingai parašyta išraiška.				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	Matematinė išraiškos struktūros patikrinimas pateikia teisingą įvertinimą.				
Užsakovo tenkinimas:	4		Užsakovo netenkinimas:		4
Priklausomybės:	Nėra		Konfliktai:		Nėra

3.4 Sistemos architektūra

3.4.1 Paketai



11 pav. Paketų diagrama

3.4.2 Paketas „StructureAnalyser“

Klasifikacija Paketas

Apibrėžimas Apjungia komponentus skirtus atlikti simbolinės funkcijos struktūros analizę.

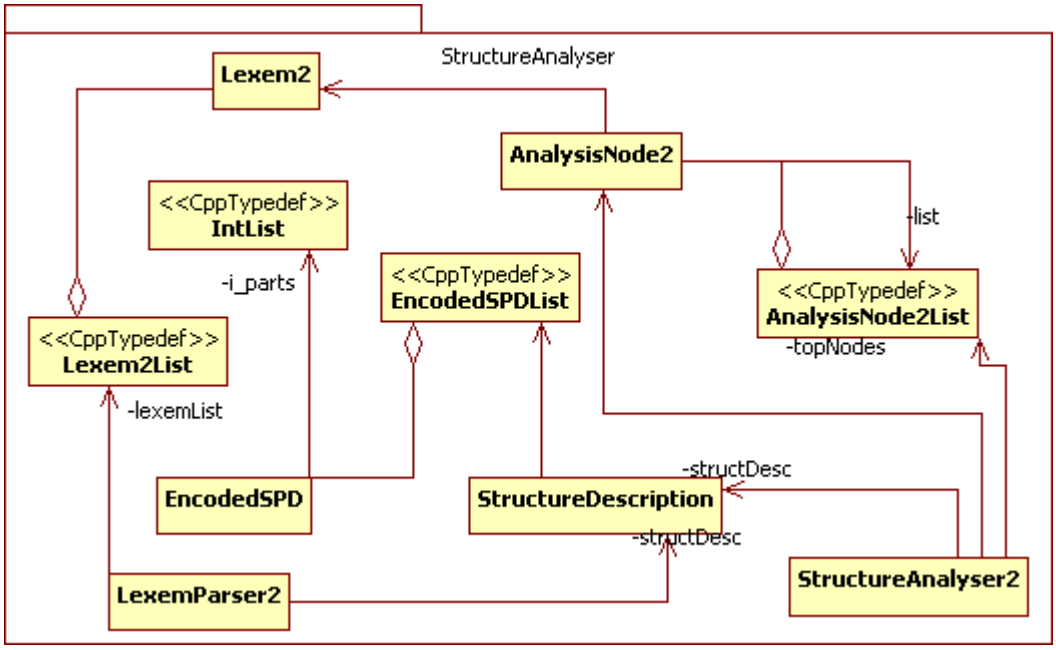
Atsakomybės Komponentas atsakingas už matematinės išraiškos įkėlimą, struktūros teisingumo patikrinimą, analizės medžio sudarymą.

Skaičiavimai Komponentas įkelia matematinę išraišką iš failo, atlieka jos leksinę ir sintaksinę analizę, sudaro analizės medį.

2 lentelė. Paketo „StructureAnalyser" klasių aprašas

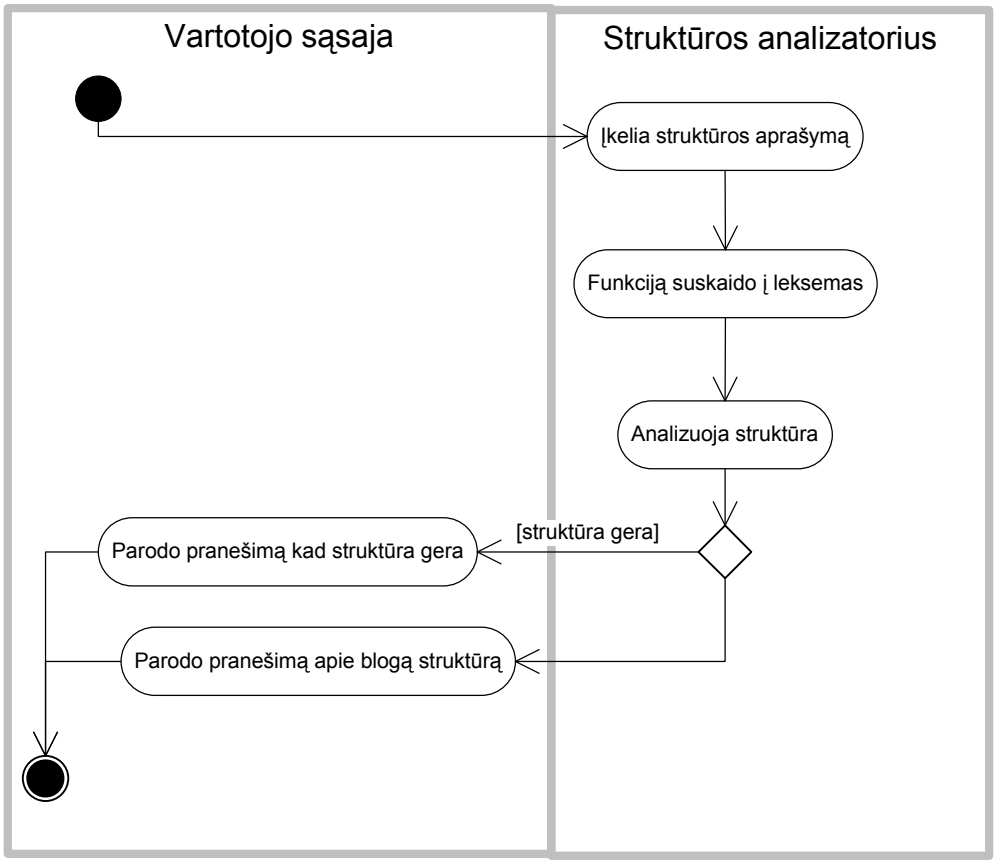
Klasė	Aprašas
Lexem2	leksemos aprašas
LexemParser2	Iš pradinių duomenų sudaro leksemas
AnalysisNode2	Analizės medžio mazgas
EncodedSPD	Struktūros aprašo elementas
StructureAnalyser2	Iš leksemų suformuoja analizės medį
StructureDescription	Struktūros aprašo taisyklių rinkinys

Detalesnės klasių diagramos yra 8.2 skyrelyje „Detalios klasių diagramos“.

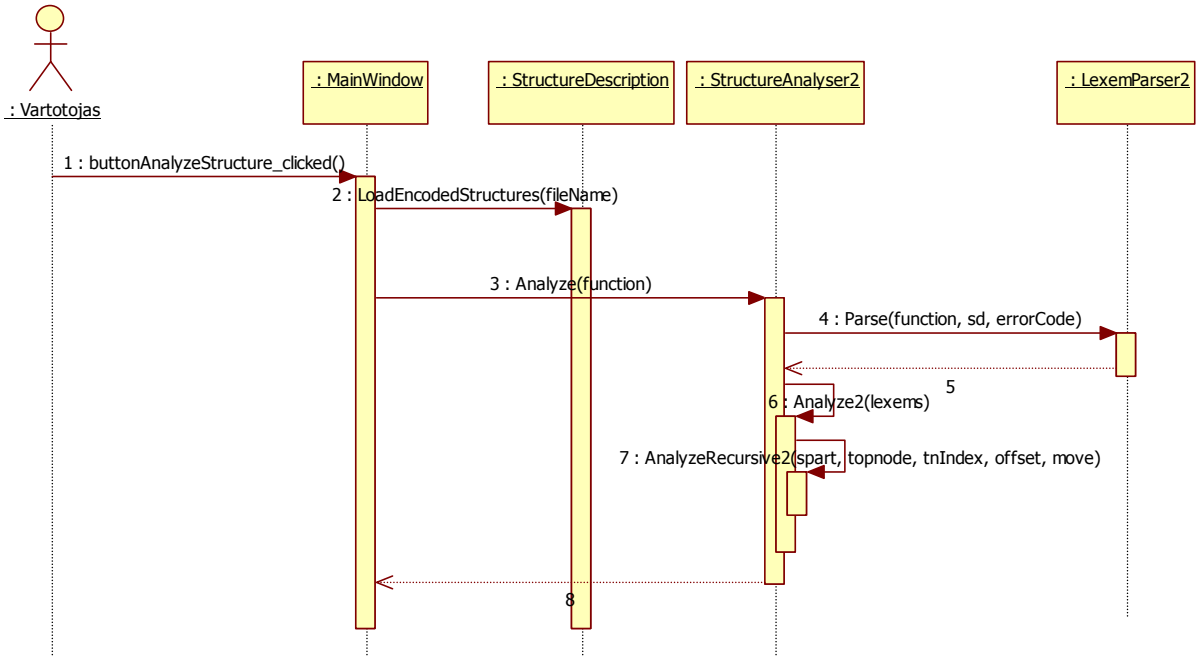


12 pav. Paketo „StructureAnalyser“ klasių diagrama

3.4.2.1 Struktūros patikrinimo diagramos



13 pav. Struktūros patikrinimo veiklos diagrama



14 pav. Struktūros patikrinimo sekos diagrama

3.4.3 Paketas „FunctionNodes“

Klasifikacija Paketas

Apibrėžimas Apjungia komponentus skirtus atlikti simbolinės funkcijų skaičiavimus.

Atsakomybės Komponentas atsakingas už matematinių operacijų medžio sudarymą, matematinių operacijų apskaičiavimą.

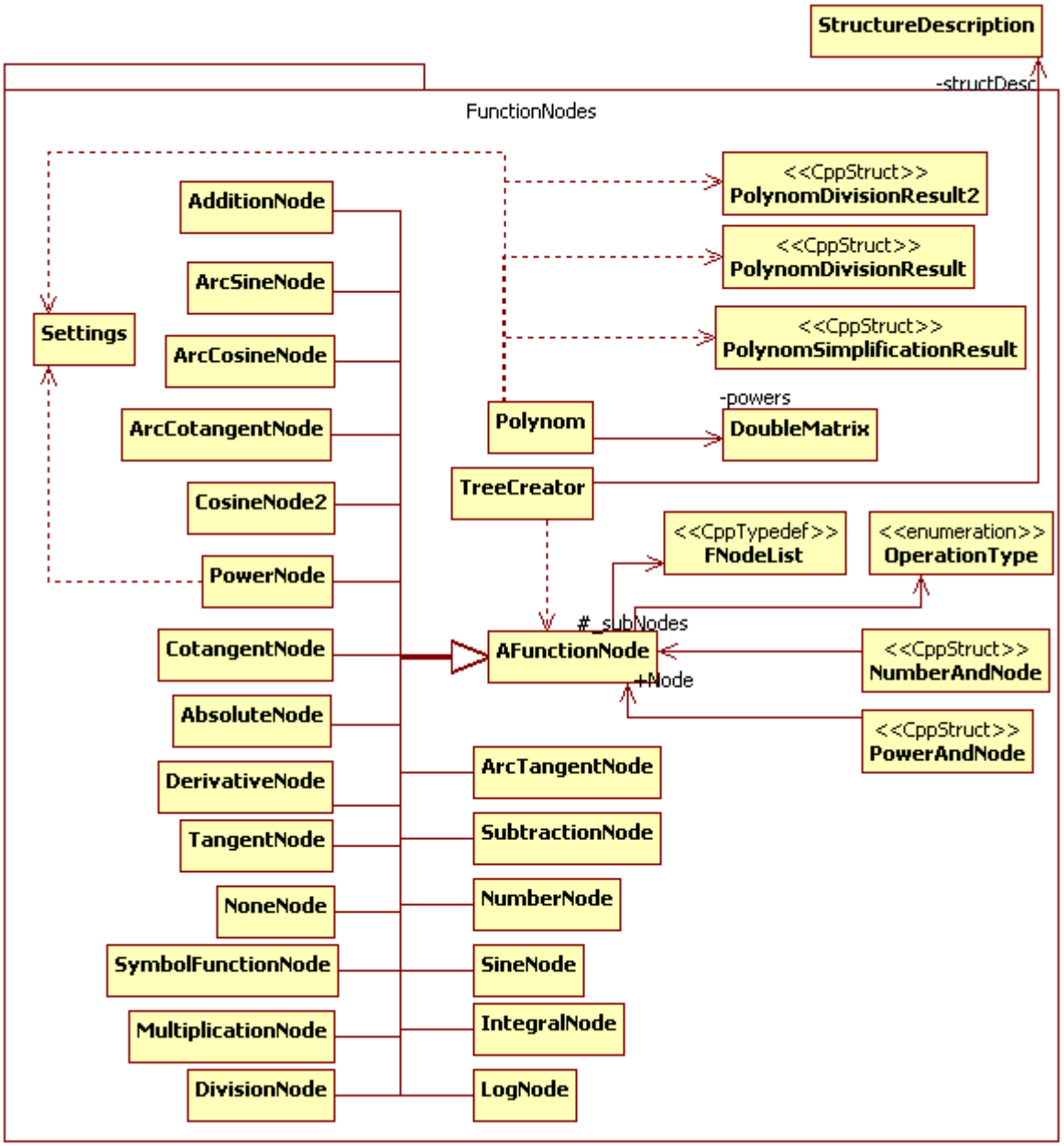
Skaičiavimai Komponentas iš analizės medžio sudaro matematinių operacijų medį. Apskaičiuoja matematinės operacijos rezultatą.

