VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Tiesioginio ir atbulinio išvedimo aktualijos WEB servisų sistemose

Backward and Forward Chaining in Modern WEB Service Systems

Bakalauro baigiamasis darbas

Atliko: Jonas Petraška (parašas)

Darbo vadovas: doc. dr. Vytautas Čyras (parašas)

Darbo recenzentas: lekt. Donatas Kimutis (parašas)

Vilnius – 2021

SANTRAUKA

Darbe tiriama galimybė panaudoti klasikinius tiesioginio ir atbulinio išvedimo algoritmus (toliau – FC/BC, forward chaining, backward chaining) šiuolaikinėse web servisų sistemose. Didėjantis web servisų kiekis skatina tyrinėti dinaminės kompozicijos galimybes. Šio darbo tikslas yra ištirti klasikinių išvedimo algoritmų modifikacijų naudojimo sunkumus web servisų sistemose, kurios servisus komponuoja. Darbe apžvelgiami klasikinių FC/BC algoritmų naudojimo sunkumai dėl pernelyg didelio šių algoritmų supaprastinimo. Klasikiniai FC/BC yra supaprastinti dėl modelio siaurumo, o šiuolaikinėse sistemose reikia atsižvelgti į sudėtingą pasaulį. Darbe pristatomos galimos FC/BC algoritmų modifikacijos, pritaikant juos šiuolaikinėms web servisų sistemoms. Darbe modeliuojamas einamasis pavyzdys iliustruoja web servisų kompozicijos poreikį ir komplikacijų mastą. Mūsų sukurtoje demonstracinėje sistemoje sumodeliuotos užsienio autorių tirtos modifikacijas. Darbe pateikiami pasiūlymai modifikuotų FC/BC algoritmų panaudojimui įvairiose dalykinėse srityse.

Raktiniai žodžiai: web servisai, web servisų kompozicija, tiesioginis išvedimas, atbulinis išvedimas, web servisų kokybės savybės.

SUMMARY

This paper examines the possibility of using traditional AI forward and backward chaining algorithms (BC/FC, forward chaining, backward chaining) in modern web service systems. Increasing number of services encourages to look for dynamic web service composition solutions. The objective of this paper is to investigate difficulties in using the traditional FC/BC algorithms in web service systems that requires web service composition. Difficulties occur because traditional FC/BC algorithms are oversimplified. Traditional FC/BC algorithms are not suitable for the complex real world scenarios. This paper presents possible modifications of traditional FC/BC algorithms. The motivating example presented in this paper demonstrates the difficulties and complications of modern web service composition. The demonstrational system presents different samples from other studies. This work presents possible usages of modified FC/BC algorithms in a number of different domains.

Keywords: web services, web service composition, forward chaining, backward chaining, quality of service.

TURINYS

ĮVADAS **6**

1. TYRIMAS **8**

1.1. Kodėl FC/BC nebėra tinkami naudojimui šiuolaikinėse web servisų sistemose? 8

1.2. Iliustruojantys pavyzdžiai 9

2. EINAMASIS PAVYZDYS **11**

3. FC/BC ALGORITMAI **13**

3.1. Produkcijų sistema 13

3.2. Klasikiniai tiesioginio ir atbulinio išvedimo algoritmai 14

3.3. Klasikinių algoritmų modifikacijos 16

3.3.1. Daugiau nei vieno tikslo palaikymas 16

3.3.2. Produkcijų vykdymas lygiagrečiai 17

3.3.3. Perteklinių produkcijų pašalinimas 17

4. PASLAUGOS KOKYBĖS SAVYBĖS **19**

4.1. Bendros paslaugos kokybės savybės 19

4.2. Paslaugos kokybės savybės, specifinės web servisų grupei 21

4.3. Paslaugos kokybės balo apskaičiavimas 23

4.4. Web servisų rikiavimas pagal paslaugos kokybės savybes 23

5. DEMONSTRACINĖ SISTEMA **24**

5.1. Technologijos ir architektūra 24

5.1.1. Išorinė sistemos dalis 24

5.1.2. Vidinė sistemos dalis 25

5.2. Įgyvendinti scenarijai 26

5.2.1. Standartinis scenarijus 26

5.2.2. Web serviso kokybės balo kitimo scenarijus 27

5.2.3. Vartotojo pasirinkimų kitimo scenarijus 28

5.3. Galimi išplėtimai ir panaudojimas 30

6. MŪSŲ DARBO SANTYKIS SU BEKKOUCHE ET AL. (2017) TYRIMU 32

REZULTATAI.....33

IŠVADOS 33

ŠALTINIAI 34

PRIEDAI 36

1 priedas. Klasikinio atbulinio išvedimo algoritmo pseudokodas.....36

2 priedas. Klasikinio tiesioginio išvedimo algoritmo pseudokodas 37

3 priedas. Modifikuoto tiesioginio išvedimo algoritmo pseudokodas.....38

4 priedas. Produkcijų grupavimo lygiagrečiam vykdymui C# programavimo kalbos kodas....39

5 priedas. Kompozicijos planavimo bei vykdymo rezultatų pavyzdys 40

ĮVADAS

Tyrimo objektas. Prieštaravimai taikant klasikinius tiesioginio ir atbulinio išvedimo algoritmus šiuolaikinėse web servisais grindžiamose sistemose.

Nagrinėjama problema. Ar tiesioginio ir atbulinio išvedimo algoritmai (toliau – FC/BC, forward chaining, backward chaining) tinka web servisais grindžiamoms sistemoms?

Tikslas. Ištirti klasikinių išvedimo algoritmų modifikacijų naudojimo sunkumus web servisų sistemose, kurios servisus komponuoja, ir sumodeliuoti mūsų sukurtoje demonstracinėje sistemoje užsienio autorių tirtas modifikacijas.

Uždaviniai:

1. Identifikuoti priežastis, kodėl klasikiniai FC/BC algoritmai gali sunkiai tikti web servisų kompozicijai šiuolaikinėse sistemose. Priežastis iliustruoti pavyzdžiais.
2. Sukurti demonstracinį pavyzdį, iš kurio matyti būtinybė modifikuoti klasikinius FC/BC.
3. Pademonstruoti sukurto pavyzdžio modifikavimo scenarijus šitaip modifikuojant klasikinius FC/BC.
4. Užprogramuoti demonstracinę sistemą ir joje įgyvendinti pasiūlytus scenarijus.
5. Pateikti pasiūlymus, kaip modifikuoti FC/BC algoritmai gali būti panaudoti šiuolaikinėse web servisų sistemose.

Aktualumas. Web servisai atsirado sprendžiant skirtingų sistemų tarpusavio sąveikos problemą. Didėjant sistemų, sukurtų naudojant skirtingas programavimo kalbas ir karkasus, kiekiui darėsi sudėtinga tarp jų užtikrinti tikslią ir standartizuotą komunikaciją. Kaip šios problemos sprendimas atsirado web servisai, kurie suteikia sistemoms, sukurtoms naudojant skirtingas programavimo kalbas ir karkasus, tikslų ir standartizuotą metodą tarpusavio sąveikai.

Šiais laikais web servisų kiekis didelėse sistemose siekia šimtus ar net tūkstančius, todėl kyla sunkumai juos valdant ir orkestruojant vartotojo užklausų atlikimą [KKC11]. Vienas iš tokių sunkumų yra servisų tarpusavio kompozicija. Turint tam tikrą kiekį web servisų ir apdorojant kompleksišką užklausą, vienas web servisas nebūtinai gali tai padaryti. Kurti atskirą servisą dėl vienos užklausos irgi ne išeitis. Todėl atsiranda poreikis išnaudoti esamus web servisus užklausai patenkinti. Turint mažiau web servisų, jų kompozicija galima rankiniu būdu, tačiau web servisų kiekiui sistemoje augant, rankiniu būdu jų sukomponuoti neįmanoma [KKC11]. Todėl atsiranda poreikis algoritmams ir karkasams, kurie gebėtų, turėdami web servisų rinkinį bei vartotojo užklausą, automatiškai, greitai bei patikimai surasti kelią (web servisų kvietimo seką) vartotojo

užklausai patenkinti. Pirmieji bandymai išspręsti šią problemą buvo sutelkti į dirbtinio intelekto panaudojimą transformuojant problemą į planavimo problemą [Pee05]. Tai padarius buvo naudojami klasikiniai dirbtinio intelekto tiesioginio ir atbulinio išvedimo (planavimo) algoritmai. Šie algoritmai randa kelią (web servisų kvietimo seką), kuris atsakytų į vartotojo užklausą.

Paprastoms užklausoms FC/BC tinka, bet sudėtingoms gali ir netikti. Šiais laikais web servisų kompozicijai svarbu rasti ne tik greitą ir patikimą kelią, bet kelią, kuris geriausiai tenkina vartotojo pateiktą užklausą atsižvelgiant ne tik į prašomą rezultatą, bet ir į paslaugos kokybės (angl. quality of service, toliau QoS) reikalavimus [BBH+17]. Tokie reikalavimai yra, pavyzdžiui, vykdymo kaina, vykdymo trukmė, reputacija/patikimumas, pasisekimo reitingas bei kiti, dažnai dalykiniai sričiai specifiniai, reikalavimai. Todėl atsirado poreikis lankstesniems algoritmams, kurie ieškodami kelio vartotojo užklausai patenkinti atsižvelgtų ne tik į semantines web serviso savybes (pvz., pora „įeitis-išeitis“), bet ir į nefunkcines savybes.

Pasaulyje yra atliekami tyrimai, kaip išplėsti klasikinių tiesioginio ir atbulinio išvedimo algoritmų taikymo galimybes, kai produkcinių taisyklių semantika vaizduojama ne tik funkcinėmis, bet ir nefunkcinėmis savybėmis. Pavyzdžiui, FC algoritmas kombinuojamas su hibridiniu rikiavimo algoritmu, kuris FC algoritmo rastus kelius surikiuoja pagal jų atitikimą nefunkcinėms vartotojo užklausos savybėms [NWG16]. Kitame darbe [CCC18] Chen, Cardozo ir Clarke pristato BC algoritmo modifikaciją, kuri kelio paieškoje atsižvelgia į kelio patikimumą ir šitaip padidina pasisekimo tikimybę. Yra siūlomos ne tik senų tiesioginio ir atbulinio išvedimo algoritmų modifikacijos, bet ir nauji algoritmai. Pavyzdžiui, Kuzu ir Cicekli [KC12] siūlo naują karkasą, kuris ne tik greitai suranda tinkamą kelią vartotojo užklausai patenkinti, bet ir sugeba plano vykdymo metu prisitaikyti prie netikėtų pasikeitimų (pvz., vienas iš servisų tampa nepasiekiamas, pasikeičia serviso kvietimo kaina ir t.t). Kitam darbe [YZZ+19] Yan, Zhang ir kiti pristato naują dinaminį sprendimą servisų atrinkimui. Šis sprendimas grindžiamas adaptyviu globalių paslaugos kokybės (angl. QoS) reikalavimų skaidymų į lokalius reikalavimus (angl. adaptive QoS constraints decomposition).

1. TYRIMAS

1.1. Kodėl FC/BC nebėra tinkami naudojimui šiuolaikinėse web servisų sistemose?

Nagrinėjant literatūrą galima rasti įvairių argumentų, kodėl įprasti tiesioginio ir atbulinio išvedimo algoritmai nebėra tinkami naudojimui šiuolaikinėse web servisų sistemose ir kodėl juos reikia modifikuoti ar keisti naujais algoritmais. Trys pagrindiniai argumentai aptariami žemiau.

Pirmas argumentas – nefunkcinės užklausos savybės. Jis siejamas su tuo, kad klasikiniai planavimo algoritmai atsižvelgia tik į semantines užklausos savybes. Modeliuoti nefunkcines užklausos savybes produkcinėmis taisyklėmis nebuvo pirmaeilė planavimo algoritmo paskirtis. Bekkouche, Benslimane, Huchard ir kiti savo darbe [BBH+17] pristato tokį algoritmą, kuris atsižvelgia į vartotojo užduotus paslaugos kokybės (angl. quality of service, toliau QoS) reikalavimus. Pastarasis algoritmas grindžiamas grafais ir harmoninės paieškos algoritmu. Jų straipsnyje pabrėžiama, jog klasikiniai dirbtinio intelekto planavimo algoritmai, tarp kurių yra ir tiesioginio bei atbulinio išvedimo algoritmai, nors ir randa kelią vartotojo užklausai patenkinti, tačiau neatsižvelgia į vartotojo iškeltus QoS reikalavimus. Tai, pavyzdžiui, yra vykdymo kaina, vykdymo trukmė, reputacija/patikimumas, pasisekimo reitingas bei kiti, dalykinei sričiai specifiniai, reikalavimai. Taigi klasikinių FC/BC algoritmų neatsižvelgimas į šiuos reikalavimus verčia ieškoti būdų, kaip juos modifikuoti arba ieškoti alternatyvių algoritmų.

