KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

VERSLO INFORMATIKOS KATEDRA

Tadas Milinis

Realiojo laiko sistemų veiksenos įvertinimas

Magistro darbas

Darbo vadovas

dr. D. Makackas

Kaunas, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

VERSLO INFORMATIKOS KATEDRA

Tadas Milinis

Realiojo laiko sistemų veiksenos įvertinimas

Magistro darbas

Kalbos konsultantė
Lietuvių k. katedros
lekt. I. Mickienė
2006-05-18

Recenzentas
doc. dr. E. Valakevičius
2006-05-26

Vadovas
dr. D. Makackas
2006-05-24

Atliko
IFM-0/1 gr. stud.
Tadas Milinis
2006-05-16

Kaunas, 2006

TURINYS

ĮVADAS

Dauguma mus supančio realaus pasaulio sistemų gali būti apibūdinamos kaip realiojo laiko sistemos – pradedant paprastomis sistemomis (elektriniai žaislai, buitinė technika) ir baigiant kritinėmis sistemomis, tokiomis kaip automobilių stabdžių sistema, ar kovinės raketos trasavimas. Labai svarbu išmokti tokias sistemas teisingai projektuoti bei analizuoti jų galimą veikseną.

Realiojo laiko sistema (RLS) – tai sistema, kuri turi atlikti veiksmus per užduotą laiką. Bet kokia sistema gali būti charakterizuojama įeinančiais kintamaisiais, kurie reiškia išorinį poveikį sistemai bei išėjimo kintamaisiais, kurie aprašo sistemos elgesį. Sistemoje galima apibrėžti būsenas, į kurias sistema patenka vienu ar kitu atveju po įvykusio įėjimo signalo, kuris ją perveda iš vienos būsenos į kitą [1].

Problemos aktualumas. Kuriant realiojo laiko sistemas, vienu iš svarbiausių faktorių išlieka sistemos saugumas ir gyvybingumas. Sistemos saugumo sąvoka reiškia, kad sistemos funkcionavime neįvyko nenumatytų situacijų. Gyvybingumo sąvoka reiškia, kad galiausiai buvo atliktas užduotas tikslas ir sistema pasiekė norimą rezultatą [2].

Vienas iš svarbiausių reikalavimų norint sukurti sistemą, tenkinančią nustatytus reikalavimus, yra teisingas jos specifikacijos sudarymas. Galutinis produkto rezultatas tiesiogiai priklauso nuo jo specifikacijos, tai žinoma daugeliui programinių (ir ne tik) sistemų kūrėjams. Sukūrus sistemos specifikaciją, būtina ją verifikuoti ir validuoti. Tai leidžia užtikrinti, kad specifikacija atitinka sistemai keliamus reikalavimus.

Realiojo laiko sistemų specifikacijų verifikavimas yra labai svarbus ir aktualus uždavinys. Nuo specifikacijos priklauso, ar sukursime produktą, kuris tenkintų jam keliamus reikalavimus. Modelių tikrinimas jau dabar naudojamas kaip praktinis įrankis sudėtingų realiojo laiko sistemų (valdikliai, tinklo protokolai) tikrinimui. Tradiciniai verifikavimo būdai neleidžia atlikti pilno sistemų, veikiančių realiame laike, tikrinimo. Pagrindinis to trūkumas, kad tradiciniai verifikavimo būdai neįvertina sistemos funkcionavimo laike.

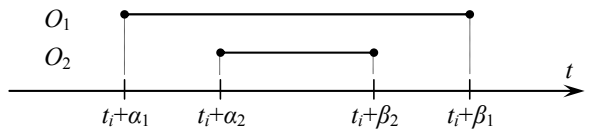
Daugelyje mokslinių darbų [3, 4], sistemos veiksena analizuojama tiriant tos sistemos funkcionavimo trajektorijas: $S_0, e_1, S_1, e_2, S_2 \ldots$, čia S_i – sistemos būsenos žymė, o e_i – įvykis, keičiantis sistemos būseną. Norint pabrėžti ir laiko momentą, kada įvyksta įvykis, keičiantis sistemos būseną, prirašomas ir laiko momentas, kada tas įvykis įvyksta: $S_0, e_1(t_1), S_1, e_2(t_2), S_2 \ldots$, čia t_i – laiko momentas, kada įvyko atitinkamas įvykis e_i.