3 lentelė. Paketo „FunctionNodes" klasių aprašas

Klasė	Aprašas
AFunctionNode	Bazinė matematinių operacijų klasė
AbsoluteNode	Modulio matematinė operacija
AdditionNode	Sudėties matematinė operacija
ArcCosineNode	Arkkosinuso matematinė operacija
ArcCotangentNode	Arkkotangento matematinė operacija
ArcSineNode	Arksinuso matematinė operacija
ArcTangentNode	Arktangento matematinė operacija
CosineNode2	Kosinuso matematinė operacija
CotangentNode	Kotangento matematinė operacija
DerivativeNode	Išvestinės matematinė operacija
DivisionNode	Dalybos matematinė operacija
DoubleMatrix	Kintamo dydžio matrica su slankaus kablelio reikšmėmis
IntegralNode	Integralo matematinė operacija
LogNode	Logoritmo matematinė operacija
MultiplicationNode	Daugybos matematinė operacija
NoneNode	Šakninė operacija
NumberNode	Skaitinio kintamojo matematinė operacija
Polynom	Polinomo aprašas
PowerNode	Kėlimo laipsniu matematinė operacija
Settings	Skaičiavimo nustatymai
SineNode	Sinuso matematinė operacija
SubtractionNode	Atimties matematinė operacija

SymbolFunctionNode	Simbolinio kintamojo matematinė operacija
TangentNode	Tangento matematinė operacija
TreeCreator	Suformuoja funkcijų medį

Detalesnės klasių diagramos yra 8.2 skyrelyje „Detalios klasių diagramos“.



15 pav. Paketo „FunctionNodes“ klasių diagrama

3.4.4 Paketas „GUI “

Klasifikacija Paketas

Apibrėžimas Apjungia vartotojo sąsajos komponentus.

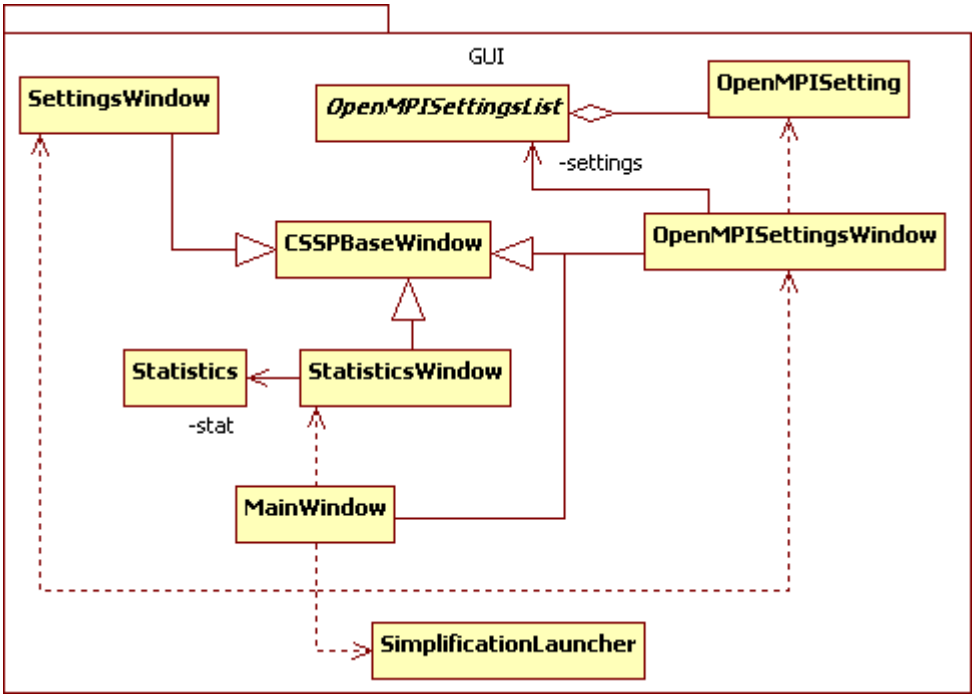
Atsakomybės Komponentas atsakingas už bendravimą su vartotoju.

Skaičiavimai Komponentas parodo ar priima duomenis iš vartotojo.

4 lentelė. Paketo „GUI" klasių aprašas

Klasė	Aprašas
CSSPBaseWindow	Bazinis programos langas
OpenMPISettingsList	Kompiuterių nustatymų sąrašas
OpenMPISetting	Vieno kompiuterio nustatymai
OpenMPISettingsWindow	OpenMPI nustatymų langas
SettingsWindow	Skaičiavimų nustatymų langas
StatisticsWindow	Statistikos peržiūros langas
Statistics	Statistikos sąrašas
SimplificationLauncher	Matematinių išraiškų skaičiavimo paleidimo klasė
MainWindow	Pagrindis programos langas

Detalesnės klasių diagramos yra 8.2 skyrelyje „Detalios klasių diagramos“.



16 pav. Paketo „GUI “ klasių diagrama

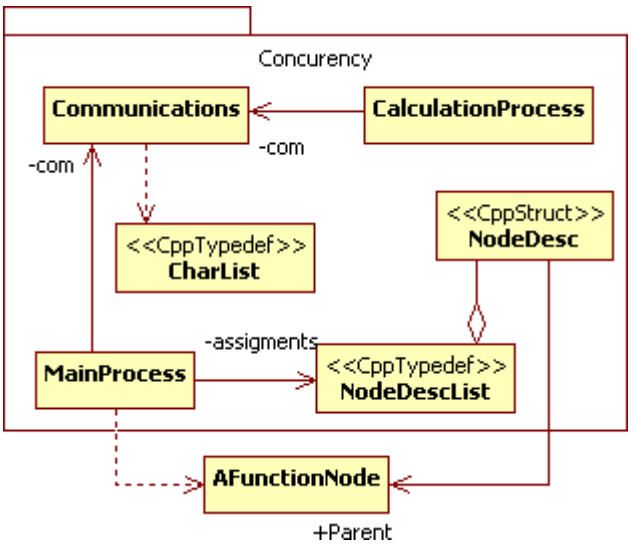
3.4.5 Paketas „ Concurency“

Klasifikacija	Paketas
Apibrėžimas	Apjungia lygiagrečių skaičiavimų procesų komponentus.
Atsakomybės	Atsakingas už skaičiavimų paskirstymą, duomenų perdavimą tarp procesų.
Skaičiavimai	Paskirsto skaičiavimus procesams, atlieka duomenų perdavimą tarp procesų.

5 lentelė. Paketo „Concurency" klasių aprašas

Klasė	Aprašas
CalculationProcess	Skaičiavimo proceso klasė
Communications	Komunikacijų tarp procesų klasė
MainProcess	Pagrindinio proceso klasė

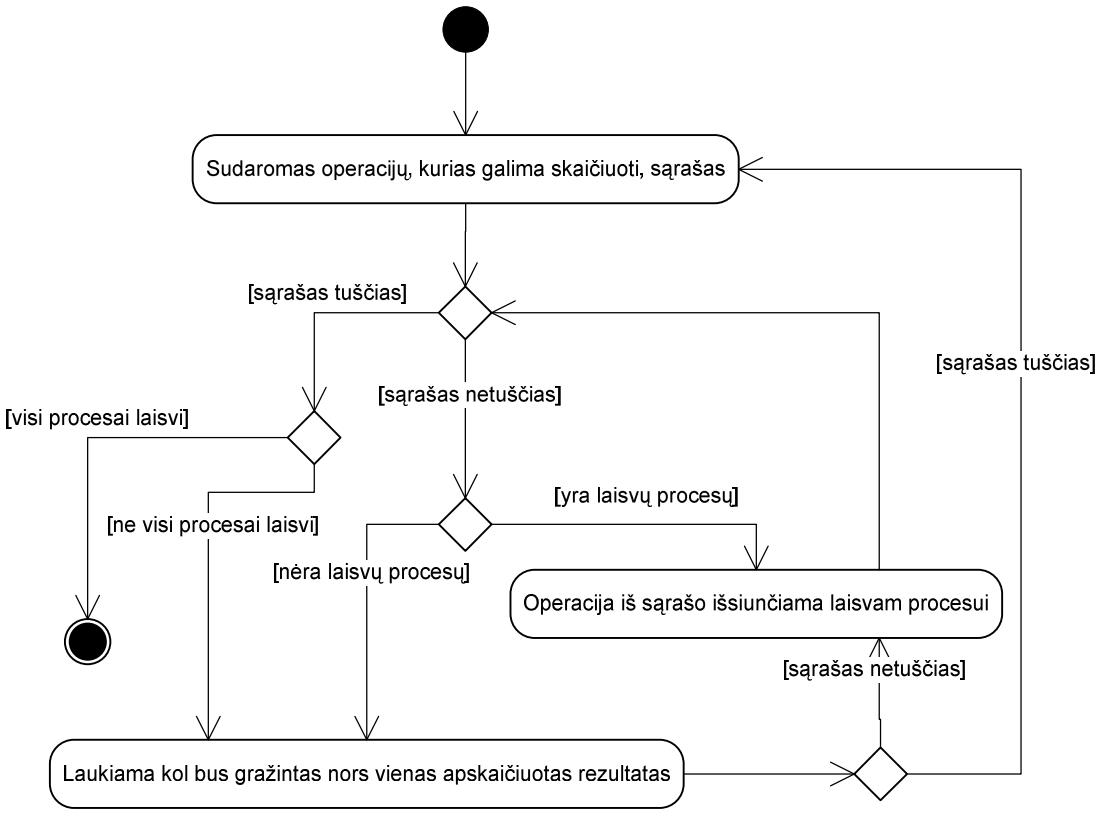
Detalesnės klasių diagramos yra 8.2 skyrelyje „Detalios klasių diagramos“.



17 pav. Paketo „Concurency “ klasių diagrama

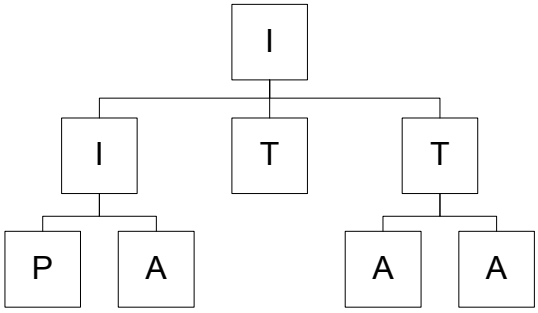
3.4.5.1 Skaičiavimų paskirstymo algoritmas

Skaičiavimų paskirstymo algoritmas yra pavaizduotas 18 paveikslėlyje.



18 pav. Skaičiavimų paskirstymo algoritmas

Skaičiavimams yra išrenkamos operacijos, kurių argumentai jau yra apskaičiuoti.