Antras argumentas – algoritmo vykdymo laikas. Jis siejamas su klasikinių planavimo algoritmų greitaveika. Khakhkhar, Kumar ir Chaudhray savo darbe [KKC12] pristato naują algoritmą dinaminei web servisų kompozicijai. Šiame straipsnyje pabrėžiama, jog klasikinių planavimo algoritmų FC ir BC vykdymo laikas didėja greičiau negu tiesiškai, kai web servisų skaičius n auga. FC algoritmas pradeda nuo faktų ir eina link tikslo. BC algoritmas pradeda nuo tikslo ir eina link faktų. Algoritmas, kuris pristatomas [KKC12], kombinuoja FC ir BC ir vienu metu eina nuo faktų link tikslo ir nuo tikslo link faktų, taip surasdamas sprendimą viduryje. Šis algoritmas apdoroja mažiau servisų bei yra žymiai greitesnis nei FC ir BC algoritmai atskirai. Šiais laikais web servisų sistemas sudaro šimtai ar tūkstančiai web servisų ir klasikiniai algoritmai netinka, nes yra per lėti, todėl tenka juos modifikuoti.

Trečias argumentas – dinamika. Web servisų aplinka kinta, tai yra, vieni web servisai pranyksta, kiti atsiranda. O klasikiniai planavimo algoritmai yra statiški – nemodeliuoja nuolat kintančios aplinkos. Chen, Cardozo ir Clarke [CCC18] pristato GoCoMo (goal-driven service composition model in mobile and persasive computing environments) algoritmą. Šis algoritmas grindžiamas atbuliniu išvedimu ir palaiko sudėtingesnius web servisų kompozicijos scenarijus. Taip pat, jis dinamiškai keičia vykdymo kelią atsižvelgdamas į aplinkos pokyčius, pavyzdžiui,

servisų prieinamumo pasikeitimą, servisų atsakymo laiko pasikeitimą ir pan. Kuzu ir Cicekli [KC12] pristato naują dinamišką web servisų kompozicijos karkasą (angl. framework). Įprastai web servisų kompozicija vyksta dviem žingsniais, pradžioje planavimas ir paskui plano vykdymas. Planavimo žingsnyje yra sudaromas planas, kitais žodžiais, web servisų kvietimo seka. Plano vykdymo žingsnyje planas yra vykdomas jau su konkrečiomis įeities reikšmėmis. Tai padaro klasikinius planavimo algoritmus statiškus. Naujai siūlomas karkasas apjungia šiuos du žingsnius, kad išspręsti statiškumo problemą. Pradinis planas yra sudaromas, kaip ir įprastai, planavimo žingsnyje. Plano vykdymo metu kiekviename žingsnyje yra stebimas web servisų sąrašas, kuris dalyvauja kompozicijoje. Atsiradus pokyčiams, pavyzdžiui, dingus servisui arba atsiradus naujam servisui, įvyksta perplanavimas. Perplanavimo metu, pagal naują web servisų sąrašą, sudaromas naujas planas. Naujas planas sudaromas tik tai daliai web servisų kvietimo sekos, kuri buvo paveikta naujai atsiradusių ar dingusių servisų. Perplanavimui įvykus, plano vykdymas toliau vyksta įprastai.

1.2. Iliustruojantys pavyzdžiai

Literatūroje yra realaus pasaulio pavyzdžių, kurie atspindi išvardintų argumentų aktualumą. Pavyzdžiui Bekkouche, Benslimane, Huchard ir kiti savo darbe [BBH+17] aptaria automobilių pirkimo pavyzdį, kuris atspindi poreikį FC/BC algoritmams atsižvelgti į nefunkcines užklausos savybes. Įsivaizduokime, jog Jonas planuoja įsigyti naudotą automobilį, jis žino jo gamintoją, modelį ir preliminarią ridą. Jonas nori gauti pilną paslaugų paketą – sužinoti apytikslę automobilio kainą, gauti automobilio istorijos ataskaitą, gauti apytikslę draudimo kainą bei galiausiai gauti pasiūlymą automobilio finansavimui. Visos šio paketo paslaugos yra teikiamos atskirų verslų. Kiekvienai paslaugai yra ne vienas verslas, kuris teikia ją, t. y. visada yra konkurentų. Jonas kiekvienai paslauga gali apibrėžti tam tikrus kriterijus, pavyzdžiui, didžiausią skiriamą sumą draudimui, didžiausią metinių palūkanų normą finansavimui ir pan. Jų darbe šis pavyzdys yra einamasis iliustruojant autorių naujai pristatomą web servisų kompozicijos algoritmą. Kiekvieną paslaugą teikiantį verslą atitinka web servisas, kuris turi paslaugos kokybės savybes: atsakymo laiką (angl. response time), kainą (angl. cost), prieinamumą (angl. availability), sėkmingumą (angl. success rate), patikimumą (angl. reputation). Būtent į šias paslaugos kokybės savybes nesugeba atsižvelgti klasikiniai FC/BC algoritmai, dėl to tenka juos modifikuoti arba ieškoti alternatyvių algoritmų, ką ir padarė darbo autoriai.

Kitame darbe [CCC18], Chen, Cardozo ir Clarke pristato apsipirkimo prekybos centre pavyzdį, kuris iliustruoja FC/BC algoritmų statiškumą. Tegu, Marytė su savo vienerių metų sūnumi atvyko į prekybos centrą. Marytė nori sužinoti greičiausią kelią iki artimiausios sauskelnių parduotuvės, o vėliau iki artimiausio mamos ir vaiko kambarėlio, kur galėtų sūnui pakeisti

sauskelnes. Tai Marytė norėtų pasiekti pasinaudojus savo išmaniuoju laikrodžiu. Prekybos centras siūlo mobiliąją programėlę arba internetinę svetainę, kur galima lengvai surasti kelią iki reikiamos parduotuvės. Deja, Marytės išmanusis laikrodis nepalaiko mobiliosios programėlės, o jo ekranas yra per mažas, kad matyti kelią internetinėje svetainėje. Prekybos centre yra išmaniųjų įrenginių, kurie priklauso tiek klientams, tiek paslaugų tiekėjams ar aptarnaujančiam personalui. Kiekvienas iš šių įrenginių savo resursus, tokius kaip GPS, navigacija, prekybos centro žemėlapiai ir pan. gali paversti prieinamus per bendrą prekybos centro tinklą kaip paslaugas (angl. service). Tokia prekybos centre esančių įrenginių visuma vadinama aprėpiančiąja skaičiavimo aplinka (angl. pervasive computing environment). Marytės išmanusis laikrodis geba komunikuoti su šioje aplinkoje esančiais įrenginiais ir taip panaudoti jų resursus. Pavyzdžiui, Marytės išmanusis laikrodis specialiai tam skirtu kanalu gali tiesiogiai gauti garso instrukcijas, kurias suformuotų ir skleistų kiti įrenginiai esantys tam pačiam tinkle. Prekybos centre esančių išmaniųjų įrenginių kiekis ir prieinamumas nuolat kinta, todėl reikia algoritmo/karkaso kuris sugeba realiu metu, pagal vartotojo pateiktą užklausą, sugeneruoti servisų vykdymo planą bei vykdymo metu reaguoti į servisų rinkinio pasikeitimus ir pergeneruoti jau anksčiau sugeneruotą planą. Klasikiniai FC/BC algoritmai nesugeba to padaryti, nes jie yra statiški.

2. EINAMASIS PAVYZDYS

Mūsų darbe panaudosime [BBH+17] pristatomą pavyzdį bei išplėsime jį iki konkretaus pavyzdinio scenarijaus su konkrečiais web servisais. Pavyzdys yra skirtas ištirti kaip išplėtus FC/BC algoritmus galima išspręsti kompleksiškesnius scenarijus atsižvelgiant tiek į semantines, tiek į nefunkcines web serviso savybes.

Tegu, Jonas ketina įsigyti naudotą automobilį. Jis žino automobilio gamintoją, modelį ir pagaminimo metus. Jonas nori įsigyti ne tik automobilį, bet kartu gauti ir automobilio draudimą, automobilio istoriją, gal būt gauti automobilio finansavimo galimybes (paskolos ar lizingo sąlygas) bei sužinoti automobilio taršos mokestį. Jonas, jeigu nori, gali gauti paskolos ar lizingo pasiūlymą. Jonas gali nurodyti maksimalią paskolos metinių palūkanų normą, lizingo laikotarpį, lizingo pradinės įmokos dydį procentais ar lizingo likutinę procentinę vertę.

Demonstracinėje sistemoje paslaugų tiekėjai skirstomi į grupes pagal teikiamą paslaugą. Yra šešios grupės: automobilių paieškos, automobilių istorijos, draudimo, paskolos, lizingo bei registracijos. Kiekvienoje grupėje gali būti daug konkurentų, kurie teikia to paties tipo paslaugą. Mes pagal grupes sistemoje turime tokius tiekėjus (dalis suteiktų pavadinimų atspindi tikrų Lietuvoje esančių paslaugos tiekėjų pavadinimus):

- Automobilių paieškos paslauga – Autogidas, Autoplius.
- Automobilių istorijos paslauga – Car vertical, Car history.
- Draudimo paslaugos – Lietuvos draudimas, BTA.
- Paskolos paslaugos – Big bank.
- Lizingo paslaugos – Swedbank.
- Registracijos paslaugos – Regitra.

Kiekviena grupė turi apibrėžtą semantinę schemą (pora, įeitis – išeities):

- Automobilių paieškos paslauga
    - Įeitis: gamintojas, modelis, pagaminimo metai.
    - Išeitis: valstybinis numeris, VIN kodas, rida, kaina.
- Automobilių istorijos paslauga
    - Įeitis: VIN kodas.
    - Išeitis: savininkų skaičius, kilmės šalis.
- Draudimo paslauga
    - Įeitis: asmens kodas, valstybinis numeris.
    - Išeitis: metinė draudimo įmoka.

- Paskolos paslauga
    - Įeitis: asmens kodas, suma, didžiausia metinė palūkanų norma.
    - Išeitis: mėnesinė įmoka, metinė palūkanų norma, trukmė metais.
- Lizingo paslauga
    - Įeitis: asmens kodas, suma, trukmė metais, pradinis įnašas procentais, likutinė procentinė vertė.
    - Išeitis: mėnesinė įmoka, pradinė įmoka, likutinė vertė.
- Registracijos paslauga
    - Įeitis: VIN kodas.
    - Išeitis: registracijos (taršos) mokestis.

Darbe [BBH+17] pristatomas web servisų kompozicijos algoritmas veikia keliais žingsniais:

- Pirmiausiai yra sugeneruojamas grafas, kuriame pateikiamos visų web servisų semantinės sąsajos. Tam tikslui yra skaičiuojamas semantinis web servisų panašumo rodiklis, kuris nusako, kiek vieno serviso įeities (angl. input) yra panaši į kito serviso išeitį (angl. output). Pagal tai grafe sužymimos visos įmanomos kompozicijos, kurios gali patenkinti vartotojo pateiktą užklausą.
- Antra, naudojantis harmoninės paieškos algoritmu (angl. harmony search), pirmame žingsnyje atrinktos kompozicijos yra surikiuojamos pagal jų atitikimą vartotojo pateiktoms paslaugos kokybės savybėms (angl. quality of service).
- Galiausiai, išrikiavus visas kompozicijas, grąžinama kompozicija, kuri labiausiai atitinka tiek semantines, tiek nefunkcines užklausos savybes.

Šiame darbe siūlome alternatyvų algoritmą, kuris naudoja FC/BC:

- Pirmiausiai, naudojantis FC/BC algoritmais yra sudaromas planas – web servisų kvietimo seka. Web servisų kvietimo seka gaunama lyginant semantines (įeities – išeities) web servisų savybes.
- Toliau kiekvienai kompozicijai priskiriamas balas. Kiekvienas web servisas turi kokybės savybes, pavyzdžiui, patikimumą, prieinamumą ir t. t., kurios turi savo svorį bei balą. Susumavus visų kompozicijoje esančių web servisų balus, gauname kompozicijos balą.
- Naudojantis anksčiau priskirtais balais, kompozicijos išrikiuojamos mažėjimo tvarka. Kuo balas didesnis, tuo kompozicija yra labiau atitinkanti vartotojo užklausą bei tuo kompozicija yra geresnė.
- Galiausiai, aukščiausią balą turinti kompozicija grąžinama vykdymui.

3. FC/BC ALGORITMAI

3.1. Produkcijų sistema

Klasikinius tiesioginio ir atbulinio išvedimo algoritmus paaiškinsime per produkcijų sistemą, plačiau apie produkcijų sistemą žr. [Cyr21]. Produkcijų sistema yra trejetas:

- Globali duomenų bazė (toliau – GDB),
- Produkcijų aibė,
- Valdymo sistema.