Šiomis sekomis patogu naudotis nagrinėjant sistemas, kurių operacijų[1] trukmės yra determinuotos. Tokiu atveju, sistemos funkcionavimas aprašomas viena trajektorija.

[1] Operacija vadinama nedalomas sistemoje vykstantis procesas, kurio pabaigoje įvyksta sistemos būsenos pasikeitimas [16].

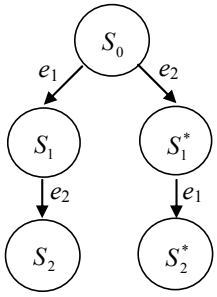
Šiame darbe nagrinėjamos sistemos, kurių operacijų trukmė priklauso tam tikram intervalui. Šiuo atveju, sistemos veikseną aprašyti viena trajektorija nebeįmanoma, nes yra be galo daug operacijos pabaigos laiko momentų (kontinuumo galios aibė). Tuomet tikslinga įvesti tokio pavidalo trajektorijas: $S_0, e_1(I_1), S_1, e_2(I_2), S_2 \ldots$, čia I_i – laiko intervalas. Tokio pavidalo trajektorijas toliau vadinsime *veiksena* ([5] darbe buvo naudojamas *elgsenos* terminas). Taip pat šios trajektorijos turi pasižymėti tokia savybe: nesvarbu, kuriuo laiko momentu iš intervalo I_i įvyks įvykis e_i, tolimesnis sistemos funkcionavimas nepriklausys nuo pasirinkto laiko momento. Šiuo atveju, sistemą galima aprašyti suskaičiuojama tokių sekų aibe [5].

Tai pademonstruosime pavyzdžiu. Tarkime, sistemoje yra dvi aktyvios operacijos O_1 ir O_2, kurios baigsis atitinkamais įvykiais e_1 ir e_2. Įvykis e_1 gali įvykti intervale $I_1 = (t_i + \alpha_1; t_i + \beta_1)$, o įvykis e_2 gali įvykti intervale $I_2 = (t_i + \alpha_2; t_i + \beta_2)$. Grafiškai situacija pavaizduota 1 pav., kai $\alpha_1 < \alpha_2$, $\beta_2 < \beta_1$.



1 pav. Aktyvių operacijų pabaigos laiko momentai

Kadangi sistemoje yra dvi aktyvios operacijos, gali įvykti tiek įvykis e_1, tiek e_2, t.y. gali būti tokio pavidalo trajektorijų šeimos: $S_0, e_1(t_m), S_1, e_2(t_{m+1}), S_2$, arba $S_0, e_1(t_m), S_1^*, e_2(t_{m+1}), S_2^*$, čia S_o – esama sistemos būsena, S_1 – būsena, į kurią sistema pereina įvykus įvykiui e_1, sistemai esant būsenoje S_o. Analogiškai reikia suprasti ir būsenas S_2, S_1^* ir S_2^*. Grafiškai tai galima atvaizduoti medžio struktūros fragmentu (2 pav.).



2 pav. Sistemos funkcionavimo trajektorijos

Sistemos funkcionavimo trajektorijas įtakos įvykių įvykimo laikas (šiuo atveju galimi trys laiko intervalai, nulemiantys skirtingą sistemos funkcionavimą) ir tų įvykių eiliškumas.

Pirmuoju atveju, įvykis e_1 gali įvykti laiko intervale $I_1 = (t_i + \alpha_1, t_i + \alpha_2)$, laiko momentu $t_m \in I_1$. Likęs antrosios aktyvios operacijos įvykis e_2 įvyks laiko intervale $I_2 = (t_i + \alpha_2, t_i + \beta_2)$, laiko momentu $t_{m+1} \in I_2$.