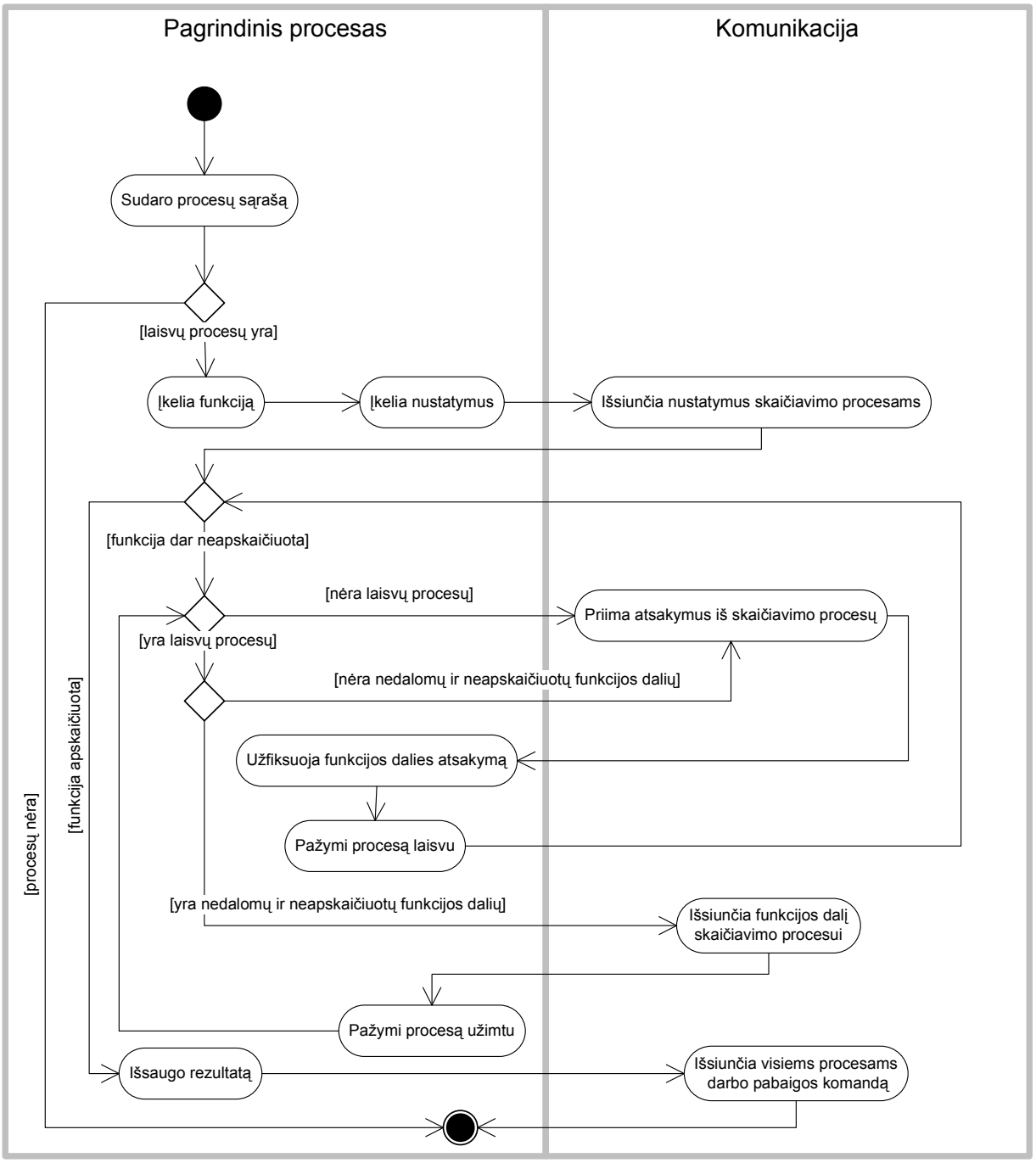


19 pav. Funkcijų išrinkimo pavyzdys

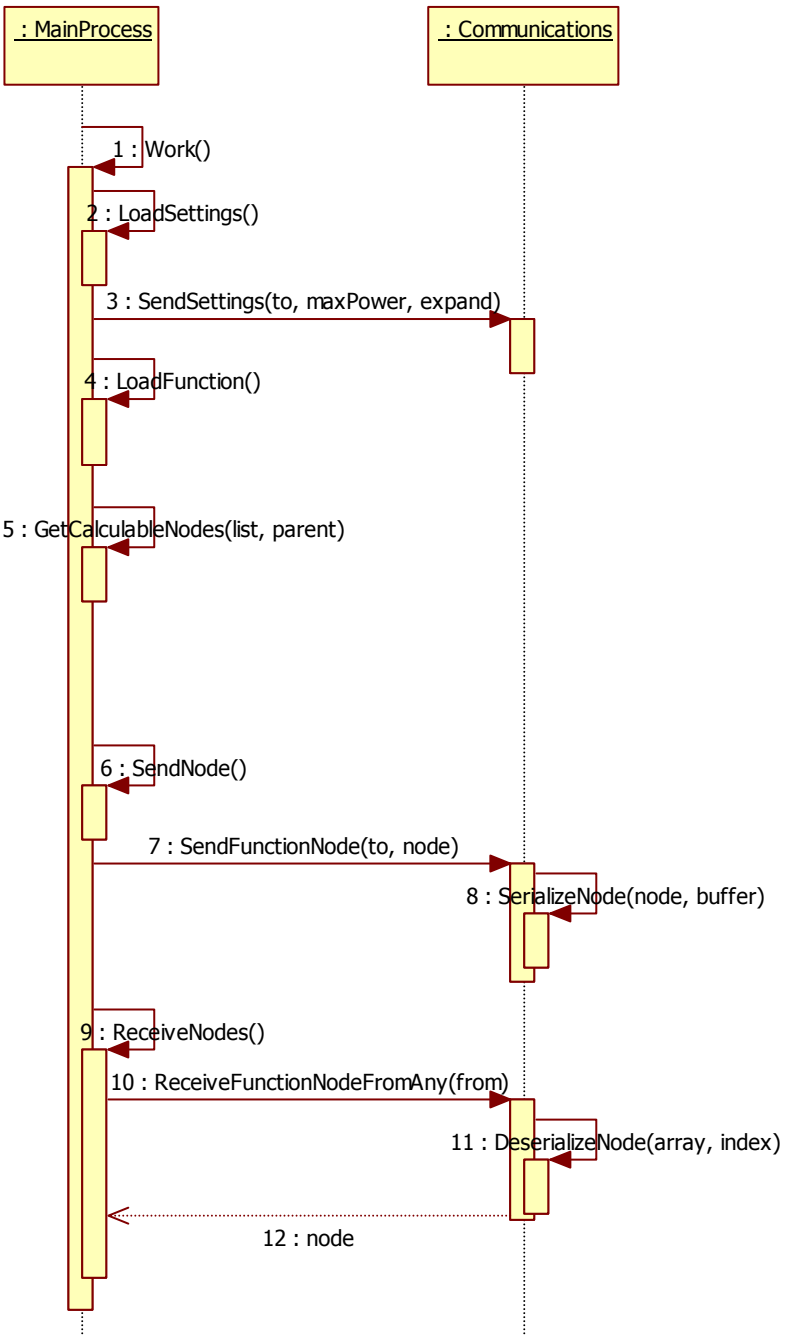
Žymėjimo paaiškinimai:

- **P**- paimta skaičiavimams funkcija.
- **A**- apskaičiuota funkcija.
- **I**- ignoruojama funkcija (tai tokia funkcija, kurios nors vienas argumentas yra paimtas skaičiavimams arba yra ignoruojamas)
- **T**- tinkama skaičiavimams funkcija.

3.4.5.2 Pagrindinio proceso diagramos

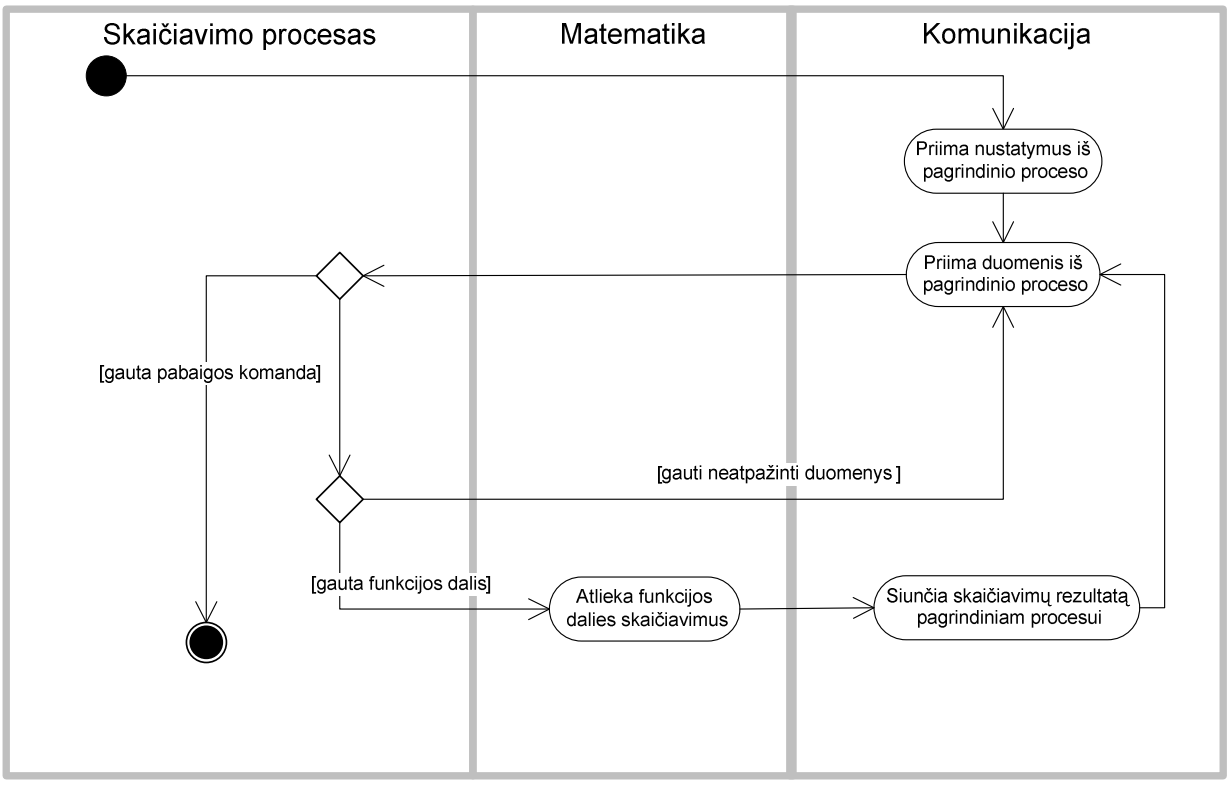


20 pav. Pagrindinio proceso veiklos diagrama

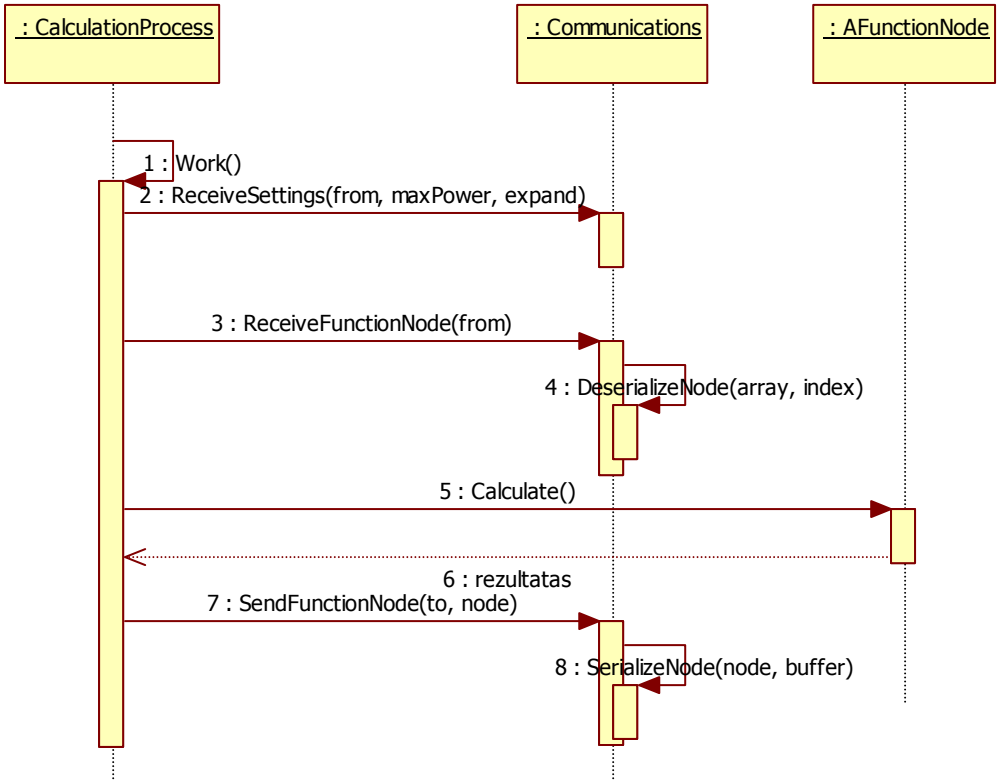


21 pav. Pagrindinio proceso sekų diagrama

3.4.5.3 Skaičiavimo proceso diagramos



22 pav. Skaičiavimo proceso veiklos diagrama



23 pav. Skaičiavimo proceso sekų diagrama

3.4.6 Paketas „Misc“

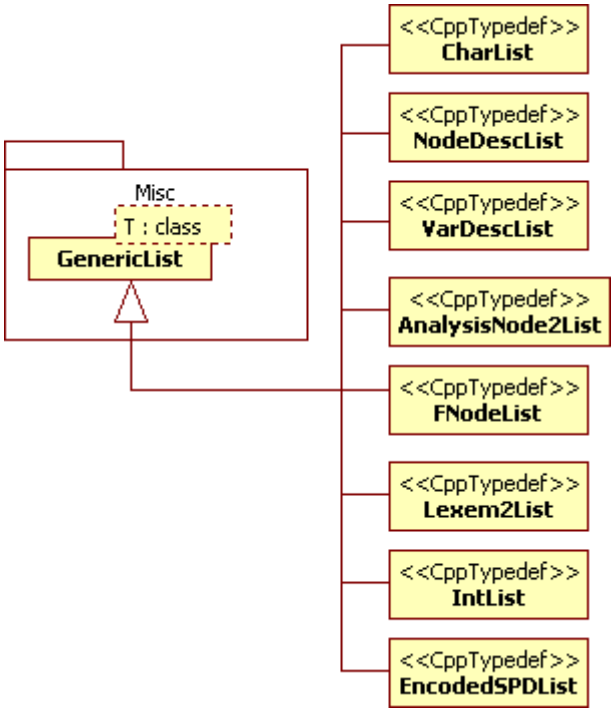
Klasifikacija Paketas

Apibrėžimas Apjungia pagalbinius komponentus.