GDB yra duomenų struktūra. Priklausomai nuo konteksto GDB gali būti tiek įprasta matrica, tiek kompleksiška duomenų bazių sistema. Produkcijų aibę sudaro produkcijos – taisyklės, kurios yra pritaikomos GDB ir po kurių pritaikymo GDB pereina iš vienos būsenos į kitą. Kurią produkciją panaudoti nusprendžia valdymo sistema – algoritmas, kuris pagal tam tikras sąlygas atrenka tinkamiausią produkciją ir ją pritaiko ant GDB. Pritaikius produkciją tikrinama ar GDB tenkina terminalinę sąlygą, jei taip, algoritmas baigią darbą ir grąžiną produkcijų seką, kurios paverčia pradinę GDB būseną į galutinę GDB būseną. Jei ne – valdymo sistema bando toliau taikyti produkcijas. Jei valdymo sistema daugiau nebegali pritaikyti produkcijų, tai sprendinys neegzistuoja. Kaip pavyzdį paimkime žirgo maršruto radimo uždavinį (angl. Knights tour). Plačiau apie šį uždavinių bei jo sprendimą per produkcinę sistemą žr. [Cyr21]. Trumpai, uždavinys yra toks: žirgas pradeda apatiniame kairiame lentos kampe ir turi apeiti visą N*N dydžio lentą, langelius aplankant tik vieną kartą. Terminalinė būsena yra tada, kai visi lentos langeliai yra užpildyti skaičiais nuo 1 iki N*N. Uždavinys išreikštas per produkcijų sistemą:

- GDB – dvimatis sveikųjų skaičių masyvas vaizduojantis kvadratinę languotą lentą,
- Produkcijų aibė – visi galimi žirgo ėjimai. Žirgas gali judėti tik L raidės forma – du langelius horizontaliai arba vertikaliai ir vieną statmenai judėjimo krypčiai,
- Valdymo sistema – algoritmas, kuris nusprendžia kuria kryptimi žirgui eiti, kad pasiektų galutinį tikslą. Algoritmo pavyzdį Pascal kalba galima rasti [Cyr21].

3.2. Klasikiniai tiesioginio ir atbulinio išvedimo algoritmai

Tiesioginio ir atbulinio išvedimo algoritmus paaiškinsime naudodami einamąjį pavyzdį, kuris buvo pristatytas anksčiau. Išreiškime šį pavyzdį per produkcijų sistemą. Turime šešias paslaugos tiekėjų grupes: automobilio paieškos paslaugos tiekėjai (P1), automobilio istorijos paslaugos tiekėjai (P2), draudimo paslaugos tiekėjai (P3), finansavimo (paskolos – P4 ar lizingo - P5) paslaugos tiekėjai bei registracijos paslaugos tiekėjai (P6). Kiekviena grupė atitinka šias produkcijas:

P1 – automobilio paieškos paslaugos tiekėjai: G, M, PM -> VN, VIN, R, K, kur G – gamintojas, M – modelis, PM – pagaminimo metai, VN – valstybinis numeris, VIN – kėbulo numeris, R – rida, K – kaina;

P2 – automobilio istorijos paslaugos tiekėjai: VIN -> SS, S, kur SS – savininkų skaičius, S – pirmos registracijos šalis;

P3 – automobilio draudimo paslaugos teikėjai: AK, VN -> MM, kur AK – asmens kodas, MM – metinis draudimo mokestis;

P4 – paskolos paslaugos tiekėjai: AK, K, MPP -> MPI, MPCT, LM, kur MPP – maksimali metinė palūkanų norma, MPM – mėnesinė įmoka, MPCT – metinė palūkanų norma, LM – laikotarpis metais;

P5 – lizingo paslaugos tiekėjai: AK, K, LM, PPCT, LPCT -> MLI, PI, L, kur PPCT – pradinė įmoka procentais, LPCT – likutinė vertė procentais, MLI – mėnesine lizingo įmoka, PI – pradinė įmoka, L – likutinė vertė;

P6 – registracijos paslaugos tiekėjai: VIN -> RM, kur RM – registracijos (taršos) mokestis.

Kiekviena produkcija savo kairėje pusėje turi faktus (antecedentą), o dešinėje pusėje – išvadą (konsekventą). Produkcija atitinka funkciją, kurios įeitis yra antecendentas, o išeitis yra konsekventas. Vartotojas sistemai paduoda sąrašą faktų bei pasako, kokių paslaugų nori. Faktai sudaro pradinę GDB būseną, kiekvienos produkcijos įvykdymas pakeičia GDB būseną pridėdamas naujų faktų arba tikslų. GDB būsena terminalinę sąlygą atitinka, kai toje būsenoje esantys faktai atitinka vartotojo norimų gauti paslaugų faktus. Atitikus terminalinę būseną, grąžinamas kelias (produkcijų seka) reikalingas, jog pradinę GDB būseną paversti galutine. Valdymo sistema šiuo atveju yra tiesioginio arba atbulinio išvedimo algoritmai.

Tiesioginio išvedimo algoritmas pradeda samprotavimą nuo faktų. Tiesioginio išvedimo algoritmas yra iteracinis algoritmas. Produkcijos yra perrankamos iteracijomis, kiekvienoje iteracijoje produkcijos perrankamos iš eilės. Kiekvienoje iteracijoje panaudotos produkcijos yra

pažymimos, jog nebūtų naudojamos sekančiose iteracijoje. Algoritmas baigia savo darbą, kai pasiekiama terminalinė būsena. Jei perrinkus visas produkcijas ji nėra pasiekiama, tai ieškoma produkcijų seka neegzistuoja.

Atbulinio išvedimo algoritmas, kitaip negu tiesioginio išvedimo algoritmas, pradeda nuo tikslo ir eina atbulomis link faktų. Algoritmas pradeda nuo tikslo ir perrenka visas produkcijas pradedant pirmąją ieškodamas tokios produkcijos, kurios konsekventas sutaptų su tuometiniu tikslu. Suradęs tokią produkciją jis jos antecedentą paverčia nauju tikslu ir toliau kartoja paiešką. Atbulinio išvedimo algoritmas yra rekursyvus. Algoritmas baigia savo darbą, kai suranda produkciją, kurios antecedentas atitinka faktus – pasiekia terminalinę būseną. Jei tokios produkcijos algoritmas neranda, tai ieškoma produkcijų seka neegzistuoja.

Išsamesnį šių algoritmo paaiškinimą bei pavyzdžius galima rasti [Cyr21]. Klasikinio atbulinio išvedimo algoritmo pseudokodas pateiktas Priede Nr. 1, o klasikinio tiesioginio išvedimo algoritmo pseudokodas pateiktas Priede Nr. 2. Tiesioginio ir atbulinio algoritmų įgyvendinimas C# programavimo kalba bei panaudojimo pavyzdžiai pateikti GitHub talpykloje https://github.com/JonasPetraska/BackwardsForwardsChaining.

3.3. Klasikinių algoritmų modifikacijos

3.3.1. Daugiau nei vieno tikslo palaikymas

Klasikiniai FC/BC algoritmai sugeba apdoroti produkcijas tik su vienu faktu konsekvente. Šiuolaikiniai web servisai retai kada grąžina vieną rezultatą, dažniausia jie grąžina aibę rezultatų. Todėl norint su FC/BC algoritmais įgyvendinti ankščiau pristatytą pavyzdį teko juos modifikuoti, jog jie sugebėtų apdoroti produkcijas su daugiau nei vienu faktu konsekvente.

Modifikuotas BC algoritmas nuo klasikinio skiriasi tik jo panaudojimu. Turint aibę faktų konsekvente BC algoritmas yra kviečiamas kiekvienam faktui. Visų iškvietimų metu produkcijų, išvestų faktų ir kiti sąrašai yra tie patys ir jie yra papildomi su kiekviena iteracija. Lentelėje Nr. 1 pateiktas klasikinio BC algoritmo iškvietimo pseudokodo pavyzdys, o lentelėje Nr. 2 palyginimui pateiktas modifikuoto BC algoritmo iškvietimo pseudokodo pavyzdys.

1 lentelė. Klasikinio atbulinio išvedimo algoritmo iškvietimo pseudokodo pavyzdys

```
char tikslas = 'A';
bool rezultatas = BC(tikslas);

if (rezultatas == true)
        {spausdinamas kelias – produkcijų sąrašas}
```

2 lentelė. Modifikuoto atbulinio išvedimo algoritmo iškvietimo pseudokodo pavyzdys

```
char[] tikslai = ['A', 'B', 'C'];
foreach (tikslas in tikslai) //pereiname per visus tikslus ir kiekvienam iškviečiame BC
algoritmą
{
        //kiekvienos iteracijos metu visi sąrašai (produkcijų, išvestų faktų ir t.t.) lieka tie patys
– yra papildomi

        bool rezultatas = BC(tikslas);
        if (!rezultatas)
        {
                break;
        }
}

if (rezultatas == true)
        {spausdinamas kelias – produkcijų sąrašas}
```

Matome, kad BC algoritmo rezultatas yra dvejopas – visų pirma jis grąžina ar pasiektas tikslas ar ne ir jei tikslas pasiektas jis grąžina produkcijų sąrašą – kelią. Klasikinio ir modifikuoti algoritmo įgyvendinimai sutampa, tačiau skiriasi iškvietimas – klasikinio atveju algoritmas yra kviečiamas vieną kartą, o modifikuotas atveju yra kviečiamas keletą kartų. Modifikuotas algoritmas sudėtingumu nusileidžia klasikiniam, kadangi vykdo daugiau iteracijų, tačiau

modifikuotas algoritmas palaiko kompleksinius scenarijus – kai norima išgauti daugiau nei vieną tikslą.

Modifikuotas FC algoritmas nuo klasikinio FC algoritmo skiriasi pačiu įgyvendinimu. Modifikavome klasikinį FC algoritmą ir gavome FC2 algoritmą. FC2 algoritmo realizacija yra sudėtingesnė. Modifikacija pavyko, kadangi klasikinis FC algoritmas vykdo paiešką į plotį. FC2 algoritmo pseudokodas pateiktas Priede Nr. 3. Palyginimui klasikinio ir modifikuoto FC algoritmo iškvietimai pateikti lentelėje Nr. 3 ir Nr. 4 atitinkamai.

3 lentelė. Klasikinio tiesioginio išvedimo algoritmo iškvietimo pseudokodo pavyzdys

```
char tikslas = 'A';
bool rezultatas = FC(tikslas);

if (rezultatas == true)
        {spausdiname kelią – produkcijų
sąrašą}
```

4 lentelė. Modifikuoto tiesioginio išvedimo algoritmo iškvietimo pseudokodo pavyzdys

```
char[] tikslai = ['A', 'B', 'C'];
bool rezultatas = FC2(tikslai);

if  (rezultatas == true)
        {spausdiname kelią – produkcijų
sąrašą}
```

Matome, kad FC algoritmo rezultatas yra dvejopas – visų pirma jis grąžina ar pasiektas tikslas ar ne ir jei tikslas pasiektas jis grąžina produkcijų sąrašą – kelią. Modifikuotas algoritmas sudėtingumu nesiskiria nuo klasikinio, tačiau modifikuotas algoritmas palaiko kompleksinius scenarijus – kai norima išgauti daugiau nei vieną tikslą.

3.3.2. Produkcijų vykdymas lygiagrečiai

Klasikiniai FC/BC algoritmai grąžiną nuoseklų planą, t. y. visos produkcijos turi būti vykdomos paeiliui. Web servisų kontekste pasitaiko atvejų, kai keli web servisai gali būti vykdomi lygiagrečiai. Todėl klasikiniai FC/BC algoritmai buvo modifikuoti, jog grąžintą produkcijų seką sugrupuotų į grupes pagal jų vykdymą – produkcijos, kurios gali būti vykdomos lygiagrečiai priklauso tai pačiai grupei. Šios modifikacijos įgyvendinimas C# programavimo kalba pateiktas Priede Nr. 4.

3.3.3. Perteklinių produkcijų pašalinimas

Klasikiniai FC/BC algoritmų surastame kelyje kartais atsiranda perteklinių – nereikalingų produkcijų. FC algoritmo pertekliškumas yra dėl pačio algoritmo prigimties. FC algoritmas vykdo paiešką į gylį. Paieškos į gylį metu suradus tikslą paieška stabdoma ir grąžinamas visas paieškos

medis nuoseklaus kelio pavidalu. Norint išvalyti kelią nuo perteklinių produkcijų reikalingas papildomas žingsnis – algoritmas, kuris pereina kelią iš naujo nuo tikslo ir surenka tik reikalingas produkcijas. Klasikinio BC algoritmo pertekliškumas turi kitokią prigimtį. BC algoritme perteklinės produkcijos atsiranda dėl nepasiekiamų tikslų arba grįžimo atgal (angl. backtracking). FC/BC pertekliškumo problema ir jos sprendimo būdai plačiau aprašyti [Cyr21]. Web servisų kontekste perteklinė produkcija gali ženkliai sulėtinti kompozicijos vykdymą, todėl norint to išvengti FC/BC algoritmai buvo modifikuoti, jog grąžintų kelią be perteklinių produkcijų.