Antruoju atveju, įvykis e_1 gali įvykti laiko intervale $I_1 = (t_i + \alpha_2, t_i + \beta_2)$, laiko momentu $t_m \in I_1$. Kadangi šiuo atveju antrosios aktyvios operacijos įvykis e_2 negali įvykti anksčiau pirmojo, jis įvyks laiko intervale $I_2 = (t_m, t_i + \beta_2)$, laiko momentu $t_{m+1} \in I_2$.

Trečiuoju atveju pirmiau įvyksta įvykis e_2, laiko intervale $I_1 = (t_i + \alpha_2, t_i + \beta_2)$, laiko momentu $t_m \in I_1$. Reiškias likusios aktyvios operacijos įvykis e_1 įvyks ne anksčiau kaip t_m, t.y. laiko momentu $t_{m+1} \in I_2$, kur $I_2 = (t_m, t_i + \beta_1)$.

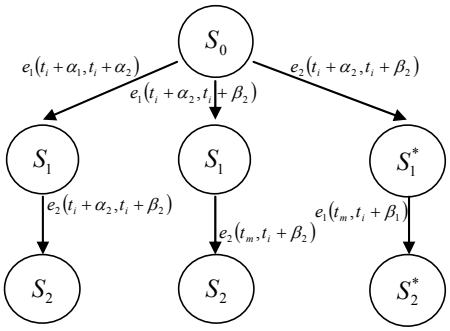
Šias trajektorijas galime išrašyti taip:

$S_0, e_1(t_i + \alpha_1, t_i + \alpha_2), S_1, e_2(t_i + \alpha_2, t_i + \beta_2), S_2$;

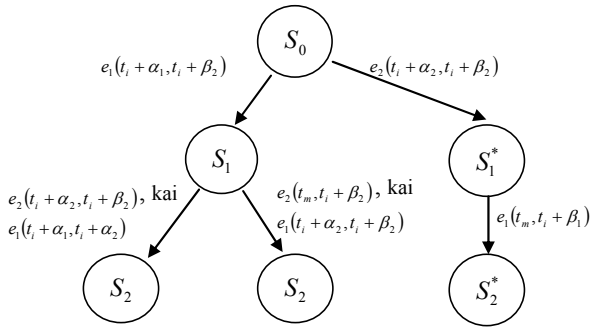
$S_0, e_1(t_i + \alpha_2, t_i + \beta_2), S_1, e_2(t_m, t_i + \beta_2), S_2$, čia $t_i + \alpha_2 < t_m < t_i + \beta_2$;

$S_0, e_2(t_i + \alpha_2, t_i + \beta_2), S_1^*, e_1(t_m, t_i + \beta_1), S_2^*$, čia $t_i + \alpha_2 < t_m < t_i + \beta_2$.

Būsenų grafą galima atvaizduoti dviem būdais (3 ir 4 pav.).



3 pav. Pirmas trajektorijų grafo vaizdavimo būdas



4 pav. Antras trajektorijų grafo vaizdavimo būdas

Šiame darbe naudosime pirmo pavidalo grafo vaizdavimo būdą.

Darbo tikslas yra sukurti matematines prielaidas ir metodus kompiuterizuotai pasiekiamų būsenų grafo, kuris vertina įvykių įvykimo laikus, sudarymui. Tam, kad šis tikslas būtų pasiektas, reikėjo atlikti tokias užduotis:

- Išnagrinėti agregatinio metodo savybes.
- Išnagrinėti pasiekiamų būsenų grafo sudarymo metodiką.
- Atlikti eksperimentus su realiojo laiko sistemomis, t. y. sudaryti realiojo laiko sistemos agregatinę specifikaciją, kuria būtų galima parodyti įrodytiems teiginiams sudarytų algoritmų teisingumą.
- Suformuluoti teiginius pasikartojančių būsenų trajektorijoje paieškai.