6 lentelė. Paketo „Misc" klasių aprašas

Klasė	Aprašas
GenericList	Universalus sąrašas

Detalesnės klasių diagramos yra 8.2 skyrelyje „Detalios klasių diagramos“.



24 pav. Paketo „Misc“ klasių diagrama

4 Skaič iavimų algoritmo tyrimas

Šio tyrimo tikslas yra nustatyti matematinių išraiškų supaprastinimo programinės įrangos greičio priklausomybę nuo sistemos ir duomenų parametrų, tokių kaip matematinės išraiškos ilgis ir naudojamų procesorių kiekis.

Tiriamoji programinė įranga buvo papildyta atskirų programos dalių vykdymo laikų skaičiavimais. Be to buvo sukurtos, matematinių išraiškų įkėlimo ir paskirstymo algoritmų modifikacijos, kurios detaliau nagrinėjamos tolesniuose skyriuose.

Tyrimas suskirstytas į dvi dalis. Buvo tiriami matematinės išraiškos įkėlimo ir skaičiavimo greičiai.

4.1 Naudojama techninė įranga

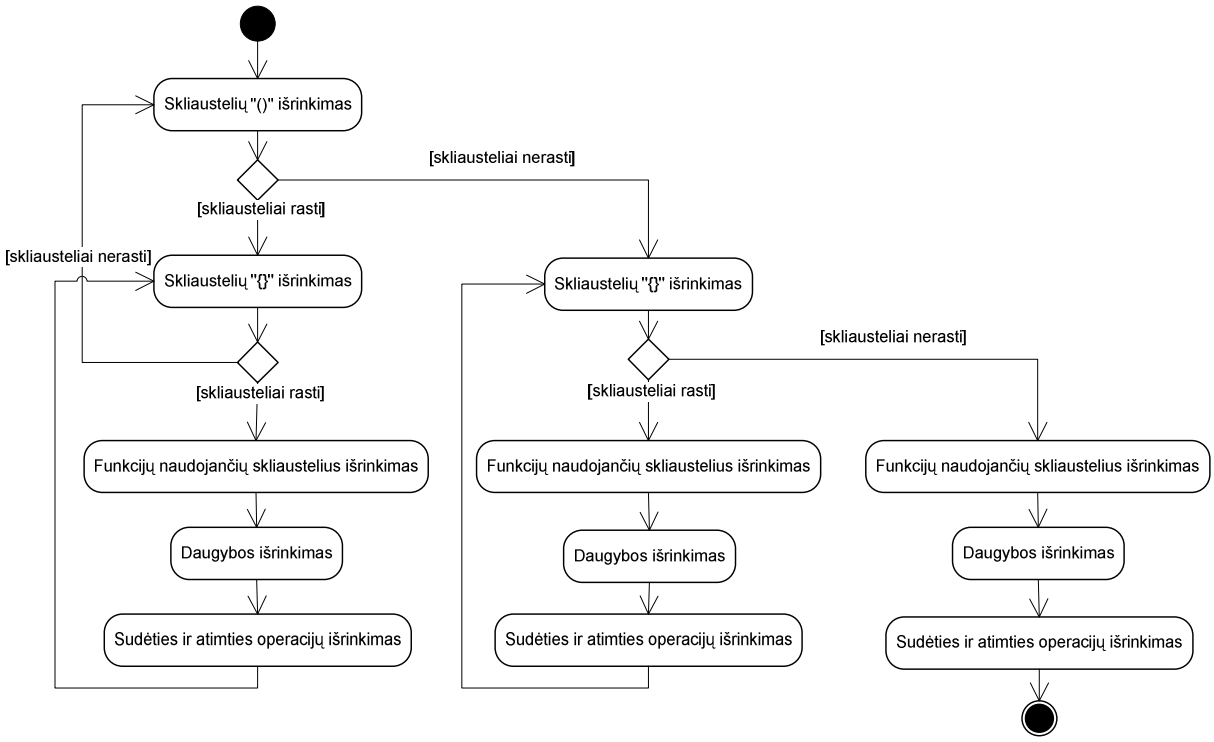
Tyrimams atlikti buvo naudojamas kompiuterių klasteris esantis Informacinių Technologijų Plėtros Instituto 320 auditorijoje.

7 lentelė. Techninės įrangos parametrai

Eil. Nr.	Kompiuteris	Operacinė sistema	Procesoriaus dažnis	Atminties kiekis
1	kopustas.elen.ktu.lt	Linux	2.992 MHz	1 GB
2	tulpe.elen.ktu.lt	Linux	2.992 MHz	1 GB
3	alyva.elen.ktu.lt	Linux	2.992 MHz	1 GB
4	zibute.elen.ktu.lt	Linux	2.992 MHz	1 GB
5	roze.elen.ktu.lt	Linux	2.992 MHz	1 GB
6	astra.elen.ktu.lt	Linux	2.992 MHz	1 GB

4.2 Matematinių išraiškų įkėlimas

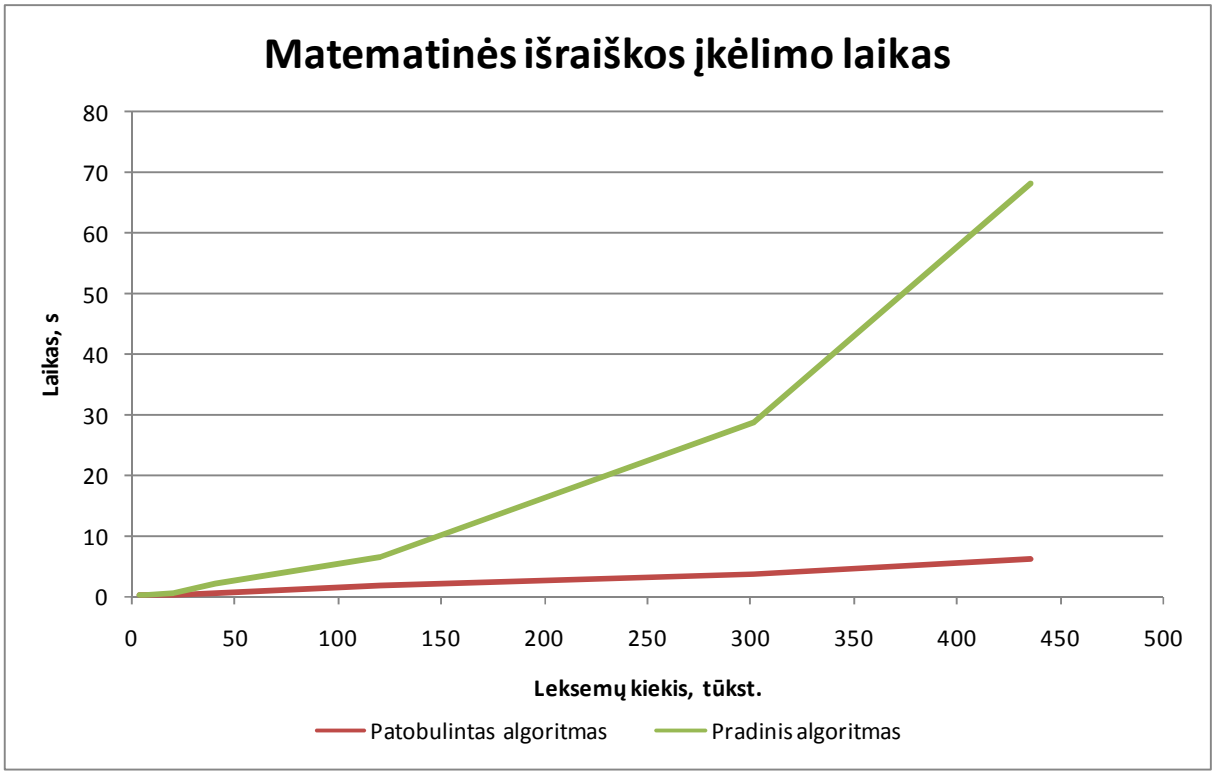
Pasirinktas matematinių išraiškų algoritmas yra universalus, norint pakeisti naudojamą gramatiką, programos kode reikia atlikti tik nedidelius pakeitimus, tačiau atliekant tyrimą buvo pastebėta, kad didelių išraiškų įkėlimas sudaro nemažą dalį visos programos veikimo laiko. Todėl buvo sukurtas kitas specializuotas ir greitesnis algoritmas.



25 pav. Patobulinto matematinės išraiškos įkėlimo algoritmo diagrama

8 lentelė. Matematinės išraiškos įkėlimo laikai

Leksemų kiekis	Laikas, ms	
	Pradinis algoritmas	Patobulintas algoritmas
3383	289	150
5813	294	184
20353	749	303
39735	2304	525
120003	6444	1865
301081	28713	3760
435143	68349	6284



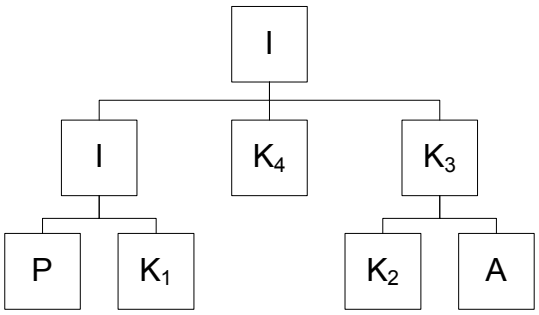
26 pav. Matematinės išraiškos įkėlimo laikai

4.3 Skaičiavimai

Atliekant skaičiavimų greičio tyrimus buvo sukurti 2 papildomi funkcijų išrinkimo algoritmai.

1. Išrinkimas pagal kainą

Funkcijos kaina- tai statistinis funkcijos skaičiavimo trukmės įvertis. Funkcija n yra paimama skaičiavimams, kai jos kaina K_n yra lygi arba viršija nustatyta ribą K, arba jos tėvinė funkcija yra ignoruojama (tai funkcija, kurios argumentas yra paimtas skaičiavimams arba pažymėtas kaip ignoruojamas). Ignoruojamų, paimtų skaičiavimams ar apskaičiuotų funkcijų kainos nėra skaičiuojamos.



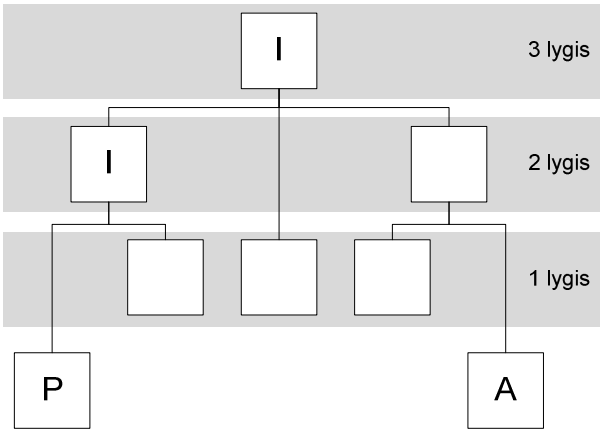
27 pav. Funkcijos išrinkimo pagal kainą pavyzdys

Žymėjimo paaiškinimai:

- **P**- paimta skaičiavimams funkcija.
- **A**- apskaičiuota funkcija.
- **I**- ignoruojama funkcija.
- **K_n**- funkcijos n kaina.

2. Išrinkimas pagal funkcijos lygį

Funkcija yra paimama skaičiavimams, kai jos lygis yra lygus arba viršija nustatytą ribą L, arba jos tėvinė funkcija yra ignoruojama. Ignoruojamų, paimtų skaičiavimams ar apskaičiuotų funkcijų lygiai nėra skaičiuojami.



28 pav. Funkcijos išrinkimo pagal lygį pavyzdys

Buvo suformuoti 5 išrinkimo metodai:

c0- pradinis išrinkimo metodas.

c1- išrinkimas pagal kainą, kai kainos riba K yra pastovaus dydžio.

c2- išrinkimas pagal kainą, kai kainos riba K kinta priklausomai nuo dar neapskaičiuotos funkcijos dalies kainos ir skaičiavimo procesų skaičiaus.