4. PASLAUGOS KOKYBĖS SAVYBĖS

Paslaugos kokybės savybės (angl. quality of service) yra nefunkcinės web serviso savybės, tokios kaip atsakymo laikas (angl. response time), pralaidumas (angl. throughput), patikimumas (angl. reputation) ir t.t. Paslaugos kokybės savybės skirstomos į dvi grupes. Pirmajai grupei priklauso tos savybės, kurių reikšmėms augant, bendra web serviso kokybė mažėja. Tokių savybių pavyzdžiai: atsakymo laikas, kaina (angl. cost). Tokios savybės vadinamos neigiamomis paslaugos kokybės savybėmis. Antrajai grupei priklauso tos savybės, kurių reikšmėms augant, bendra web serviso kokybė irgi auga. Tokių savybių pavyzdžiai yra: pralaidumas, patikimumas. Tokios savybės vadinamos teigiamomis paslaugos kokybės savybėmis.

4.1. Bendros paslaugos kokybės savybės

Visoms apibrėžtoms paslaugų tiekėjų grupėms (žr. 2. Einamasis pavyzdys) apibrėšime bendras jų paslaugos kokybės savybes, kurias naudosime jų paslaugos kokybei vertinti. Kiekviena savybė turės svorį (angl. weight) procentinėje skalėje, kuris nulems jos svarbą vertinant bendrą paslaugos kokybę. Bendras paslaugos kokybes paimsime tokias pačias kaip darbe [BBH+17]:

- Atsakymo laikas (angl. response time) su svoriu 0,9 (90%). Ši savybė yra neigiama.
- Įvykdymo kaina, toliau - kaina (angl. cost) su svoriu 0,5 (50%). Ši savybė yra neigiama.
- Prieinamumas (angl. availability) su svoriu 1 (100%). Ši savybė yra teigiama.
- Patikimumas (angl. reputation) su svoriu 0,7 (70%). Ši savybė yra teigiama.
- Pasisekimas (angl. success rate) su svoriu 0,9 (90%). Ši savybė yra teigiama.

Kiekviena iš aukščiau pristatytų savybių laikui bėgant kinta, todėl yra apskaičiuojamos naudojant taisykles. Dalį taisyklių imsime iš [ZBN+04]:

- **Kaina.** Paslaugos tiekėjai kainą nusistato statinę arba leidžia paslaugos naudotojams gauti kainą dinamiškai, pvz., kreipiantis per API.
- **Patikimumas.** Paslaugos patikimumas priklauso nuo to kaip paslaugos naudotojai jį įvertina. Patikimumui vertinti dažniausiai priskiriamas rėžis, kuriame paslaugos naudotojai įvertina paslaugą. Pavyzdžiui, Amazon.com rėžis yra [0, 5]. Turint daug naudotojų įvertinimų paslaugos patikimumas apskaičiuojamas susumuojant visų naudotojų įvertinimus ir padalinus sumą iš įvertinimų skaičiaus, t. y. imant visų įvertinimų vidurkį.
- **Prieinamumas.** Paslaugos prieinamumas yra tikimybė, kai bet kuriuo momentu kreipiantis į paslaugą ji bus prieinama. Prieinamumas apibrėžiamas kaip laikas, kurį paslauga yra prieinama stebint ją tam tikrą laiko tarpą (konstanta). Pavyzdžiui,

stebime tam tikrą web servisą 60s, iš kurių jis prieinamas 40s, tada web serviso prieinamumas yra 40/60 ~ 0,67. Tai reiškia, kad web servisas vidutiniškai yra prieinamas 67% laiko. Web servisų kontekste prieinamumas dažniausiai yra stebimas pastoviai, naudojant prieinamumo stebėjimo sistemas (angl. availability monitoring systems). Turint daug prieinamumo duomenų per tam tikrą laiko tarpą, norint gauti web serviso prieinamumą imamas visų duomenų vidurkis.

- **Atsakymo laikas.** Paslaugos atsakymo laikas apskaičiuojamas pasinaudojant istoriniais duomenimis. Kiekvieną kartą iškviečiant paslaugą yra saugomas atsakymo laikas. Paslaugos atsakymo laikas yra dalies jos istorinių atsakymo laikų vidurkis. Vidurkis skaičiuojamas tik apsibrėžtos dalies istorinių duomenų, o ne visų, kadangi laikui bėgant paslauga gali tobulėti arba blogėti ir skaičiuojant tam tikro laikotarpio istorinių duomenų vidurkį yra gaunami tikslesni rezultatai. Kuo daugiau turima duomenų, tuo tikslesnį paslaugos atsakymo laiką galima apskaičiuoti.
- **Pasisekimas.** Paslaugos pasisekimas apibūdinamas tikimybe, kad užklausa bus sėkmingai įvykdyta, t. y. bus gautas sėkmę rodantis atsakymas (web servisų atveju 2xx HTTP kodas) bei netuščias atsakas. Paslaugos pasisekimas yra apskaičiuojamas pasinaudojant istoriniais duomenimis. Kiekvieną kartą iškviečiant paslaugą yra saugoma, ar buvo gautas sėkmingas netuščias atsakas. Paslaugos pasisekimo reitingas yra dalies jos istorinių duomenų vidurkis. Tikslesni rezultatai gaunami skaičiuojant tik tam tikro laikotarpio istorinių duomenų vidurkį, kadangi laikui bėgant paslauga gali tobulėti arba prastėti. Kuo daugiau turima duomenų, tuo tikslesnį paslaugos pasisekimo rodiklį galima apskaičiuoti.

4.2. Paslaugos kokybės savybės, specifinės web servisų grupei

Kiekvienai paslaugų tiekėjų grupei (žr. 2. Einamasis pavyzdys) apibrėšime specifines paslaugos kokybės savybes vertinti paslaugos kokybę. Specifinės savybės parinktos pagal kiekvienos grupės dalykinę sritį. Kiekviena savybė turi procentinį svorį (angl. weight), nulemiantį jos svarbą vertinant bendrą paslaugos kokybę. Visos toliau pateikiamos savybės yra statinės, tai yra pateikiamos web serviso pridėjimo prie bendro sąrašo metu. Šios savybės atspindi paslaugos patikimumą ir kokybę dalykinėje srityje. Visos grupės specifinės savybės yra teigiamos.

- **Automobilių paieškos paslauga**
  - **Rankiniu būdu patvirtinamas kiekvienas skelbimas** – rodiklis nurodyti, ar platformoje įkeliamus skelbimus privalo peržiūrėti ir patvirtinti žmogus (administratorius). Svoris yra 1 (100%).
  - **Audituojamumas** – ar platformą audituoja auditoriai. Svoris yra 0,8 (80%).
  - **Apsauga nuo sukčiavimo** – ar veikia kokia nors apsaugos nuo sukčiavimo sistema. Svoris 1 (100%).
  - **Pardavėjų stebėjimas ir patvirtinimas** – ar skelbimus keliantys pardavėjai yra rankiniu būdu patvirtinami ir nuolat stebimi. Svoris 0,7 (70%).
- **Automobilių istorijos paslauga**
  - **Šalių skaičius, kurių registrai yra tikrinami** – kelių šalių registre yra tikrinama automobilio istorija. Šalių skaičius čia yra svarbus, kadangi tikrinant ar automobilis yra nevogtas, draudimo nenurašytas ir pan. yra svarbu patikrinti kiek įmanoma daugiau šalių. Svoris 0,7 (70%).
  - **Bendradarbiaujama su valstybinėmis institucijomis** – ar platforma tikrindama automobilio istoriją bendradarbiauja su valstybinėmis institucijomis, pvz., policija ar muitine. Svoris 1 (100%).
  - **Garantijos suteikimas** – ar platforma suteikia garantiją sugeneruotoms automobilio istorijos ataskaitoms. Ši savybė suteikia pasitikėjimo paslaugos naudotojui. Svoris 0,5 (50%).
  - **Pateikiamos senesnių skelbimų automobilio nuotraukos** – ar platforma tikrindama automobilio istoriją ieško ir praeitų automobilio pardavimo skelbimų bei juos radusi pateikia nuotraukas. Ši savybė yra svarbi perkant automobilį iš Amerikos, nes ten dažnai jie būna daužti ir nurašyti, o kiek stipriai daužti sunku pasakyti iš aprašymo. Todėl nuotraukos yra svarbios apsisprendžiant pirkti. Svoris yra 0,4 (40%).

- **Automobilių istorijos paslauga**
    - **Šalių skaičius, kurių registrai yra tikrinami** – kelių šalių registre yra tikrinama automobilio istorija. Šalių skaičius čia yra svarbus, kadangi tikrinant ar automobilis yra nevogtas, draudimo nenurašytas ir pan. yra svarbu patikrinti kiek įmanoma daugiau šalių. Svoris 0,7 (70%).
    - **Bendradarbiaujama su valstybinėmis institucijomis** – ar platforma tikrindama automobilio istoriją bendradarbiauja su valstybinėmis institucijomis, pvz., policija ar muitine. Svoris 1 (100%).
    - **Garantijos suteikimas** – ar platforma suteikia garantiją sugeneruotoms automobilio istorijos ataskaitoms. Ši savybė suteikia pasitikėjimo paslaugos naudotojui. Svoris 0,5 (50%).
    - **Pateikiamos senesnių skelbimų automobilio nuotraukos** – ar platforma tikrindama automobilio istoriją ieško ir praeitų automobilio pardavimo skelbimų bei juos radusi pateikia nuotraukas. Ši savybė yra svarbi perkant automobilį iš Amerikos, nes ten dažnai jie būna daužti ir nurašyti, o kiek stipriai daužti sunku pasakyti iš aprašymo. Todėl nuotraukos yra svarbios apsisprendžiant pirkti. Svoris yra 0,4 (40%).
- **Draudimo paslauga**
    - **Kasko draudimas** – ar draudimo bendrovė gali suteikti kasko draudimą. Ši savybė mažiau įtakoja paslaugos kokybę, tačiau dažnai bendrovėmis, kurios nesuteikia kasko, žmonės mažiau pasitiki. Svoris 0,5 (50%).
    - **Registruota Lietuvoje** – ar draudimo bendrovė teikianti šią paslaugą registruota Lietuvoje. Svoris 1 (100%).
    - **Metai rinkoje** – kiek metų draudimo bendrovė teikia draudimo paslaugas. Svoris 1 (100%).

4.3. Paslaugos kokybės balo apskaičiavimas

Paslauga turi dviejų tipų kokybės savybes: bendras (žr. 4.1. Bendros paslaugos kokybės) bei paslaugos grupei specifines (žr. 4.2. Paslaugos kokybės savybės, specifinės web servisų grupei). Paslaugos kokybės balas apskaičiuojamas taip:

1. **Normalizavimas.** Kokybės savybė gali būti įvertintos tam tikroje skalėje, pvz., patikimumo reikšmės skalė yra [1, 10]. Kai kurių savybių, tokių kaip kaina ar atsakymo laikas, skales sunku įvertinti, todėl jiems suteikėme maksimalią 1000 įverčio ribą tyrimo tikslais. Kiekvienas savybės įvertis yra normalizuojama iki skalės [1, 10] pagal formulę |a / 1000 * 10|, kur a yra normalizuojamas skaičius. Jeigu savybės įvertis yra taip arba ne, ji paverčiama į 10 arba 0 atitinkamai.
2. **Sumavimas.** Sunormalizavus savybių įverčius apskaičiuojamas paslaugos kokybės balas. Tai padaroma kiekvieną sunormalizuotą savybės įvertį padauginus iš jos svorio ir visas tokias sandaugas susumavus. Pavyzdžiui, web servisas A turi tokius kokybės savybių įverčius: atsakymo laikas – 100 ms, kaina – 500, pasiekiamumas – 9, patikimumas – 10, pasisekimas – 8. Sunormalizuoti savybių įverčiai: atsakymo laikas – 1, kaina – 5, pasiekiamumas – 9, patikimumas – 10, pasisekimas – 8. Sudauginus savybės įverčius iš savybės svorio gauname tokias sandaugas: 0,9; 2,5; 9; 7; 7,2. Susumavus šias sandaugas gauname paslaugos kokybės balą: 0,9 + 2,5 + 9 + 7 + 7,2 = 26,6. Šiame pavyzdyje yra tik bendros visoms grupėms paslaugos kokybės savybės. Jeigu būtų specifinių grupei savybių, jos irgi būtų susumuotos.