Mokslinis naujumas. Šiame darbe pasiūlyta ir ištirta realiojo laiko sistemų, aprašytų agregatais, pasiekiamų būsenų grafo, kuris vertina įvykių įvykimo laikus, sudarymo metodas. Metodas naudoja Simplekso optimizavimo metodą, įvykių įvykimo sritims nustatyti.

Be to, pasikartojančių būsenų paieškai trajektorijoje siūloma lyginti ne tik būsenų struktūrą ir nelygybių formą, bet ir jų apribojimų aibes, kurios aprašo daugiamačius briaunainius, kuriuose yra sukaupta informacija apie įvykių laiko momentus.

Darbo struktūra. *Pirmajame darbo skyriuje* pateikta sudėtingų sistemų formalizavimo metodų, imitacinių modelių apžvalga. Daugiau dėmesio skirta PLA metodui, kuris darbe naudojamas sistemos veiksenos įvertinimui. Taip pat šiame skyriuje yra pateikta lanksčios gamybinės sistemos specifikacija, kurios pagrindu iliustruojami gauti rezultatai. *Antrajame skyriuje* analizuojamos sistemos veiksenos savybės, suformuluojamos penkios lemos ir dvi teoremos. Teoriškai pagrįstas dviejų laiko momentų palyginimas, pasiūlytas būseną apribojančios aibės minimizavimo algoritmas. Taip pat šiame skyriuje aprašomas optimizavimo uždavinio sprendimo Simplekso metodu panaudojimas lyginant laiko momentus. Be to, pasiūlyta metodika ekvivalenčių būsenų paieškai. *Trečiajame skyriuje* analizuojama sistemos veiksena, aprašytas pasiekiamų būsenų grafo sudarymo mechanizmas. Taip pat išanalizuotas atskiras būsenos skaičiavimo atvejis, pateikti pavyzdžiai. *Ketvirtajame skyriuje* plačiau aprašomi ankstesniuose skyriuose naudotų metodų algoritmai, pateikiami šių algoritmų pseudo kodai.

1 SUDĖTINGŲ SISTEMŲ MODELIAVIMAS

Sistema gali būti apibrėžiama kaip esybių (pvz. žmonių, mašinų,…) rinkinys, kurios tarpusavyje sąveikauja siekdamos tam tikrą loginę prasmę turinčio tikslo [6]. Praktikoje, tai ką vadiname sistema, priklauso nuo konkretaus tyrimo tikslų. Nagrinėjamos sistemos sudėtingumas nenusakomas vienareikšmiškai, tai lemia pasirinktas detalizavimo lygis. Nors tikslaus sudėtingų sistemų apibrėžimo nėra, dauguma autorių pateikia tokį bendrų esminių savybių sąrašą [7]:

- jos susideda iš didelio kiekio tarpusavyje sąveikaujančių esybių;
- žinodami esybių veikseną negalime daryti prielaidų apie visos sistemos veikseną;
- sistemos elgesys nėra nulemtas vieno (pagrindinio) valdiklio.

Sudėtingų sistemų modeliavimo svarbą rodo vykstančios tradicinės konferencijos, pavyzdžiui kasmetinės tarptautinės vasaros[2] ir žiemos[3] imitacinio modeliavimo konferencijos, kiekvienais metais pritraukiančios vis daugiau pasaulio mokslininkų. Taigi sistemų modeliavimas – viena iš labiausiai paplitusių metodikų, vartojamų tyrimų ir valdymo moksle.

Šiame skyriuje apibrėšime realiojo laiko sistemą, apžvelgsime formalizavimo metodus bei išsamiau panagrinėsime agregatinį modelių imitavimo ir verifikavimo metodą.

1.1 Realiojo laiko sistemos

Realiojo laiko sistemos apibrėžiamos „…Tai aplinka, charakterizuojama, kaip reaguojanti į atsitiktinai pasirodančius išorinius įvykius. Reakcijos veiksmas atliekamas konkrečiam įvykiui yra eilė vykdomų veiksmų, kurių kiekvienas turi būti atliktas tam tikruose laiko apribojimuose. … Kompiuterinė sistema, kuri yra visapusiškai prilyginama valdymo programai ir yra nustatyta garantuoti atsakymus laiku netgi pikiniais apkrovimais“ [8].