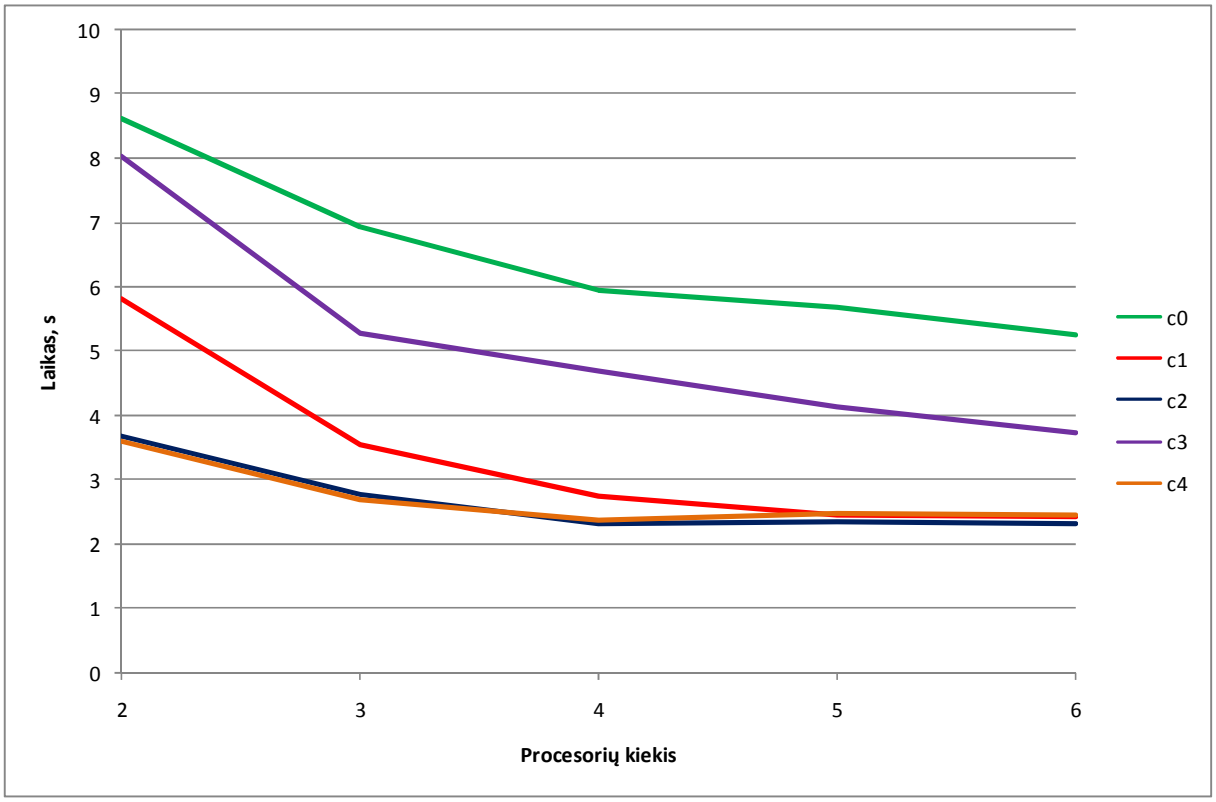
c3- išrinkimas pagal funkcijos lygį, kai lygio riba L yra pastovaus dydžio.

c4- išrinkimas pagal funkcijos lygį, kai lygio riba L kinta priklausomai nuo dar neapskaičiuotos funkcijos dalies lygių skaičiaus ir skaičiavimo procesų skaičiaus.

4.3.1 Skaičiavimas su 50903 leksemomis

9 lentelė. Skaičiavimo su 50903 leksemomis laikai

Leksemų kiekis: 50903	Procesorių kiekis				
Funkcijų išrinkimo metodas	2	3	4	5	6
c0	9 s	7 s	6 s	6 s	5 s
c1	6 s	4 s	3 s	2 s	2 s
c2	4 s	3 s	2 s	2 s	2 s
c3	8 s	5 s	5 s	4 s	4 s
c4	4 s	3 s	2 s	2 s	2 s
Geriausi metodai	c2/c4	c2/c4	c2/c4	c1/c2/c4	c1/c2/c4



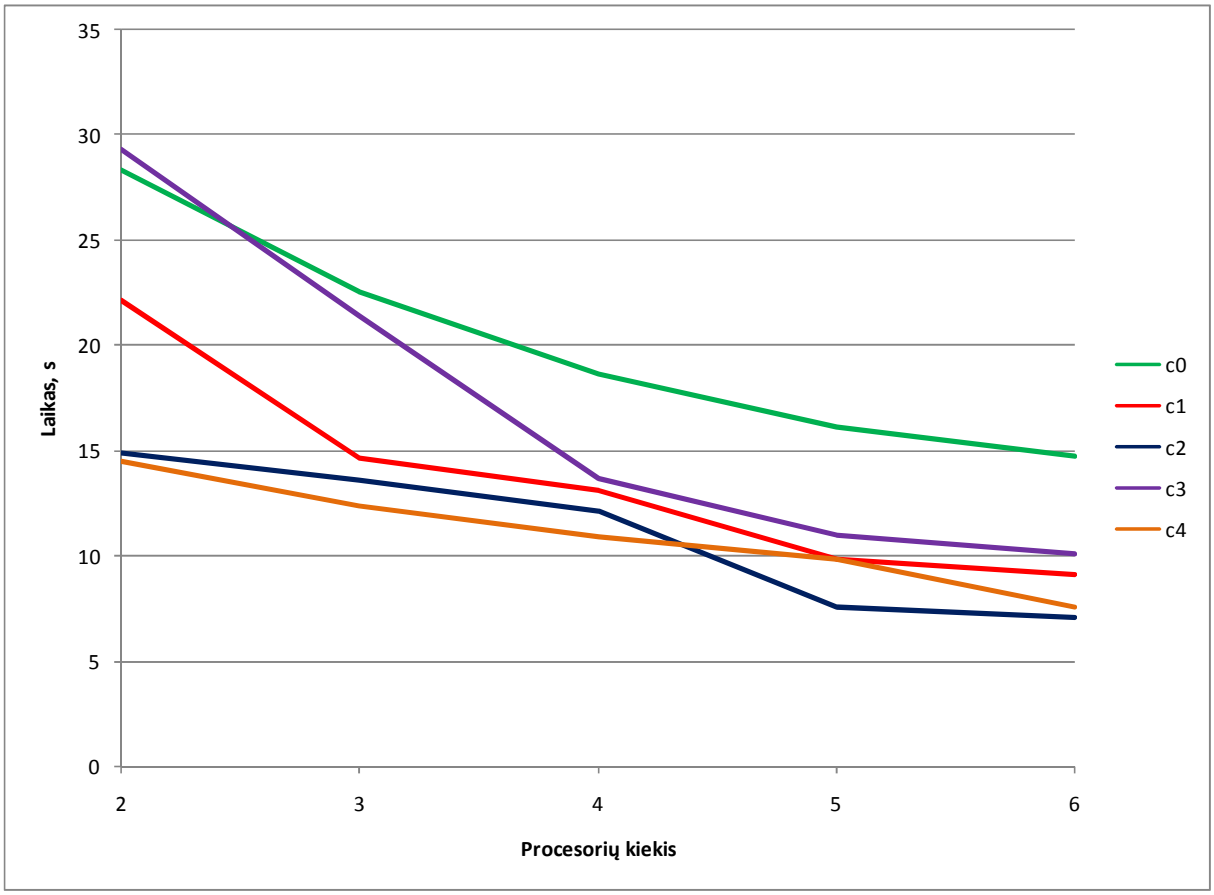
29 pav. Skaičiavimo su 50903 leksemomis laikai

Pastaba: vienas iš procesorių yra naudojamas valdančiojo proceso.

4.3.2 Skaičiavimas su 137859 leksemomis

10 lentelė. Skaičiavimo su 137859 leksemomis laikai

Leksemų kiekis: 137859	Procesorių kiekis				
Funkcijų išrinkimo metodas	2	3	4	5	6
c0	28 s	23 s	19 s	16 s	15 s
c1	22 s	15 s	13 s	10 s	9 s
c2	15 s	14 s	12 s	8 s	7 s
c3	29 s	21 s	14 s	11 s	10 s
c4	15 s	12 s	11 s	10 s	8 s
Geriausi metodai	c2/c4	c4	c4	c2	c2



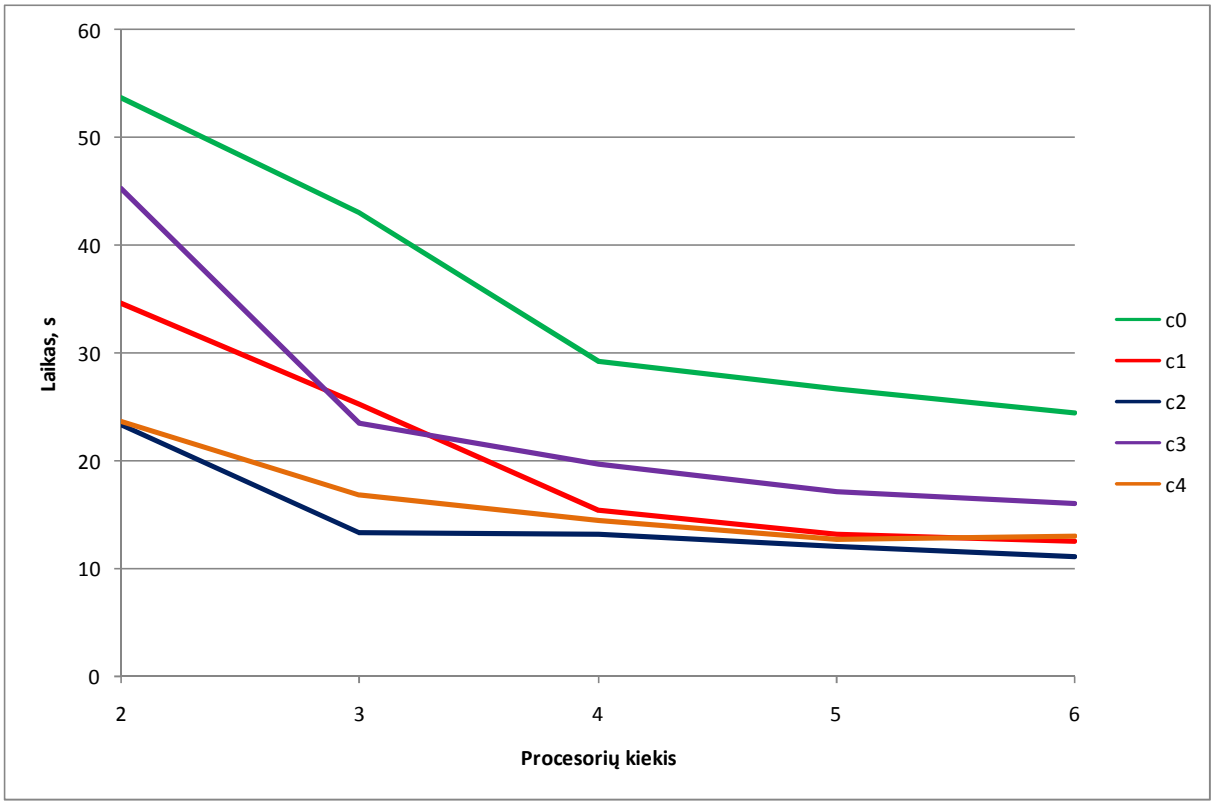
30 pav. Skaičiavimo su 137859 leksemomis laikai

Pastaba: vienas iš procesorių yra naudojamas valdančiojo proceso.

4.3.3 Skaičiavimas su 198861 leksemomis

11 lentelė. Skaičiavimo su 198861 leksemomis laikai

Leksemų kiekis: 198861	Procesorių kiekis				
Funkcijų išrinkimo metodas	2	3	4	5	6
c0	54 s	43 s	29 s	27 s	24 s
c1	35 s	25 s	15 s	13 s	13 s
c2	23 s	13 s	13 s	12 s	11 s
c3	45 s	24 s	20 s	17 s	16 s
c4	24 s	17 s	15 s	13 s	13 s
Geriausi metodai	c2	c2	c2	c2	c2



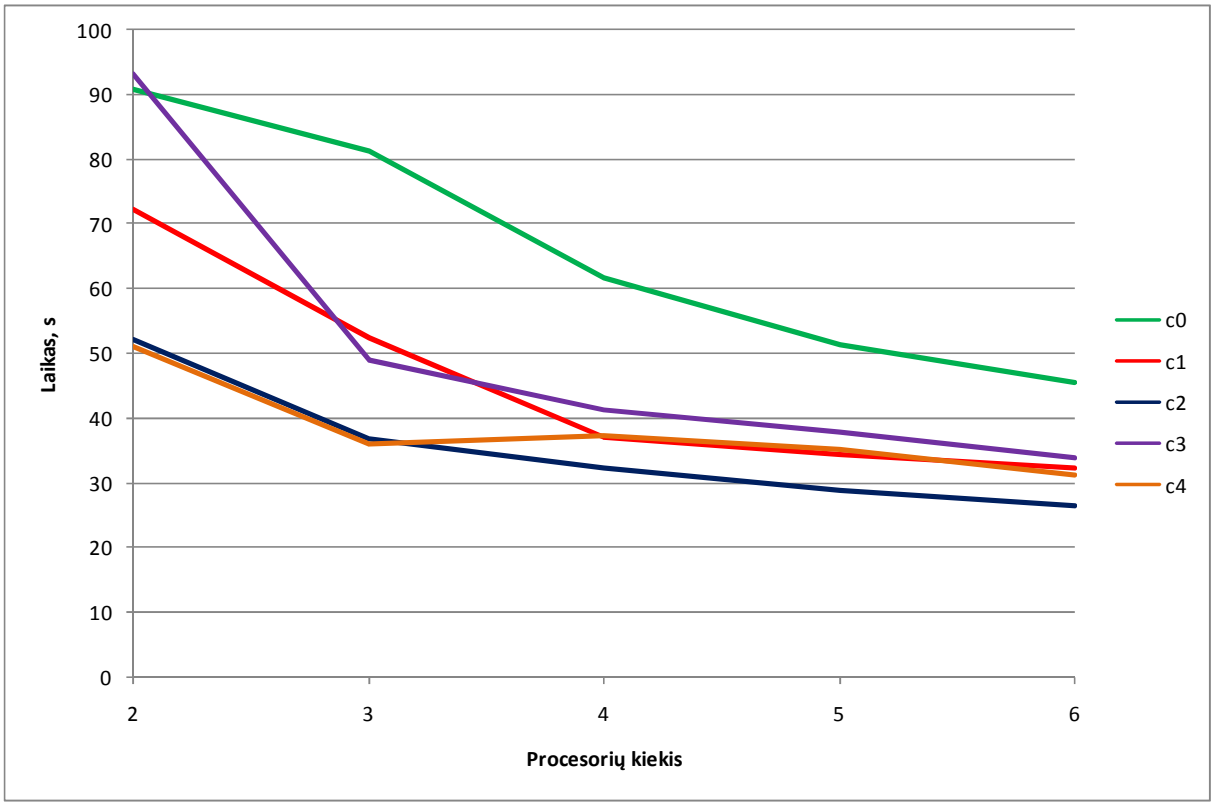
31 pav. Skaičiavimo su 198861 leksemomis laikai

Pastaba: vienas iš procesorių yra naudojamas valdančiojo proceso.