Kalbant apie servisų kompoziciją neužtenka žinoti kiekvienos paslaugos kokybės balo, bet svarbus visos kompozicijos kokybės balas. Kompozicijos balas yra visų joje dalyvaujančių paslaugų kokybės balų suma.

4.4. Web servisų rikiavimas pagal paslaugos kokybės savybes

Web servisų kvietimo planą pagamina FC/BC algoritmai. Vėliau sudaromos visos įmanomas kompozicijas su konkrečiais web servisais. Toliau kiekvienai kompozicijai apskaičiuojamas kokybės balas. Galiausiai kompozicijos išrikiuojamos mažėjimo tvarka pagal jų balą. Pirmasis elementas išrikiuotame sąraše bus kokybiškiausia ir geriausiai vartotojo užklausą atitinkanti kompozicija. Ši kompozicija yra parenkama vykdymui.

5. DEMONSTRACINĖ SISTEMA

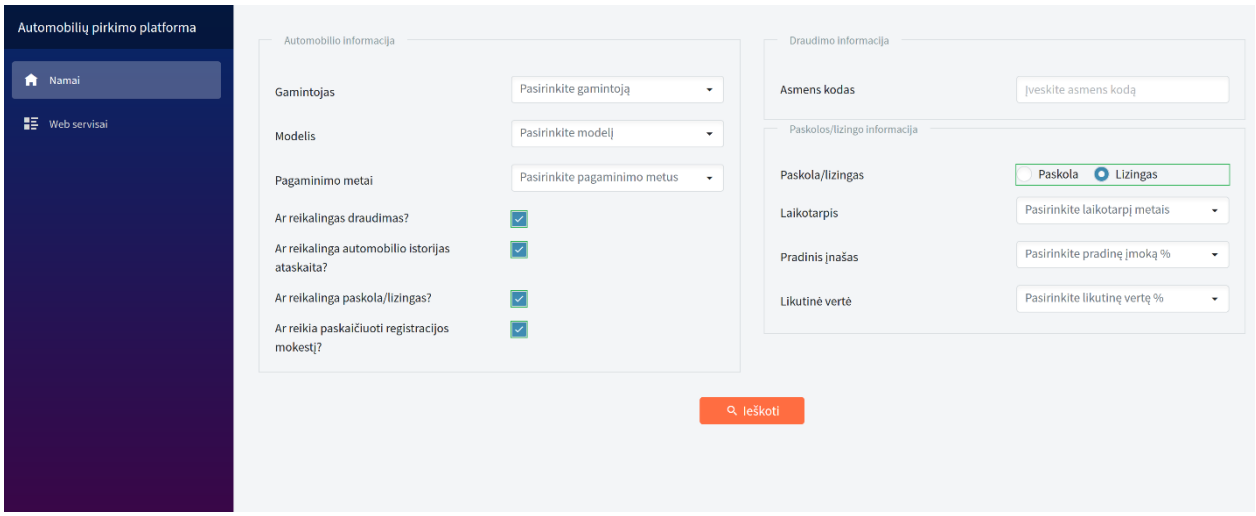
Demonstracinė sistema iliustruoja einamąjį pavyzdį (žr. 2. Einamasis pavyzdys) ir įgyvendina aprašytus algoritmus (žr. 3. FC/BC algoritmai ir 4. Paslaugos kokybės savybės). Toliau aprašomos demonstracinės sistemos įgyvendinimo ypatybės bei įgyvendinti scenarijai. Demonstracinės sistemos programa patalpinta GitHub talpykloje https://github.com/JonasPetraska/CarPurchasePlatform.

5.1. Technologijos ir architektūra

Demonstracinę sistemą sudaro išorinė (angl. frontend) ir vidinė (angl. backend) dalys. Išorinė dalis, tai vartotojo sąsaja, kurioje vartotojas pateikia užklausą. Vidinė dalis, tai algoritmas, kuris pateiktą užklausą apdoroja ir kreipiasi į web servisus. Taip pat, vidinę dalį sudaro pavyzdiniai web servisai, kurie dalyvauja kompozicijoje. Išorinė ir vidinė sistemos dalys suprogramuotos „Visual Studio“ aplinkoje naudojant C# programavimo kalbą. Išeities kodo valdymui ir talpinimui panaudota „Git“ programinė įranga ir „GitHub“ talpykla.

5.1.1. Išorinė sistemos dalis

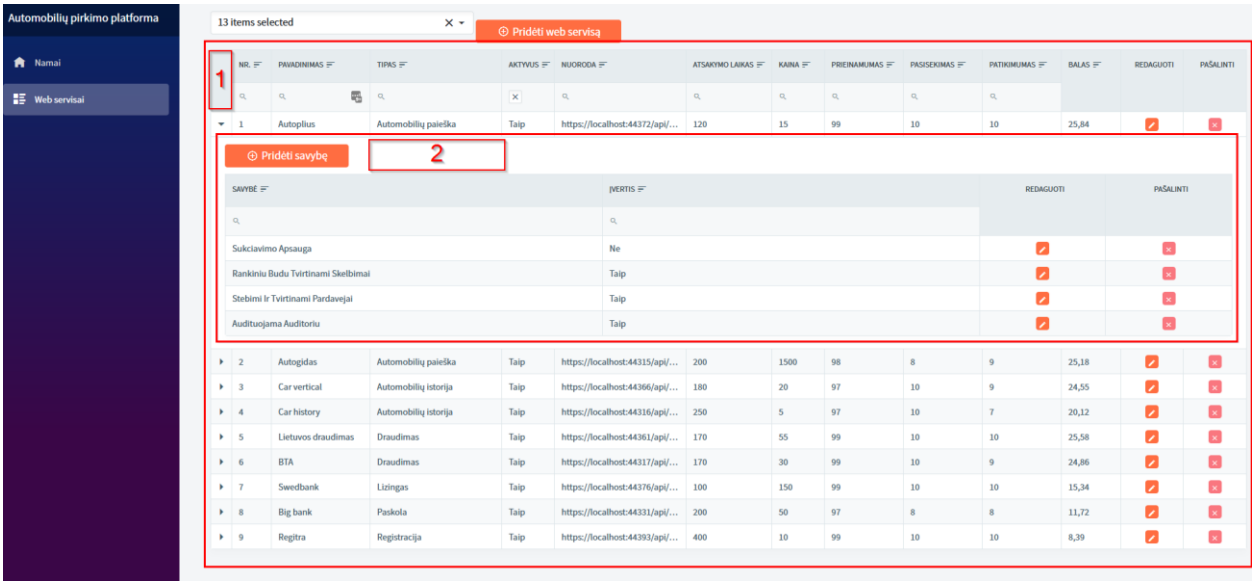
Išorinėje sistemos dalyje vartotojas pasirenka užklausos kriterijus ir mato užklausos rezultatus. Ši dalis sukurta su nauju Microsoft karkasu „Blazor“. Vartotojo sąsaja suprogramuota HTML žymenų bei CSS kalbomis. Siekiant pagreitinti demonstracinės sistemos kūrimo procesą buvo panaudota vartotojo sąsajos elementų biblioteka „Radzen“. Išorinę sistemos dalį sudaro du langai, pirmas, paieškos langas ir, antras, web servisų sąrašo administravimo langas.



1 pav. Demonstracinės sistemos paieškos langas

Pirmajame lange, žr. 1 pav., vartotojas įveda paieškos parametrus, pvz., automobilio gamintoją, modelį ir pagaminimo metus. Vartotojas pažymėdamas varneles pasirenka jam reikalingas paslaugas, pavyzdžiui, draudimo, finansavimo paslaugą ir pan. Pažymėjus varneles jo

paprašoma įvesti papildomą informaciją (asmens kodą, lizingo laikotarpį, pradinį įnašą bei likutinę vertę, paskolos maksimalią metinę palūkanų normą). Šiame lange pateiktą informaciją FC/BC algoritmai naudoja kaip faktus sudarydami kelią. Įvairių paslaugų pasirinkimas įtakoja sudaromą kelią.



2 pav. Demonstracinės sistemos web servisų sąrašo administravimo langas

Antras langas 2 pav. yra skirtas web servisų sąrašo administravimui. Demonstraciniais tikslais šis langas matomas visiems vartotojams, tačiau realioje sistemoje jis turėtų būti matomas tik sistemos administratoriams. Šiame lange matomi visi sistemoje esantys web servisai. Paveiksle pažymėtoje srityje Nr. 1 matomi bendrieji web serviso parametrai: pavadinimas, grupė, API nuoroda, bendrųjų paslaugos kokybės savybių įverčiai bei paskaičiuotas kokybės balas. Srityje Nr. 2 matomi tik web serviso grupei būdingos paslaugos kokybės savybės ir jų įverčiai. Web servisus ir jų kokybės savybes galima pridėti, redaguoti bei šalinti.

5.1.2. Vidinė sistemos dalis

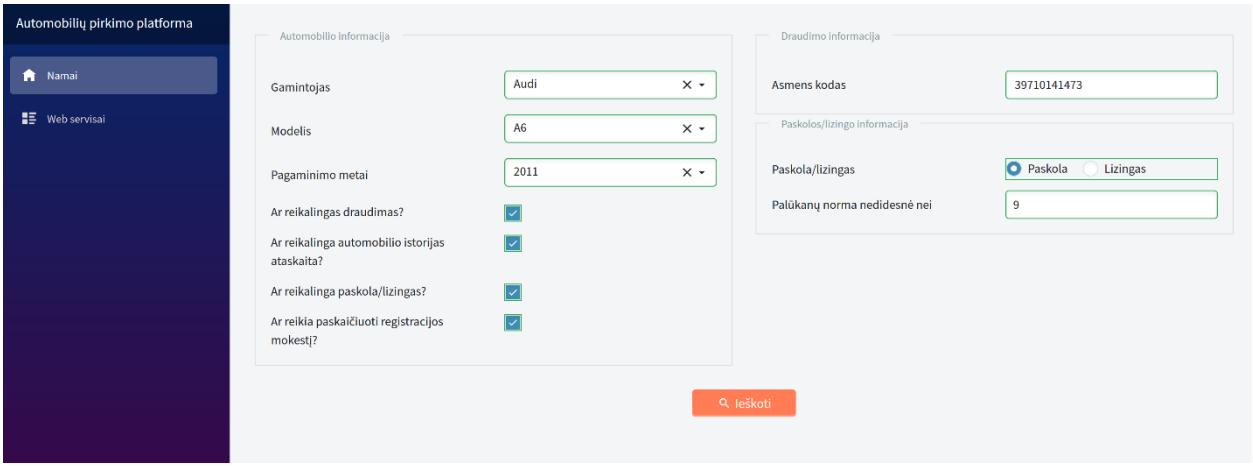
Vidinė sistemos dalis susideda iš, pirma, pavyzdinių web servisų ir, antra, išorinėje dalyje įgyvendintos logikos. Web servisai realizuoti .NET Core karkasu. Web servisuose sukurtos API sąsajos (angl. interface). Demonstraciniais tikslais užklausų vykdymo logika ir skaičiavimai pavyzdiniuose web servisuose yra imituojami. Duomenys saugomi JSON dokumentuose arba atmintyje (angl. in-memory). Siekiant supaprastinti demonstracinės sistemos įgyvendinimą nenaudojome reliacinių duomenų bazių.

5.2. Įgyvendinti scenarijai

Einamajam pavyzdžiui (žr. 2. Einamasis pavyzdys) iliustruoti demonstracinėje sistemoje įgyvendinti įvairūs scenarijai. Pagrindiniai trys (standartinis, web serviso kokybės balo kitimo, vartotojo pasirinkimų kitimo) aptariami žemiau.