Griežti laikiniai apribojimai ir savalaikiai atsakymai yra pagrindiniai RLS apibrėžiantys faktoriai. Šių faktorių svarbumas yra išreiškiamas dviejuose apibrėžimuose, pateiktuose 1990 m. [9]. Remiantis pirmu apibrėžimu, realiojo laiko sistema turi griežtas, fiksuotas laikines sąlygas. Veiksmai turi būti atlikti atsižvelgiant į apibrėžtas sąlygas, arba sistema neveiks. Realiojo laiko sistema teisingai funkcionuos tik tuo atveju, jei ji grąžins teisingą rezultatą arba atliks teisingus veiksmus per apibrėžtas laikines sąlygas.

Antras apibrėžimas aiškina, kad realiojo laiko sistema yra tokia, kurioje skaičiavimų teisingumas priklauso ne tik nuo loginio teisingumo, bet taip pat ir nuo laiko, per kurį gautas rezultatas. Jeigu sistemoje nėra tenkinamos užduotos laikinės sąlygos, tai sakoma, kad sistemoje įvyko klaida.

[2] "Summer Computer Simulation Conference" – The international forum dedicated to research in modeling and simulation, http://www.scs.org
[3] "Winter Simulation Conference" – Premier international forum, http://www.wintersim.org

Taigi realiojo laiko sistemos veikimo teisingumas gali būti įvertintas matuojant atskirų jos veiksmų atlikimo laiką ir lyginant tai su RLS keliamais reikalavimais. Galima teigti, kad veiksmų atlikimo laikas yra vienas iš svarbiausių iš visų realiojo laiko sistemai keliamų reikalavimų.

1990 m. pateikti apibrėžimai neprieštarauja 1980 m. pateiktiems apibrėžimams [9]. Tačiau skirtingai nuo ankstesnių apibrėžimų, vėlesniuose šaltiniuose realiojo laiko sistemos yra skirstomos į dvi kategorijas: realiojo laiko sistemas, turinčias griežtus reikalavimus ir realiojo laiko sistemas, turinčias tikimybinius reikalavimus. Šiame darbe analizuosime sistemas, kurios turi griežtus reikalavimus.

Realiojo laiko sistemose, turinčiose griežtus reikalavimus, turi būti užtikrintas teisingas veiksmų atlikimas per tiksliai nurodytą laiko tarpą. Tam, kad būtų galima užtikrinti šių reikalavimų išpildymą, reikia apriboti visus sistemoje galimus užlaikymus.

Realiojo laiko sistemų, turinčių griežtus reikalavimus atveju, teisingai atliekant veiksmus, bet jeigu jie užtrunka ilgiau negu per iš anksto nustatytą laiko tarpą, veiksmų atlikimo procesas nutraukiamas. Pavyzdys, iliustruojantis tokį požiūrį:

„ … robotas turi pakelti tam tikrą daiktą nuo konvejerio juostos. Šie daiktai juda, o robotas turi mažą langelį, per kurį turi pakelti daiktus. Jeigu robotas vėluoja privažiuoti prie langelio, tuomet tuo momentu, kai jis norės kelti daiktą, to daikto jau nebebus ir visas roboto darbas bus atliktas neteisingai, nors jis ir bus privažiavęs prie teisingo langelio“ [8].

Realiojo laiko sistemų, turinčių griežtus reikalavimais atveju, veiksmų atlikimo tvarka turi labai svarbią rolę.

Realiojo laiko sistemą, turinčią griežtus reikalavimais, galima apibrėžti kaip sistemą, kuri privalo atlikti tam tikrus veiksmus per nustatytą laiką T, t. y. jei $\tau = \tau(e)$ – sistemos reakcijos laikas į įvykį e, tai turi būti tenkinama sąlyga: $\tau < T$.