4.3.4 Skaičiavimas su 381971 leksemomis

12 lentelė. Skaičiavimo su 381971 leksemomis laikai

Leksemų kiekis: 381971	Procesorių kiekis				
Funkcijų išrinkimo metodas	2	3	4	5	6
c0	91 s	82 s	62 s	51 s	46 s
c1	72 s	52 s	37 s	34 s	32 s
c2	52 s	37 s	32 s	29 s	26 s
c3	93 s	49 s	41 s	38 s	34 s
c4	51 s	36 s	37 s	35 s	31 s
Geriausi metodai	c4	c2	c2	c2	c2



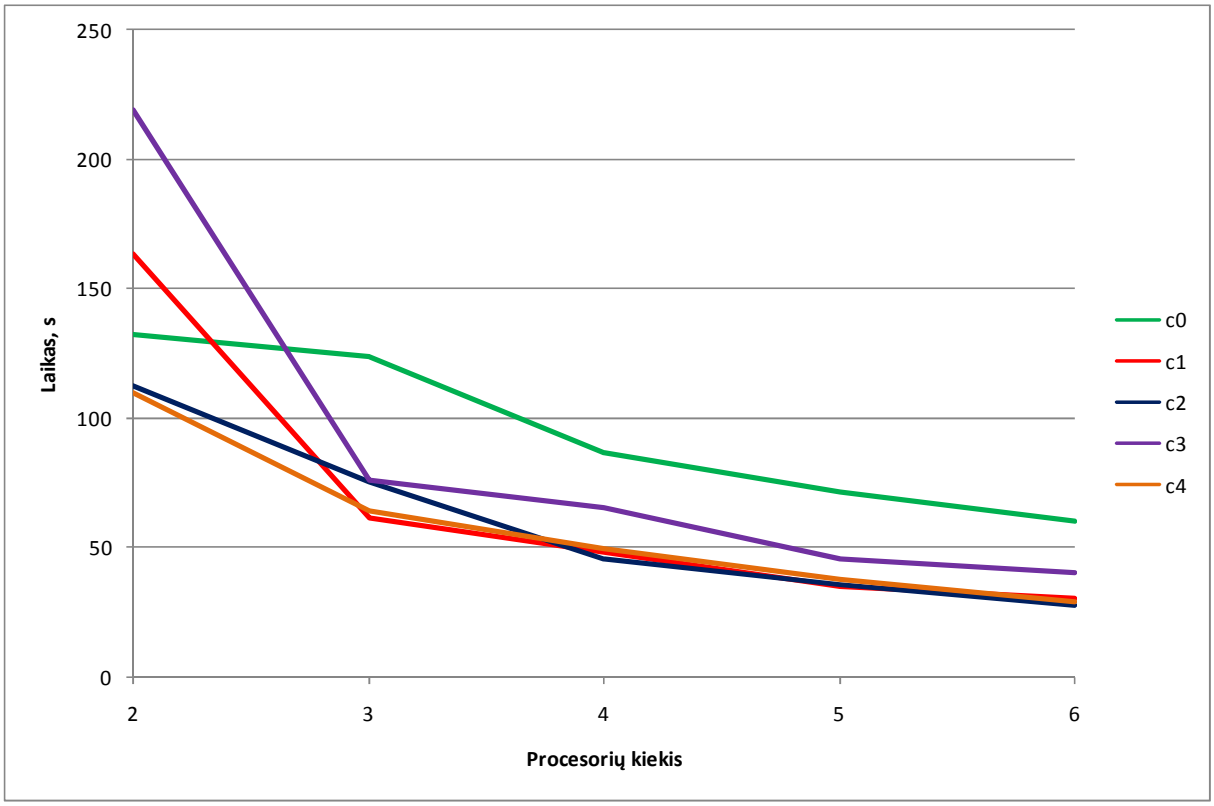
32 pav. Skaičiavimo su 381971 leksemomis laikai

Pastaba: vienas iš procesorių yra naudojamas valdančiojo proceso.

4.3.5 Skaičiavimas su 518035 leksemomis

13 lentelė. Skaičiavimo su 518035 leksemomis laikai

Leksemų kiekis: 518035	Procesorių kiekis				
Funkcijų išrinkimo metodas	2	3	4	5	6
c0	132 s	124 s	87 s	72 s	60 s
c1	163 s	61 s	48 s	35 s	31 s
c2	113 s	62 s	45 s	35 s	28 s
c3	219 s	76 s	66 s	46 s	40 s
c4	110 s	64 s	49 s	37 s	29 s
Geriausi metodai	c4	c2	c2	c2	c4



33 pav. Skaičiavimo su 518035 leksemomis laikai

Pastaba: vienas iš procesorių yra naudojamas valdančiojo proceso.

4.4 Eksperimentų rezultatai

Kaip matyti iš 14 lentelės geriausi funkcijų išrinkimo metodas yra c2.

Eksperimentai parodė, kad:

- c0 metodas yra pats prasčiausias iš visų.
- Išrinkimo pagal kainą metodas yra geresnis nei išrinkimo pagal lygį.
- Kai yra naudojamas tik 1 skaičiuojantis procesas efektyviausias yra c4 metodas.

14 lentelė. Geriausi išrinkimo metodai pagal procesorių ir leksemų kiekį

Leksemų kiekis	Procesorių kiekis				
	2	3	4	5	6
50903	c2/c4	c2/c4	c2/c4	c1/c2/c4	c1/c2/c4
137589	c2/c4	c4	c4	c2	c2
198681	c2	c2	c2	c2	c2
381971	c4	c2	c2	c2	c2
518035	c4	c2	c2	c2	c2

15 lentelė. Leksemų kieki lentelėje yra pavaizduoti leksemų kiekių sumažėjimas po supaprastinimo.

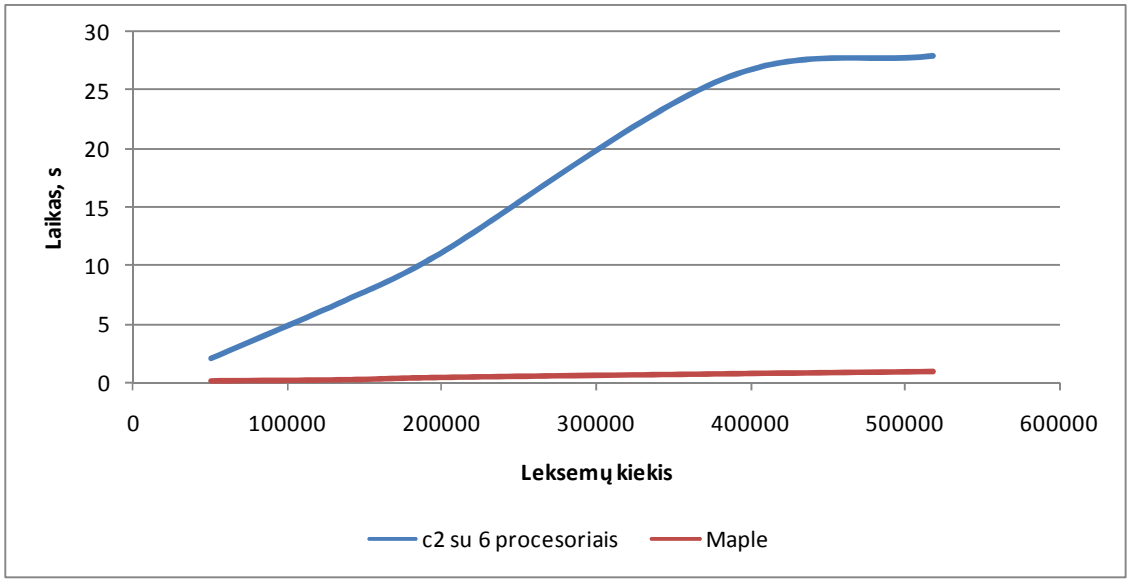
15 lentelė. Leksemų kiekio sumažėjimas po supaprastinimo

Pradinis leksemų kiekis	Leksemų kiekis po supaprastinimo	Sumažėjimas
50903	25358	50,2%
137859	13342	90,3%
198861	70680	64,5%
381971	21457	94,4%
518035	100438	80,6%

Tolimesnis duomenų paskirstymo algoritmo gerinimas galėtų būti atliekamas tobulinant kainų algoritmą, parenkant tikslesnius operacijų trukmės įvertinimus. Eksperimentai buvo atliekami naudojant kompiuterius su vienodais techniniais parametrais, tačiau realioje situacijoje tokia skaičiavimo aplinka yra mažai tikėtina, todėl reikėtų į paskirstymo algoritmą įtraukti ir naudojamos techninės įrangos įvertinimą.

Kaip galima matyti iš 34 pav. Maple skaičiavimo laikas yra žymiai mažesnis, tai rodo kad tiriamoji programinė įranga yra neefektyvi laiko atžvilgiu. Lėčiausioji vieta yra skaičiavimai. Tai galima spręsti iš skaičiavimų laiko, kai yra naudojamas c4 metodas su 2

procesoriais, nes tuo atveju skaičiavimui yra perduodama iš karto visa matematinė išraiška. Sistemos tobulinimas turėtų būti vykdomas optimizuojant skaičiavimo algoritmus. Gali reikėti peržiūrėti ir pačią duomenų struktūrą.



34 pav. Sukurtos PĮ ir Maple skaičiavimo laikai

Sistemą galima pagerinti ir papildant ją naujomis matematinėmis funkcijomis ar jų užrašymo sintakse. Detalesnį funkcijų papildymo aprašą galima paskaityti skyrelyje „8.3 Programos papildymas naujomis matematinėmis funkcijomis“.

5 Išvados

- Atlikus tyrimus buvo nustatyta, kad efektyviausias yra algoritmas, kuriame skaičiavimai paskirstomi pagal matematinių operacijų kainą.
- Nustatyta, kad programinė įranga dirba neefektyviai. Lėčiausia jos vieta yra skaičiavimo algoritmai.
- Tolesnis darbo vystymas turėtų būti vykdomas optimizuojant skaičiavimo algoritmus.

6 Literatūra

1. Petcu D., et al. *Symbolic Computations on Grids*. [Žiūrėta 2007-10-19]. Prieiga per internetą <http://mpaprzycki.swps.edu.pl/mp/cvr/research/varia_papers/Grid_Chapter_2006.pdf>.
2. *HPC-Grid*. [Žiūrėta 2007-10-26]. Prieiga per internetą <http://www.maplesoft.com/products/thirdparty/HPCGrid/>
3. *Parallel Computing Toolkit*. [Žiūrėta 2007-10-25]. Prieiga per internetą <http://www.wolfram.com/products/applications/parallel/>
4. *MPI: A Message-Passing Interface Standard.* [Žiūrėta 2007-11-04]. Prieiga per internetą <http://www.mpi-forum.org/docs/mpi1-report.pdf>
5. *Open MPI*. [Žiūrėta 2007-11-10]. Prieiga per internetą <http://www.open-mpi.org/>.
6. *Augmented BNF for Syntax Specifications: ABNF*. [Žiūrėta 2009-01-10]. Prieiga per internetą <ftp://ftp.rfc-editor.org/in-notes/rfc4234.txt>
7. Grune D., Jacobs C.. *Parsing Techniques - A Practical Guide*. England, 1990. ISBN 0136-514-31-6
8. Vaškevičienė A., Vaškevičius V. *Matematinių išraiškų pertvarkymo programinė įranga*. Magistro darbas. KTU Informatikos fakultetas, 2004.
9. Steinhaus S.. *Comparison of mathematical programs for data analysis*. [Žiūrėta 2009-04-11]. Prieiga per internetą <http://www.scientificweb.com/ncrunch/ncrunch5.pdf>
10. *MPICH2*. [Žiūrėta 2007-11-10]. Prieiga per internetą <http://www-unix.mcs.anl.gov/mpi/mpich2/>
11. *LaTeX – A document preparation system*. [Žiūrėta 2008-09-10]. Prieiga per internetą < http://www.latex-project.org//>

1.