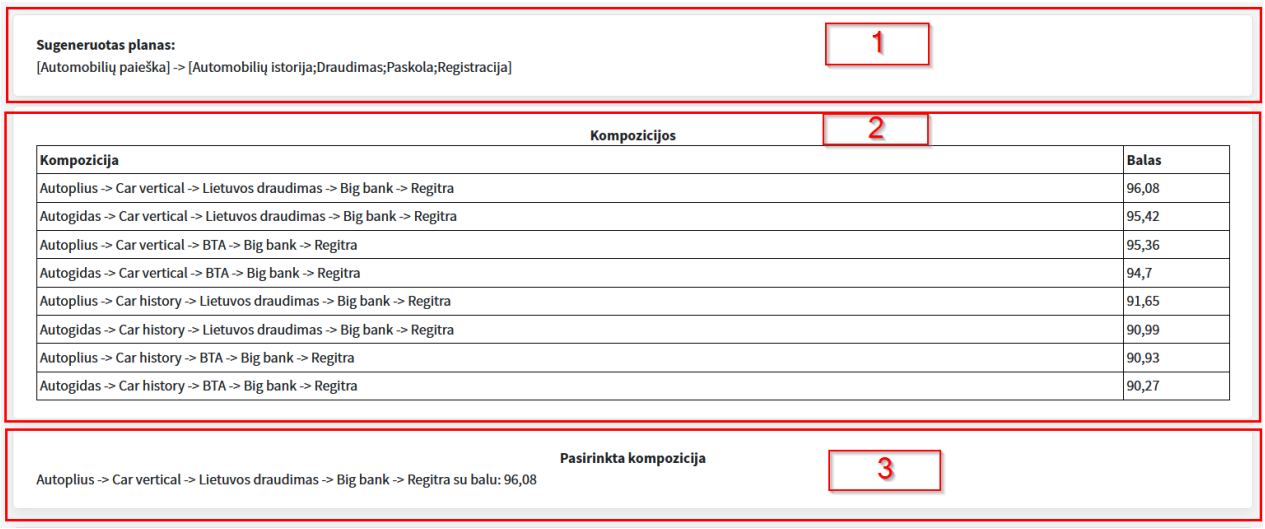
5.2.1. Standartinis scenarijus

Standartiniu scenarijumi vadinamas einamojo pavyzdžio konkretizavimas parametrais. Tegu, Jonas nori pirkti naudotą automobilį Audi A6 2011 metų. Jonas nori gauti automobilio draudimo pasiūlymą, sužinoti automobilio istoriją, gauti paskolos pasiūlymą bei sužinoti taršos mokestį. Jonas pageidauja, jog metinė palūkanų norma būtų iki 9%. Jonui pažymėjus, jog nori gauti draudimo pasiūlymą, sistema paprašo įvesti asmens kodą. Langas su Jono užpildytais užklausos duomenimis parodytas 3 pav.



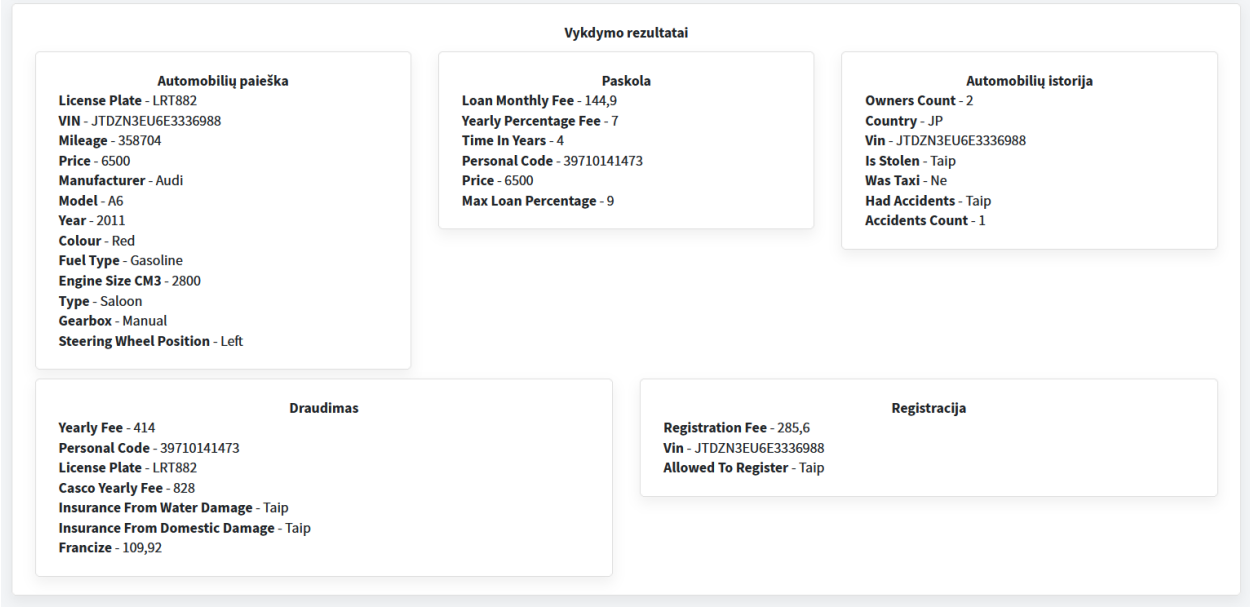
3 pav. Jono užklausos duomenimis užpildytas demonstracinės sistemos langas

Jonui paspaudus mygtuką „Ieškoti“ sistema vykdo paiešką. Paieškos rezultatai parodyti 4 pav. ir 5 pav. (vientisą langą žr. Priede Nr. 5). Demonstraciniais tikslais vartotojas mato ir tarpinius žingsnius, žr. 4 pav. Realioje sistemoje vartotojas matytų tik galutinį rezultatą, žr. 5 pav.



4 pav. Paieškos tarpiniai žingsniai

Pirmiausiai sistema siunčia iš vartotojo gautus duomenis FC/BC algoritmui, kuris sugeneruoja bendrąjį kelią pagal paslaugų grupes, žr. 4 pav. 1-oje srityje, kur kelias atskirtas rodyklėmis „->“, o lygiagrečiai vykdomos produkcijos sugrupuotos naudojant laužtinius skliaustus „[“ ir „]“. Toliau sistema sugeneruoja visas galimas kompozicijas su konkrečiais web servisais ir suskaičiuoja tų kompozicijų balus, žr. 4 pav. 2-oje srityje. Galiausiai sistema parenka kokybiškiausią kompoziciją pagal didžiausią balą; žr. 4 pav. 3-ioje srityje.



5 pav. Kompozicijos vykdymo rezultatas

Kompozicijos vykdymo rezultatas parodytas 5 pav. Sistema kompozicijos produkcijas (web servisus) kviečia pagal sudarytą planą. Web servisai sugrupuoti laužtiniais skliaustais kviečiami lygiagrečiai. Atsakymai, gauti iš atskirų web servisų, pateikiami vartotojui.

5.2.2. Web serviso kokybės balo kitimo scenarijus

Web servisų sistemose servisų kokybės savybės laikui bėgant kinta. Todėl svarbu, kad planuodama kelią sistema atsižvelgtų į pokyčius. Demonstracinėje sistemoje visi kokybės įverčiai yra suvedami ranka, realioje sistemoje dalis jų būtų skaičiuojami pagal taisykles (žr. 4. Paslaugos kokybės savybės). Tegu „Autoplius“ web serviso atsakymo laikas pailgėja, nes jų serveriai tapo lėtesni. Tegu, web serviso „Lietuvos draudimas“ patikimumas nukrito dėl neseniai įvykusios rezonansinės sukčiavimo bylos. Pakeiskime „Autoplius“ web serviso atsakymo laiką iš 120 ms į 300 ms ir web serviso „Lietuvos draudimas“ patikimumą iš 10 į 7. Pasikeitusių web servisų sąrašas parodytas 6 pav. Pakeitę šių kokybės savybių įverčių reikšmes, vykdome pakartotinę paiešką su tomis pačiomis reikšmėmis.

NR.	PAVADINIMAS	TIPAS	AKTYVUS	ATSAKYMO LAIKAS	KAINA	PRIEINAMUMAS	PASISEKIMAS	PATIKIMUMAS	BALAS	REDAGUOTI	PAŠALINTI
1	Autoplius	Automobilių paieška	Taip	300	15	99	10	10	24,22		
2	Autogidas	Automobilių paieška	Taip	200	1500	98	8	9	25,18		
3	Car vertical	Automobilių istorija	Taip	180	20	97	10	9	24,55		
4	Car history	Automobilių istorija	Taip	250	5	97	10	7	20,12		
5	Lietuvos draudimas	Draudimas	Taip	170	55	99	10	7	23,48		
6	BTA	Draudimas	Taip	170	30	99	10	9	24,86		
7	Swedbank	Lizingas	Taip	100	150	99	10	10	15,34		
8	Big bank	Paskola	Taip	200	50	97	8	8	11,72		
9	Regitra	Registracija	Taip	400	10	99	10	10	8,39		

6 pav. Web servisų sąrašas pakeitus „Autoplius“ ir „Lietuvos draudimas“ web servisų kokybės savybių įverčius

Pakartotinės kompozicijos rezultatai parodyti 7 pav. Po pakeitimų standartiniame scenarijuje (žr. 5.2.1. Standartinis scenarijus) parinktos kompozicijos (žr. 4 pav. 3-ia sritis) balas nukrito nuo 96,08 iki 92,36. Todėl šį kartą pasirenkama kita kompozicija su didesniu balu. Pakeitus „Autoplius“ ir „Lietuvos draudimas“ web servisų kokybės įverčius, naujai sistemos pagamintoje kompozicijoje juos pakeitė konkurentai „Autogidas“ ir „BTA“.

Sugeneruotas planas:
[Automobilių paieška] -> [Automobilių istorija;Draudimas;Paskola;Registracija]

1

Kompozicijos

2

Kompozicija	Balas
Autogidas -> Car vertical -> BTA -> Big bank -> Regitra	94,7
Autoplius -> Car vertical -> BTA -> Big bank -> Regitra	93,74
Autogidas -> Car vertical -> Lietuvos draudimas -> Big bank -> Regitra	93,32
Autoplius -> Car vertical -> Lietuvos draudimas -> Big bank -> Regitra	92,36
Autogidas -> Car history -> BTA -> Big bank -> Regitra	90,27
Autoplius -> Car history -> BTA -> Big bank -> Regitra	89,31
Autogidas -> Car history -> Lietuvos draudimas -> Big bank -> Regitra	88,89
Autoplius -> Car history -> Lietuvos draudimas -> Big bank -> Regitra	87,93

Pasirinkta kompozicija

3

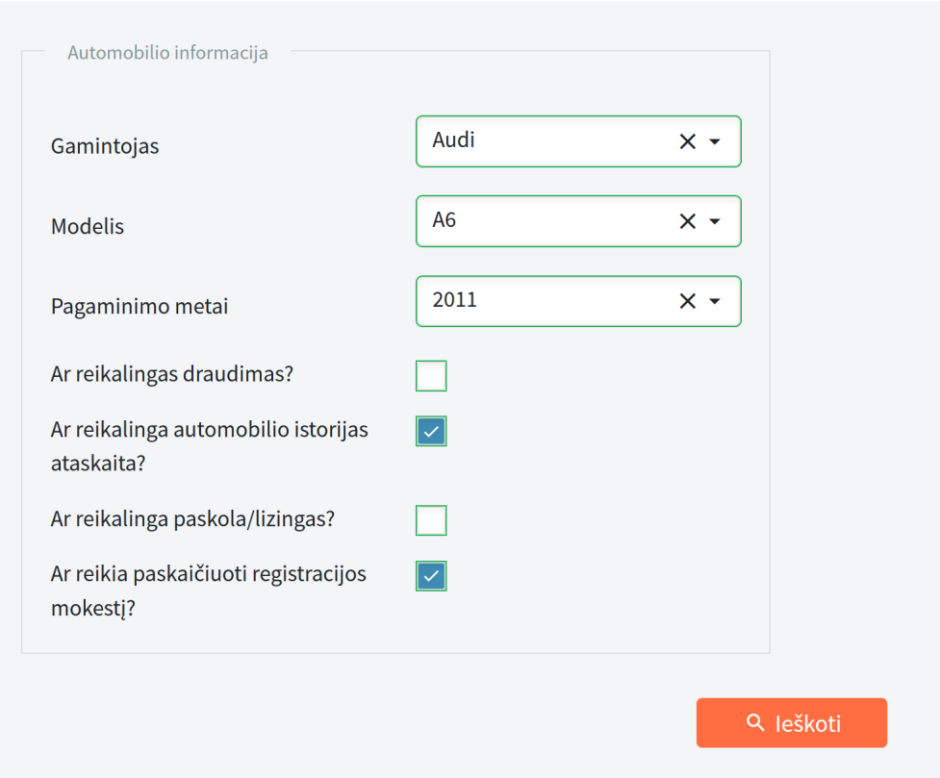
Autogidas -> Car vertical -> BTA -> Big bank -> Regitra su balu: 94,7

7 pav. Kompozicijos rezultatas pakeitus dalies web servisų kokybės savybių įverčius

Realioje sistemoje šie kokybės savybių pokyčiai vyktų nepastebimai ir balai būtų perskaičiuojami automatiškai, todėl vartotojai nepastebėtų trikdžių ir gautų kokybiškiausią paslaugą.

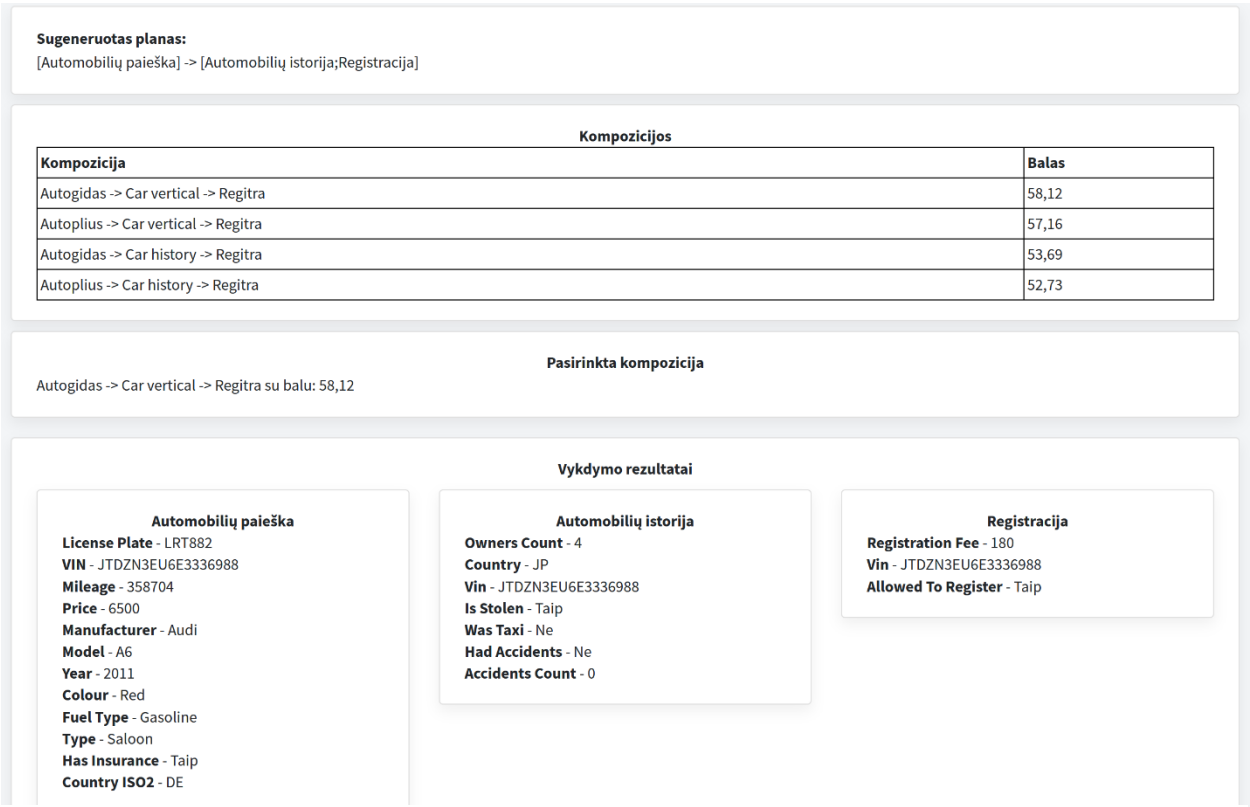
5.2.3. Vartotojo pasirinkimų kitimo scenarijus

Tegu Jonas nebenori gauti nei draudimo pasiūlymo, nei paskolos pasiūlymo, nes apsisprendė automobilį pirkti mokėdamas iš karto, o draudimą jam padarys pusseserė brokerė. Tokiu atveju Jonas nuima varneles nuo „Ar reikalingas draudimas?“ ir „Ar reikalinga paskola/lizingas?“. Naujai nurodyti paieškos kriterijai parodyti 8 pav. Jonui nuėmus varneles, sistema nebeprašo įvesti asmens kodo ir nurodyti paskolos sąlygas.



8 pav. Jono naujai nurodyti paieškos kriterijai

Pakartotinai vykdome paiešką, jos rezultatai parodyti 9 pav. Iš sugeneruoto plano dingo draudimo ir paskolos paslaugos, o kompozicijų sąrašas sutrumpėjo. Iš web servisų grąžinti atsakymai yra trys – lygiai tiek kiek paslaugų Jonas ir pasirinko.



9 pav. Paieškos, pakeitus paslaugų kiekį, rezultatai

5.3. Galimi išplėtimai ir panaudojimas

Demonstracinė sistema yra supaprastina iki apibrėžtų scenarijų, todėl norint panašią sistemą pristatyti kaip produktą ją reikėtų išplėsti. Galimi patobulinimai yra šie:

1. Demonstracinė sistema nesugeba apsvarstyti alternatyvių web servisų, jeigu pagrindinis web servisas negrąžina ieškomo rezultato. Pavyzdžiui, jeigu „Autoplius“ web servisas neranda ieškomo automobilio, tai sistema interpretuoja, kad automobilis iš viso nerastas. Tačiau gali būti, kad web servisas „Autogidas“ suras. Todėl sistema nesėkmės atveju galėtų apsvarstyti ir iškviesti alternatyvius web servisus.
2. Bendrieji paslaugos kokybės parametrai, tokie kaip atsakymo laikas, patikimumas, prieinamumas ir pan., į sistemą yra suvedami ranka ir todėl yra statiški. Išplėtimui būtinos stebėjimo ir informacijos rinkimo sistemos, kurios nuolatos stebėtų visus web servisus, rinktų apie juos duomenis ir pagal formules (žr. 4.1. Bendros paslaugos kokybės savybės) perskaičiuotų paslaugos kokybės balą.
3. Planavimas sistemoje vyksta kiekvieną kartą iškvietus paiešką, tačiau dažnai planai kartojasi. Norint patobulinti planavimo algoritmą, sugeneruoti planai galėtų būti išsaugomi ir naudojami pakartotinai.
4. Web servisų grupių, tokių kaip automobilių paieška, automobilių istorija ir pan., semantinės (įeities–išeities) schemos saugomos atmintyje. Keičiant schemą būtina keisti programos išeities kodą. Šiai modifikacijai semantines schemas tektų saugoti duomenų talpykloje ir redaguoti per vartotojo sąsają.
5. Demonstracinė sistema palaiko tik HTTP GET metodo užklausas. Tektų išplėsti sistemą, kad aprašant web servisą būtų nurodomas HTTP metodas.
6. Duomenys vidinėje demonstracinės sistemos dalyje yra saugomi atmintyje arba JSON rinkmena. Duomenys būtų saugomi reliacinėje duomenų bazėje, pavyzdžiui MS SQL.
7. Sistema kreipdamasi į web servisus tikisi atsakymo iš vieno elemento, o ne elementų masyvo. Pavyzdžiui, automobilių paieškos atveju sistema tikisi, jog bus grąžintas vienas automobilis. Realybėje web servisai gali grąžinti daug alternatyvų, pavyzdžiui, automobilių paieška gali grąžinti daugiau nei vieną kriterijus atitinkantį automobilį. Todėl reikėtų modifikuoti sistemą, jog būtų palaikomi atsakymai masyvai.
8. Sistema kreipdamasi į web servisus tikisi, jog atsakymą sudarys tik primityvūs objektai, pavyzdžiui, tekstas, skaičius ar raidė. Reikėtų modifikuoti sistemą, kad būtų

palaikomi kompleksiniai objektai ir sistema galėtų juos interpretuoti gavusi atsakymą iš web serviso.

9. Sistema nepalaiko kompleksinių scenarijų, kai į tą patį paslaugos teikėją kreiptųsi kelis kartus. Tokiems scenarijams, reikėtų modifikuoti web servisų aprašymus. Galimas būdas jiems aprašyti yra blokinė diagrama (angl. flowchart), kurią vykdymo metu sistema interpretuoja.

Mūsų pateiktame automobilių pirkimo pavyzdyje yra sujungiamos įvairios su automobiliu susijusios paslaugos, tokios kaip draudimas, finansavimas, istorija, registracija. Jo apjungiamos į paslaugų komplektus (angl. bundles) pasinaudojant web servisų kompozicija. Šiais laikais kiekviena sistema turi savo API interfeisą, todėl jau nebėra sunku inicijuoti tarpsisteminę komunikaciją. Mes matome dar bent tris mūsų sistemos naudojimu atvejus ateityje. Jie būtų naudingi ateities tyrimams.

1. Nekilnojamo turto sektoriuje į web servisų kompoziciją galima būtų įjungti nekilnojamo turto paieškos, notaro, registrų centro ir finansavimo paslaugas.
2. Web servisų kompozicija gali būti naudojama ir fizinių įrenginių kompozicijai, kai įrenginiai dalį resursų leidžia naudoti kaip paslaugas. Pavyzdžiui, prekybos centre galėtų būti sukomponuoti įvairių mobiliųjų įrenginių resursai, tokie kaip GPS, žemėlapis, vertėjas ir pan. Tokiu atveju prekybos centro lankytojai galėtų naudotis paslaugomis, kurių jų asmeniniai išmanieji įrenginiai nepalaiko.
3. Turizmo sektoriuje web servisų kompoziciją galima panaudoti, kad apjungti skrydžių bendrovių, apgyvendinimo įstaigų, automobilių nuomos ar transportavimo, pramoginių paslaugų tiekėjų ir pan. paslaugas. Taip būtų sukurta platforma turistui dinamiškai pagal poreikius suplanuoti savo keliones ir gauti pasiūlymus.

6. MŪSŲ DARBO SANTYKIS SU BEKKOUCHE ET AL. (2017) TYRIMU

Didžiausią įtaką mūsų bakalauriniam darbui padarė Bekkouche et al. (2017) ankščiau atliktas tyrimas [BBH+17]. Bekkouche et al. tyrimas leido išplėsti pažinimą web servisų kompozicijos tema. Jų tyrime nagrinėtas pavyzdys buvo atspirties taškas mums galvojant apie savo einamąjį pavyzdį (žr. 2. Einamasis pavyzdys). Mūsų bakalauriniame darbe pristatomas algoritmas yra jų algoritmo modifikacija. Mes jų algoritmą pritaikėme konkrečiam automobilių pirkimo pavyzdžiui ir panaudojome FC/BC algoritmus. Nagrinėdami šį ir kitus tyrimus pastebėjome, kad nėra daug tyrimų, kurie tiria FC/BC šiuolaikinėse sistemose. Dauguma tyrimų pristato naujus algoritmus ar karkasus web servisų kompozicijai. Bekkouche et al. tyrimas įkvėpė mus ieškoti FC/BC vietos šiuolaikinėse web servisų sistemose. Jis parodė kryptį modifikuoti klasikinius FC/BC ir rasti savo vietą tyrimuose.

REZULTATAI

1. Identifikuotos trys priežastys, kodėl klasikiniai FC/BC algoritmai gali sunkiai tikti web servisų kompozicijai šiuolaikinėse web servisų sistemose. Pateikti du pavyzdžiai iliustruoti šias priežastis: pirmas atsižvelgia į web servisų kokybės savybes, o antras į – web servisų statiškumą.
2. Sumodeliuotas einamasis pavyzdys apie naudotų automobilių pirkimą. Sukurti trys pavyzdiniai scenarijai apie skirtingus web servisų kompozicijos atvejus.
3. Įgyvendintos trys klasikinių FC/BC algoritmų modifikacijos pavyzdiniams scenarijams pademonstruoti.
4. Sukurta demonstracinė sistema su web servisų pavyzdžiais. Užprogramuota C# kalba naudojant .NET Core karkasą.
5. Pateikti pasiūlymai panaudoti modifikuotus FC/BC algoritmus kitose dalykinėse srityse – nekilnojamojo turto prekyboje, prekybos centrų veikloje ir turizme.

IŠVADOS

1. Modifikavus klasikinius FC/BC algoritmus, jie tinka šiuolaikinėms web servisų sistemoms, kuriose vyksta dinaminis servisų komponavimas. Šitaip galima pasiekti panašius rezultatus į tuos, kurie pristatyti literatūroje apie web servisus, jų kompoziciją ir karkasus.
2. Pagrindiniai klasikinių FC/BC algoritmų trūkumai yra šie: pirma, greitai augantis, gali būti ir eksponentinis, kompozicijos planavimo laikas, kai web servisų skaičius didėja; antra, būtinybė atsižvelgti į web servisų kokybės savybes; trečia, būtinybė išvengti statiškumo ir atsižvelgti į dinamiškai kintančią web servisų aplinką.
3. Modifikuoti FC/BC algoritmai galėtų būti panaudoti įvairiose dalykinėse srityse: pirma, perkant automobilį komponuoti automobilių pirkimo, draudimo, finansavimo, registravimo ir pan. paslaugas; antra, nekilnojamo turto prekyboje komponuoti turto pirkimo, notaro, registrų centro, finansavimo ir pan. paslaugas; trečia, turizme komponuoti skrydžių, apgyvendinimo, transportavimo ir pan. paslaugas; ketvirta, komponuoti fizinių išmaniųjų įrenginių resursus, tokius kaip GPS, žemėlapiai ir pan. bei juos naudoti kaip servisus.

ŠALTINIAI

[AP13] G. H. Alférez, V. Pelechano. *Facing Uncertainty in Web Service Compositions*. 2013 IEEE 20th International Conference on Web Services, Santa Clara, CA, USA, 2013, pp. 219-226. https://doi.org/10.1109/ICWS.2013.38.