7 Terminų ir santrumpų žodynas

MPI	Žinučių perdavimo sąsaja (angl. Message Passing Interface)
Leksema	Mažiausias kalbos žodyno vienetas
Ad-hoc	Bevielių įrenginių tinklas bendraujantis tiesiogiai nenaudodami prieigos taškų
Scenarijus	Komandų seka skirta automatizuoti tam tikram darbui
Terminaliniai simboliai	Simboliai, iš kurių formuojamos eilutės sugeneruotos pagal pasirinktą gramatiką
Neterminaliniai simboliai	Simboliai, kurie gali būti pakeisti kitais neterminaliniais ar terminaliniais simboliais
Simbolinė išraiška	Matematinė simbolinė funkcija.
Kompiuterių klasteris	Spartaus ryšio kanalais sujungtų kompiuterių grupė, kuri vartotojo požiūriu veikia kaip vieningas įrenginys.
OpenMPI	Viena iš MPI standarto realizacijų.
ABNF	Išplėsta Backus-Naur forma (angl. Augmented Backus–Naur Form)

8 Priedai

8.1 Gramatika

Gramatika, kuri bus naudojama sistemoje, užrašyta ABNF forma:

```
<DIGIT> ::= "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
<NOSIGN_NUM> ::= <DIGIT> | <DIGIT> <NOSIGN_NUM>
<SIGN_NUM> ::= "-" <NOSIGN_NUM>
<LETTER> ::= "q"|"w"|"e"|"r"|"t"|"y"|"u"|"i"|"o"|"p"|"a"|"s"
|"d"|"f"|"g"|"h"|"j"|"k"|"l"|"z"|"x"|"c"|"v"|"b"|"n"|"m"|"Q"|"
W"|"E"|"R"|"T"|"Y"|"U"|"I"|"O"|"P"|"A"|"S"|"D"|"F"|"G"|"H"|"J"
|"K"|"L"|"Z"|"X"|"C"|"V"|"B"|"N"|"M"
<STRING> ::= <LETTER> (<LETTER> | <LETTER> <STRING>)
[NOSIGN_NUM]
<PLUS> ::= "+"
<MINUS> ::= "-"
<LEFT_BRACKET> ::= "("
<RIGHT_BRACKET> ::= ")"
<CURLY_LEFT> ::= "{"
<CURLY_RIGHT> ::= "}"
<DECIMAL_POINT> ::= "."
<DOLLAR> ::= "$"
<INTD> ::= "\,d"
<POWER> ::= "^"
<UNDERSCORE> ::= "_"
<ABS_BRACKET> ::= "|"
<LOG> ::="\log"
<PARTIAL> ::= "\partial"
<SIN> ::= "\sin"
<COS> ::= "\cos"
<TAN> ::= "\tan"
<CTAN> ::= "\cot"
<ASIN> ::= "\arcsin"
<ACOS> ::= "\arccos"
<ATAN> ::= "\arctan"
<INT> ::= "\int"
```

<LIMIT> ::= "\limits"

<MULTIPLY> ::= "\cdotp" | "\times"

<FRAC> ::= "\frac"

Sudėties operacija: <PLUSOP> ::= (<TERM> | <MINUSOP>) <PLUS> <EXPRESSION>

<EXPRESSION> ::= <PLUSOP> | <MINUSOP> | <TERM>

Atimties operacija: <MINUSOP> ::= <TERM> <MINUS> (<TERM> | <MINUSOP>)

<BRACKETS> ::= <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

<DECIMALNUM> ::= <NUM> <DECIMAL_POINT> <NOSIGN_NUM>

<NUM> ::= <NOSIGN_NUM> | <SIGN_NUM>

<ARG> ::= <NUM> | <DECIMALNUM> | <STRING>

<TERM> ::= <MINITERM> | <MULTIPLYOP>

<MINITERM> ::= <DIVIDEOP> | <ARG> | <BRACKETS> | <POWEROP> | <LOGOP> | <TRIG>| <DEROP>| <INTOP> | <ABSOP>

<EXPRESSION_W_CURLY> ::= <CURLY_LEFT EXPRESSION> <CURLY_RIGHT>

Dalybos operacija: <DIVIDEOP> ::= <FRAC> <EXPRESSION_W_CURLY> <EXPRESSION_W_CURLY>

Daugybos operacija: <MULTIPLYOP> ::= <MINITERM> <MULTIPLY> <TERM>

Kėlimo laipsniu operacija: <POWEROP> ::= (<ARG> | <BRACKETS>) <POWER> <EXPRESSION_W_CURLY>

Logaritmo operacija: <LOGOP> ::= <LOG> <UNDERSCORE> <EXPRESSION_W_CURLY> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

Sinuso operacija: <SINOP> ::= <SIN> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

Kosinuso operacija: <COSOP> ::= <COS> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

Tangento operacija: <TANOP> ::= <TAN> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

Kotangento operacija: <CTANOP> ::= <CTAN> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

Arksinuso operacija: <ASINOP> ::= <ASIN> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

Arkkosinuso operacija: <ACOSOP> ::= <ACOS> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

Arktangento operacija: <ATANOP> ::= <ATAN> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

Arkkotangento operacija: <ACTANOP > ::= <ACTAN> <LEFT_BRACKET> <EXPRESSION> <RIGHT_BRACKET>

<TRIG> ::= <SINOP> | <COSOP> | <TANOP> |<CTANOP> |<ASINOP> |<ACOSOP> |<ATANOP> |<ACTANOP>

Išvestinės operacija: <DEROP> ::= <DERSTART> (<ARG> | <BRACKETS>) <DERMIDLE> <STRING> <CURLY_RIGHT>

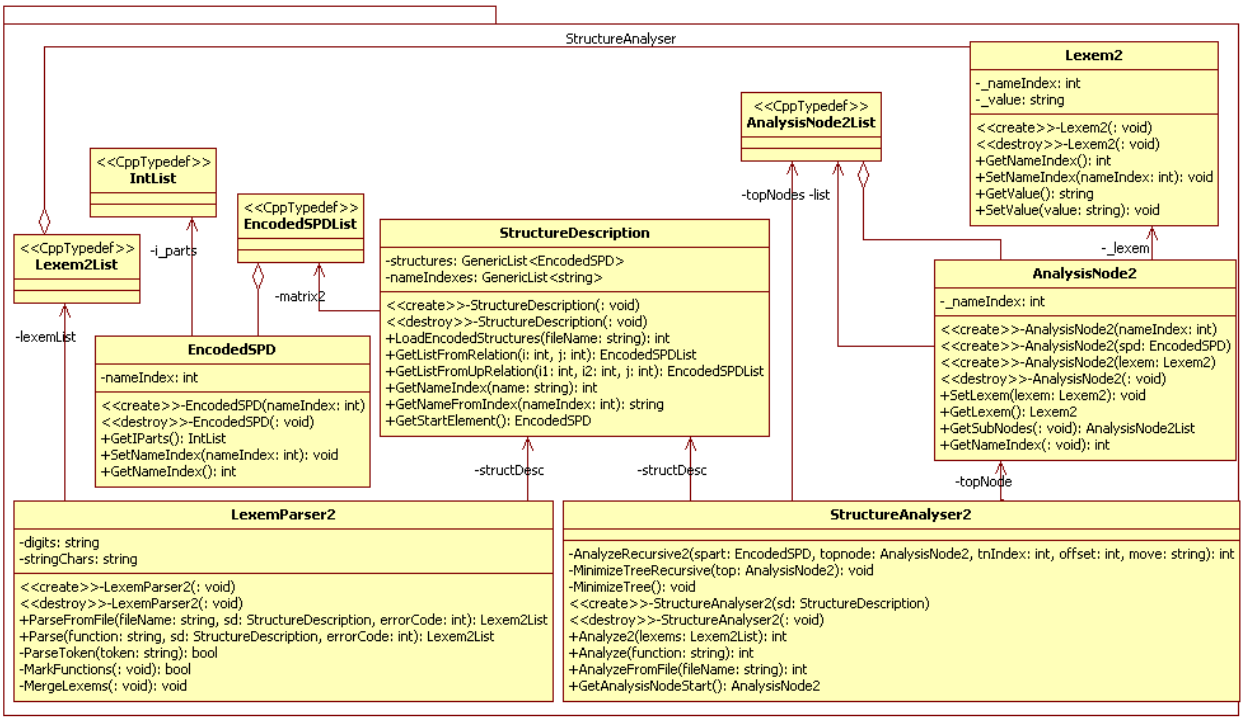
Integralo operacija: <INTOP> ::= <INT> [<LIMIT> <UNDERSCORE> <EXPRESSION_W_CURLY> <POWER> <EXPRESSION_W_CURLY>] (<ARG> | <BRACKETS>) <INTD> <STRING>

Modulio operacija: <ABSOP> ::= <ABS_BRACKET> <EXPRESSION> <ABS_BRACKET>

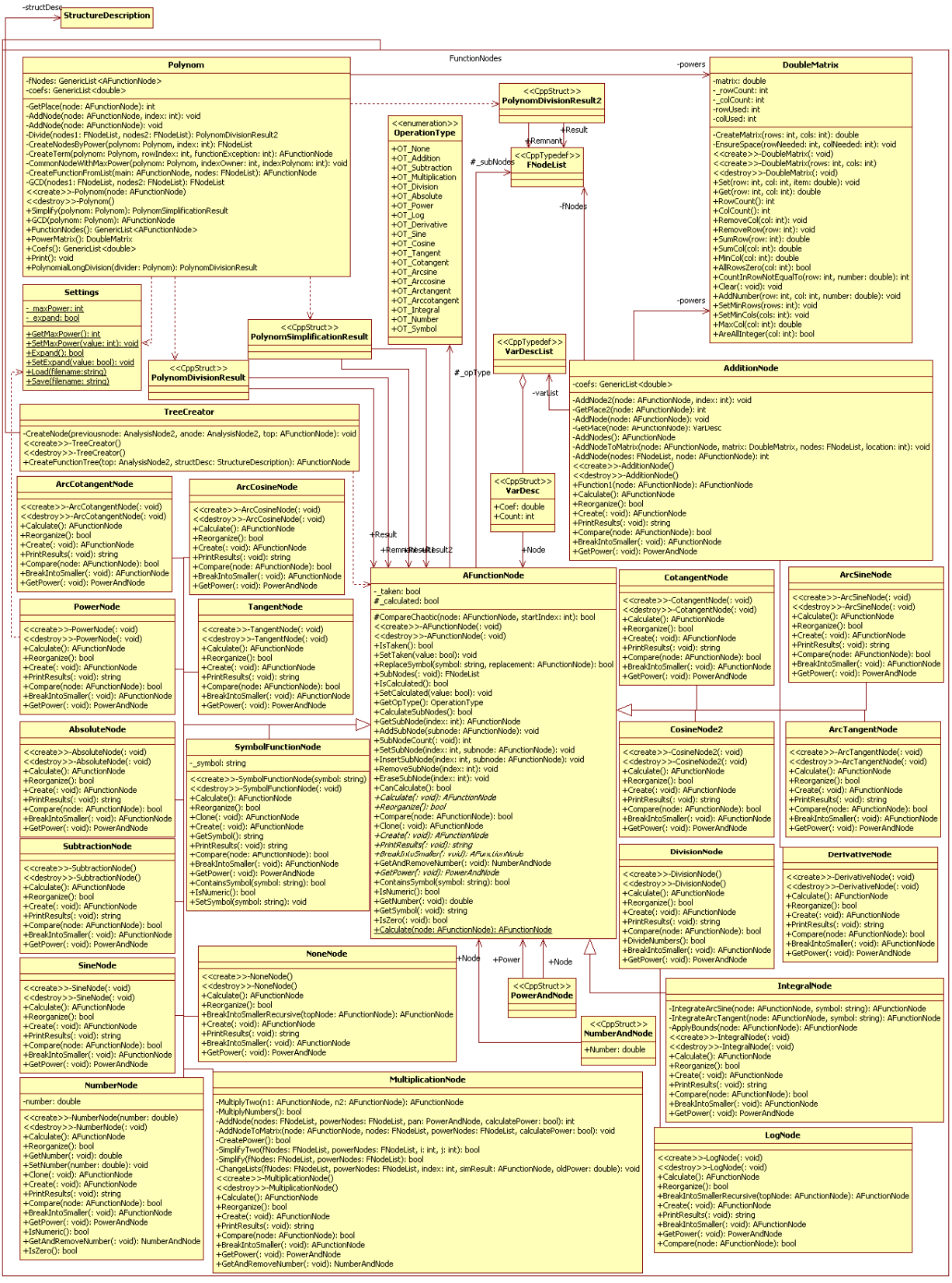
<START> ::= <DOLLAR> <EXPRESSION> <DOLLAR>

Startinis simbolis: <START>.