[BBH+17] A. Bekkouche, S. M. Benslimane, M. Huchard, C. Tibermacine, F. Hadjila, M. Merzoug. (2017). *QoS-Aware Optimal and Automated Semantic Web Service Composition With User's Constraints*. Service Oriented Computing and Applications. to appear. https://doi.org/10.1007/s11761-017-0205-1.

[CCC18] N. Chen, N. Cardozo, S. Clarke, *Goal-Driven Service Composition in Mobile and Pervasive Computing* in *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 49-62, 1 Jan.-Feb. 2018. https://doi.org/10.1109/TSC.2016.2533348.

[CFB04] I. Constantinescu, B. Faltings, W. Binder. *Large scale testbed for type compatible service composition*. Prieiga per internetą: https://www.isi.edu/ikcap/workshops/icaps04/final/constantinescu.pdf, 2004.

[Cyr21] V. Čyras. *Intelektualios sistemos: Dirbtinis intelektas ir žinių vaizdavimas,* Prieiga per internetą: https://klevas.mif.vu.lt/~cyras/AI/ai-cyras.pdf, 2021.

[CMC+15] S. Chhun, K. Malang, C. Cherifi, N. Moalla, Y. Ouzrout. *A Web Service Composition Framework Based on Centrality and Community Structure*. 11th IEEE International Conference on Signal-Image Technology & Internet0Based Systems (SITIS 2015), Nov 2015, Bangkok, Thailand. https://doi.org/10.1109/SITIS.2015.34.

[Dia12] R. Daigneau. *Service Design Patterns , Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services*, 2012, Addison-Wesley, Westford MA

[DYY+07] L. Di, P. Yue, W. Yang, P. Zhao, Y. Wei. (2007). *Ontology-supported automatic service chaining for geospatial knowledge discovery*. 1. 183-192. Prieiga per internetą: http://www.asprs.org/a/publications/proceedings/tampa2007/0019.pdf

[DPW08] M. DiBernardo, R. Pottinger, M. Wilkinson, *Semi-automatic web service composition for the life sciences using the BioMoby semantic web framework*, Journal of Biomedical Informatics, Volume 41, Issue 5, 2008, Pages 837-847, ISSN 1532-0464. https://doi.org/10.1016/j.jbi.2008.02.005.

[YZZ+19] Y. Yuan, W. Zhang, X. Xiuguo, H. Zhai. 2019. "*Dynamic Service Selection Based on Adaptive Global QoS Constraints Decomposition*" *Symmetry* 11, no. 3: 403. https://doi.org/10.3390/sym11030403.

[Jav20] Javatpoint, *Difference between backward chaining and forward chaining.* Prieiga per internetą: https://www.javatpoint.com/difference-between-backward-chaining-and-forward-chaining.

[KC12] M. Kuzu, N.K. Cicekli. *Dynamic planning approach to automated web service composition*. *Appl Intell* 36**,** 1–28 (2012). https://doi.org/10.1007/s10489-010-0238-z.

[KKC11] E. Kaya, M. Kuzu, N.K. Cicekli (2011) *Providing Scalability for an Automated Web Service Composition Framework*. In: Gelenbe E., Lent R., Sakellari G., Sacan A., Toroslu H., Yazici A. (eds) Computer and Information Sciences. Lecture Notes in Electrical Engineering, vol 62. Springer, Dordrecht. https://doi.org/10.1007/978-90-481-9794-1_62.

[KKC12] S. Khakhar, V. Kumar, S. Chaudhary, *Dynamic Service Composition, A Bidirectional Approach using forward and Backward Chaining* in International Journal of Computer Science and Artificial Intelligence, vol. 2, no. 3, pp. 32-42, Sept. 2012. https://doi.org/10.5963/IJCSAI0203004.

[KSR05] U. Küster, M. Stern, B. König-Ries. *A classification of issues and approaches in automatic service composition*. In Proceedings of the First International Workshop on Engineering Service Compositions (WESC05), Amsterdam, The

Netherlands, 1 December 2005. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=61ED6C1F09449AD96367473740A2231F?doi=10.1.1.91.438&rep=rep1&type=pdf.

[Neg11] M. Negnevitsky. *Artificial intelligence: a guide to intelligent systems(3rded.)*. Addison-Wesley, 2011.

[NWG16] H. A.H. Naji, C.Z. Wu, S. Gao, *A Semantic-Aware Approach for Automatic Cloud Services Composition* in International Journal of Grid and Dsitributed Computing, vol. 9, no. 8, pp. 181-196, 2016, doi: https://doi.org/10.14257/ijgdc.2016.9.8.16.

[Omer11] A.M. Omer. *A framework for automatic web service composition based on service dependency analysis*. Prieiga per internetą: https://semanticscholar.org/paper/A-framework-for-Automatic-Web-Service-Composition__Omer/aaac27a1e52c299b288cf40188784193c6d08b79, 2011.

[Pee05] J. Peer. *Web service composition as AI planning – a Survey*. Prieiga per internetą: http://decsai.ugr.es/~faro/CDoctorado/bibliografia/refPlanning4SW/LinkedDocuments/webservice-composition-as-aiplanning-pfwsc.pdf, 2005.

[Rom19] O.J. Romero. (2019) *Cognitively-Inspired Agent-Based Service Composition for Mobile and Pervasive Computing*. In: Wang D., Zhang LJ. (eds) Artificial Intelligence and Mobile Services – AIMS 2019. AIMS 2019. Lecture Notes in Computer Science, vol 11516. Springer, Cham. https://doi.org/10.1007/978-3-030-23367-9_8.

[Soh10] S. Sohrabi (2010) *Customizing the Composition of Actions, Programs, and Web Services with User Preferences*. In: Patel-Schneider P.F. et al. (eds) The Semantic Web – ISWC 2010. ISWC 2010. Lecture Notes in Computer Science, vol 6497. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17749-1_23.

[SPS11] E.G. Silva, L.F. Pires, M. Sinderen. *Towards runtime discovery, selection and composition of semantic service* in Computer Communications journal, vol. 34, no. 2, 2011, https://doi.org/10.1016/j.comcom.2010.04.003.

[TKA+02] Snehal Thakkar, Craig A. Knoblock, Jose Luis Ambite, Cyrus Shahabi. *Dynamically composing Web Services from On-line sources*. Prieiga per internetą: https://aaai.org/Papers/Workshops/2002/WS-02-07/WS02-07-001.pdf, 2002.

[UBM+17] A. Urbieta, A. González-Beltrán, S. Ben Mokhtar, M. Anwar Hossain, L. Capra, *Adaptive and context-aware service composition for IoT-based smart cities*, Future Generation Computer Systems, Volume 76, 2017, Pages 262-274, ISSN 0167-739X, https://doi.org/10.1016/j.future.2016.12.038.

[VPB11] V. Vanitha, V. Palanisamy, K. Baskaran, *Automatic Service Graph Generation for Service Composition in Wireless Sensor Networks*, Procedia Engineering, Volume 30, 2012, Pages 591-597, ISSN 1877-7058, https://doi.org/10.1016/j.proeng.2012.01.903.

[ZBN+04] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, H. Chang. (2004). *QoS-Aware Middleware for Web Services Composition*. Software Engineering, IEEE Transactions on. 30. 311 - 327. https://doi.org/10.1109/TSE.2004.11.

PRIEDAI

1 priedas. Klasikinio atbulinio išvedimo algoritmo pseudokodas

```
bool BC(tikslas)
{
	if (tikslas nepatenka į visų tikslų sąrašą)
	{
		if (tikslas patenka į pradinių faktų sąrašą)
		{
			return true;
		}
		else if (tikslas patenka į jau išvestų faktų sąrašą)
		{
			return true;
		}
		else
		{
			{tikslas pridedamas į visų tikslų sąrašą}
			foreach (taisyklė in taisyklės) //pereiname visas taisykles iš eilės
			{
				if (taisyklės konsekventas yra tikslas)
				{
					{išsaugomi išvestų faktų ir produkcinų sąrašai}
					bool taisyklę_galima_naudoti = true;
					//pereiname visus taisyklės kairės pusės faktus
					foreach (faktas in taisyklė.Antecedentas)
					{
						if (!BC(faktas)
						{
							taisyklę_galima_naudoti = false;
							break;
						}
					}

					if (taisyklę_galima_naudoti == true) //jei taisyklė yra tinkama
					{
						{taisyklės konsekventas pridedamas prie išvestų faktų
sąrašo}

						{taisyklė pridedama prie produkcijų sąrašo}
						{tikslas ištrinamas iš visų tikslų sąrašo}
						return true;
					}
				}
			}

			//jei pereiname per visas taisykles, bet nerandame tinkamos
			{tikslas ištrinamas iš visų tikslų sąrašo}
			return false;
		}
	}
	else
	{
		return true;
	}
}
```

2 priedas. Klasikinio tiesioginio išvedimo algoritmo pseudokodas

```
bool FC(tikslas)
{
	//inicijuojame globalią duomenų bazę su faktais
	//COPY - nukopijuoja paduotą sąrašą (deep copy)
	char[] GDB = COPY(facts);
	bool sustoti;
	//inicijuojame amžinąjį ciklą
	while (true)
	{
		if (tikslas yra GDB)
		{
			return true;
		}

		sustoti = true;
		{padidiname iteracijų skaičių}

		foreach (taisyklė in taisyklės) //pereiname visas taisykles iš eilės
		{
			if (tikslas yra GDB)
			{
				return true;
			}

			if (taisyklėje nėra iškeltas Flag2 ir taisyklėje nėra iškeltas Flag1)
			{
				if (taisyklės konsekventas yra GDB)
				{
					{taisyklei iškeliame Flag2}
				}
				else
				{
					if (taisyklės antecedentas yra GDB)
					{
						//apverčiame sustoti reikšmę
						sustoti = !sustoti;
						{konsekventą pridedame į GDB}
						{taisyklei iškeliame Flag1}
						{taisyklę pridedame į produkcijų sąrašą}
						break;
					}
				}
			}
		}

		if (sustoti == true)
			return false;
	}
}
```

3 priedas. Modifikuoto tiesioginio išvedimo algoritmo pseudokodas

```
bool FC2(tikslai[])
{
	//inicijuojame globalią duomenų bazę su faktais
	//COPY - nukopijuoja paduotą sąrašą (deep copy)
	char[] GDB = COPY(facts);
	bool sustoti;
	while (true) //inicijuojame amžinąjį ciklą
	{
		if (visi tikslai yra GDB)
		{
			return true;
		}

		sustoti = true;
		foreach (taisyklė in taisyklės) //pereiname visas taisykles iš eilės
		{
			if (visi tikslai yra GDB)
			{
				return true;
			}

			if (taisyklėje nėra iškeltas Flag2 ir taisyklėje nėra iškeltas Flag1)
			{
				if (taisyklės kosekvento visi faktai yra GDB)
				{
					{taisyklei iškeliame Flag2}

				}
				else
				{
					if (taisyklės antecedentas yra GDB)
					{
						//apverčiame sustoti reikšmę
						sustoti = !sustoti;

						{konsekvento visus faktus pridedame į GDB}
						{taisyklei iškeliame Flag1}
						{taisyklę pridedame į produkcijų sąrašą}
						break;
					}
				}
			}
		}

		if (sustoti == true)
			return false;
	}
}
```

4 priedas. Produkcijų grupavimo lygiagrečiam vykdymui C# programavimo kalbos kodas

```
//Rule yra taisyklės objektas turinti antecedentą ir konsekventą
var groups = new List<List<Rule>>();

//Produkcijos. Pateikiamos produkcijos yra jau teisingai eilės tvarka.
var productionsCopy = new List<Rule>(_productions);

//Laikinas kintamasis būsenos sekimui
var tempFacts = new List<string>();
tempFacts.AddRange(_facts);

//Vykdome tol kol sugrupuosime visas produkcijas
do
{
	//Ieškoje tinkamų produkcijų į vieną grupę lygindami jų antedecentą su laikinu faktų sąrašo. Surandame visas panašias produkcijas, kurių antecedentas jau yra laikiname faktų sąraše.
	var prodValid = productionsCopy.Where(x => x.LeftSide.All(y => 
	tempFacts.Contains(y)) && !groups.Any(y => y.Any(z => z.Number == 
	x.Number))).ToList();

	groups.Add(new List<Rule>(prodValid));

	//Visų grupės produkcijų konsekventus pridedame prie laikino faktų sąrašo, kad sekančios iteracijos metu galėtumėme rasti sekančią grupę.

	tempFacts.AddRange(prodValid.SelectMany(x => x.RightSide).ToList());

} while (groups.SelectMany(x => x.ToList()).Count() != _productions.Count);
```

5 priedas. Kompozicijos planavimo bei vykdymo rezultatų pavyzdys