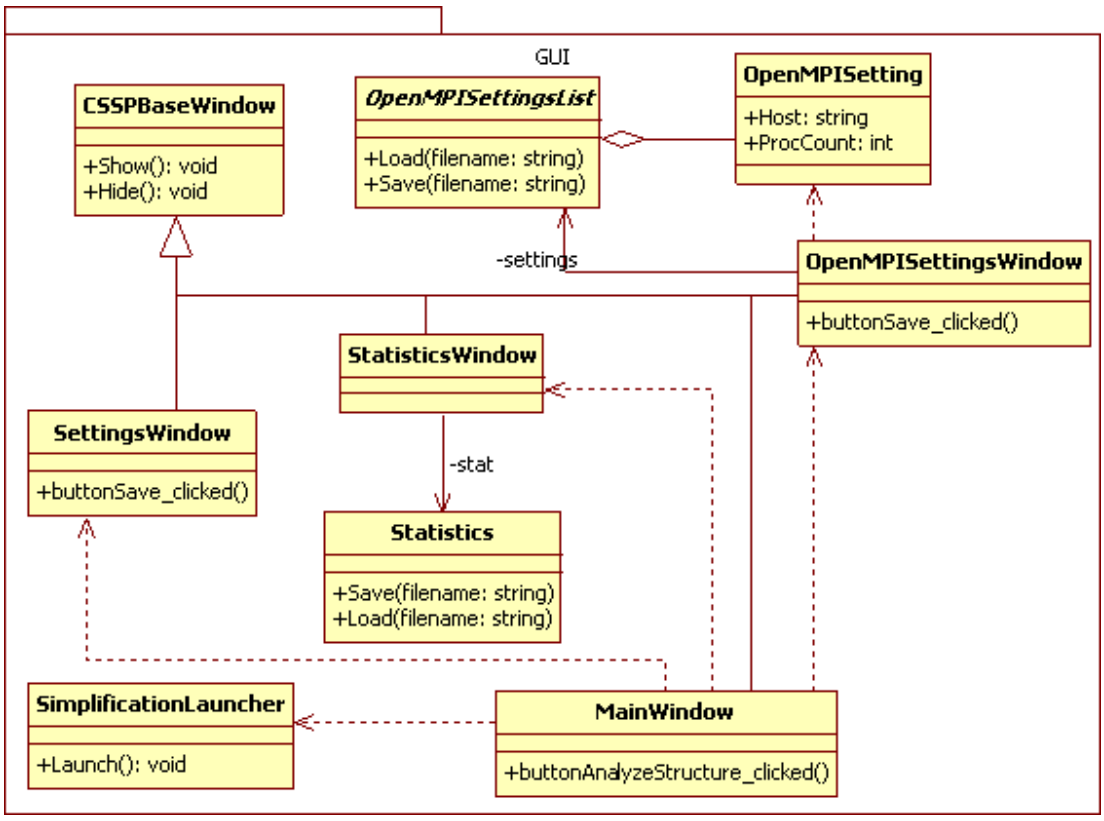
8.2 Detalios klasių diagramos



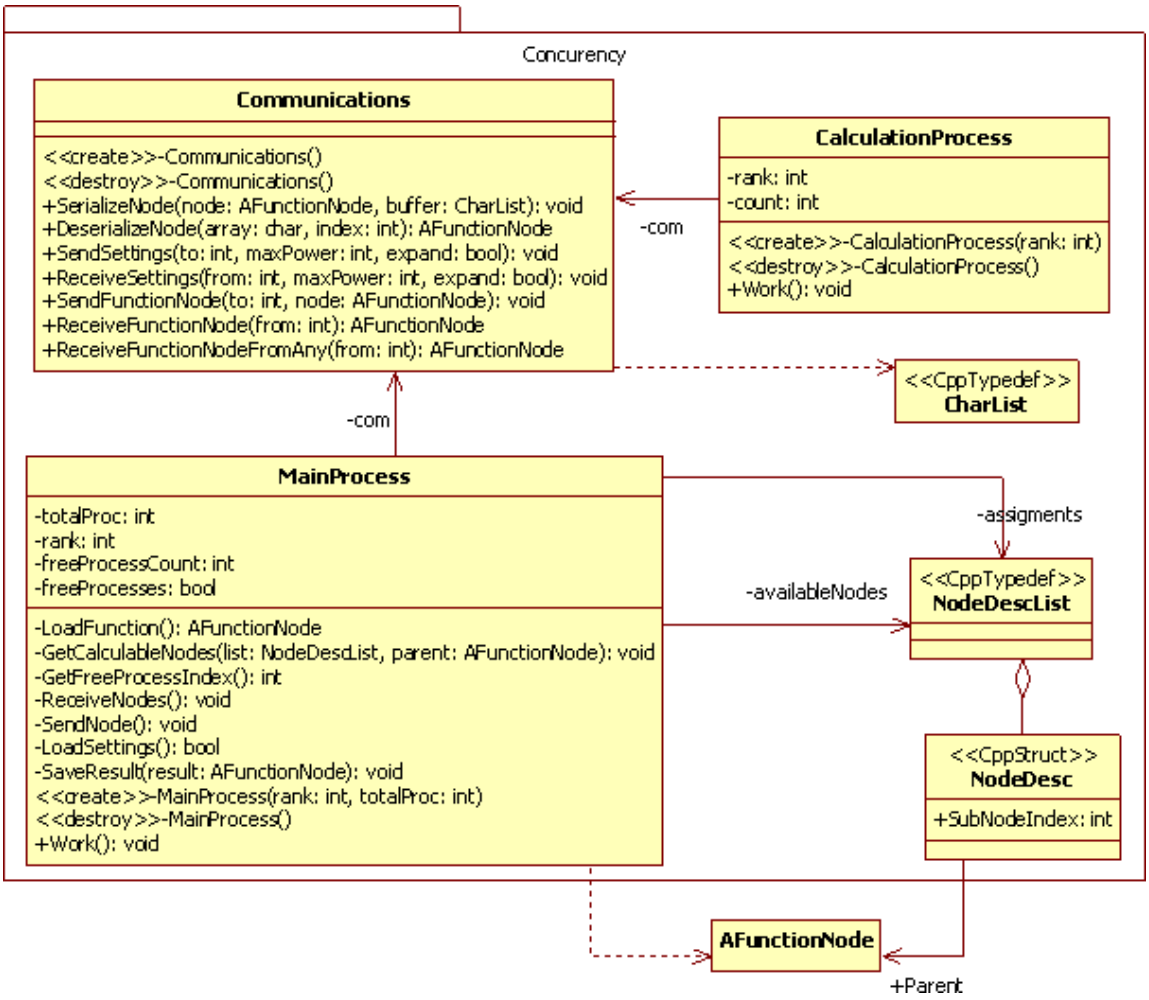
35 pav. Detali paketo „StructureAnalyser“ klasių diagrama



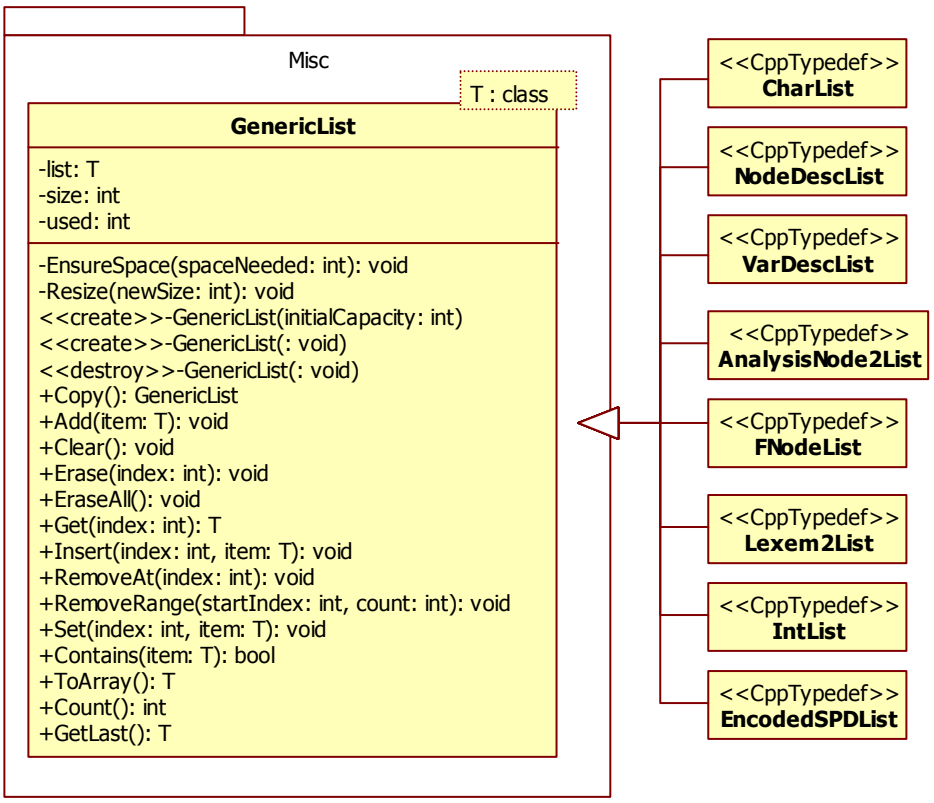
36 pav. Detali paketo „FunctionNodes“ klasių diagrama



37 pav. Detali paketo „GUI “ klasių diagrama



38 pav. Detali paketo „Concurency “ klasių diagrama



39 pav. Detali paketo „Misc“ klasių diagrama

8.3 Programos papildymas naujomis matematinėmis funkcijomis

8.3.1 Gramatikos papildymas

Pirmiausia reikia papildyti gramatikos aprašą esantį faile „struct.txt“. Gramatika aprašoma tokia forma:

X = Y Z G$

X- tai neterminalinis simbolis, kuris gali būti pakeistas dešinėje pusėje esančiais simboliais.

Y, Z, G- bet kokie neterminaliniai simboliai. Dešinėje taisyklės pusėje gali būti daugiau nei vienas arba vienas neterminalinis simbolis.

Norint aprašyti taisyklę, kurioje yra „arba“ sąlygų, ją reikia išskaidyti į kelias atskiras taisykles. Pvz.:

<X> := <G> (<Y> | <Z>) => X = G Y $
X = G Z $

Papildžius naujomis taisyklėmis gramatikos failą reikia apdoroti su programa CSSPEncoder, ji sugeneruoja failą „encodedstr.txt“, kurį naudoja gramatikos analizatorius.

Jeigu yra reikalingi nauji terminaliniai simboliai reikia papildyti LexemParser2 klasę. Jei reikia įdėti naują leksemą kuri prasideda pasvirusiu brūkšniu (pvz.: „\xxxx“), užtenka papildyti „GetFunctionIndex“ metodą. Kitu atveju reikia papildyti metodą „ParseToken“.

8.3.2 Funkcijų medžio papildymas

Visos matematinės operacijos paveldi tą pačią klasę AFunctionNode.

Svarbiausi AFunctionNode metodai:

16 lentelė. Svarbiausi „AFunctionNode" klasės metodai

Metodas	Ar būtina perkrauti	Aprašas
Calculate	Taip	Matematinės operacijos apskaičiavimo metodas.
AFunctionNode *		Gražina **NULL**, jei nieko nebuvo pakeista.
Reorganize	Taip	Pertvarko matematinę operaciją po jos argumentų apskaičiavimo.
bool		Gražina **true**- jei kas nors buvo pakeista, **false**- jei ne.
Compare	Ne	Atlieka matematinių operacijų ir jų argumentų palyginimą.
bool		Gražina **true-** jei operacijos lygios.
AFunctionNode * node		Operacija su kuria lyginama.
Clone	Ne	Klonuoja matematinę operaciją ir jos argumentus.
AFunctionNode*		Gražina klasės kloną.
Create	Taip	Sukuria tos pačios klasės objektą.
AFunctionNode*		Gražina sukurtą klasės objektą.
PrintResults	Taip	Matematinę operaciją paverčia į LaTeX eilutę.
string		Simbolių eilutė su LaTeX eilute.
BreakIntoSmaller	Taip	Suskaldo operaciją į mažesnius gabalus.
AFunctionNode*		Gražina **ne NULL**, jei pasikeitė operacijos tipas.
GetAndRemoveNumber	Ne	Išskiria koeficientą.
NumberAndNode		Gražina koeficientą ir operacijos dalį be koeficiento.
GetPower	Taip	Išskiria laipsnį.
PowerAndNode		Gražina laipsnį ir operacijos dalį be laipsnio.
Serialize	Ne	Paverčia operaciją baitų masyvu. Reikalingas tik tuo atveju jei operacija turi papildomų duomenų.
CharList * buffer		Baitų masyvas į kurį įrašoma papildomi duomenys.
Deserialize	Ne	Iš baitų masyvo atkuria objekto duomenis.

	char* array		Baitų masyvas.
	int * index		Skaitymo pozicijos žymeklis.
OperationComplexity		Taip	Apskaičiuoja operacijos kainą.
	double		Gražina kainą.
EnhanceVisual		Ne	Pertvarko operaciją, kad ją pavertus į LaTeX dokumentą, ji būtų vizualiai suprantamesnė.
	AFunctionNode*		Gražina **ne NULL-** jei pasikeitė operacijos tipas.

Kitas žingsnis yra klasės „TreeCreator“ metodo „CreateNode“ papildymas. Šis metodas iš analizės medžio sugeneruoja funkcijų medį.