Pagrindinė realiojo laiko sistemos su griežtais reikalavimais savybė yra savalaikis veiksmų atlikimas ir jų atitikimas sistemos specifikacijoms.

1.2 Sudėtingų sistemų formalizavimo metodai

Kompiuterių moksle formalizavimo metodai apibrėžiami kaip matematika pagrįsta metodika, skirta programinės ar techninės įrangos sistemų specifikavimui, plėtojimui ir verifikavimui [10]. Formalizavimas ypač reikšmingas didelės integracijos sistemose (kur svarbus saugumas ir patikimumas), kad būtų užtikrintas teisingas sistemos tobulinimo procesas ir išvengta klaidų atsiradimo. Formalizavimo metodai efektyviausi ankstyvosiose vystymo, reikalavimų ir specifikacijų dokumentų ruošimo stadijose, tačiau gali būti naudojamas ir jau įgyvendintos sistemos formalizavimui.

Formalizavimo metodai gali būti naudojami keliais būdais [11]:

a) Sistema specifikuojama formaliai, bet tolesnėse stadijose tobulinama neformaliai. Dažniausiai tai mažiausiai resursų reikalaujantis ir duodantis pakankamai gerus rezultatus panaudojimo atvejis.

b) Formalus tobulinimas ir verifikavimas gali būti naudojamas bandant kurti sistemą formalesniais metodais. Pavyzdžiui, specifikacijoje pateiktos ir įrodytos savybės vystymo metu perkeliamos į sistemos realizaciją. Šis naudojimo atvejis tinkamiausias toms sistemoms, kuriose ypatingas dėmesys turi būti skiriamas saugumui ar patikimumui.

c) Teoremų įrodymo priemonės gali būti panaudotos tam, kad būtų garantuotos visiškai formalios automatiškai patikrinamos tiesos. Tai labai brangus panaudojimo atvejis, tačiau praktikoje dažnai atsiperkantis, ypač jei klaidų kainos sistemoje labai didelės (pvz. kritinių mikroprocesoriaus dalių projektavimas).

Formalizavimo metodus grubiai galime sukvalifikuoti į grupes pagal jų semantines savybes ir pateikimo stilių:

- **Žymėjimo semantikos** grupei priklauso tie metodai, kuriais sistemos prasmė išreiškiama matematinės teorijos erdvėje. Šie metodai remiasi tuo, kad aprašinėjama sistema gali būti nesunkiai suprasta ir aprašyta matematinėje erdvėje, tačiau toli gražu ne kiekviena sistema intuityviai ar iš prigimties gali būti išreiškiama kaip funkcija.
- **Veikimo semantikos** grupei priklauso metodai, kuriais sistemos prasmė išreiškiama kaip paprastesnio skaičiavimo modelio įvykių seka. Šios grupės metodų šalininkai pabrėžia jų modelių aprašų paprastumą ir aiškumą, tačiau kritikai teigia, kad neišspręsta pačios semantikos problema. Nors sudaromi ir modeliai, tačiau semantika nepakankamai apibrėžta ir naudojant sukelti problemų.
- **Aksiomatinės semantikos** grupei priskiriami metodai, kuriais sistemos prasmė išreiškiama išankstinėmis ir po to einančiomis sąlygomis, kurios yra atitinkamai teisingos prieš ir po tam tikros užduoties atlikimo. Gana ryškus šių metodų ryšys su klasikine logika. Tačiau šios grupės metodų kritikai pastebi, kad aksiomatine semantika aprašytos sistemos niekada tiksliai neapibūdina kaip sistema iš tikrųjų „elgiasi“ (tik kokia jos būsena prieš ir po tam tikrų įvykių).

Kai kurie praktikai tvirtina, kad formalizavimo metodų visuma pernelyg viršija pilnai sistemos specifikacijai ar modeliui aprašyti reikalingą formalizmą [12]. Jie tvirtina, kad aprašomosios kalbos išraiškingumas, taip pat modeliuojamų sistemų sudėtingumas padaro visapusiško formalizavimo procesą labai sudėtingu ir sprendimui reikalaujančiu didelių sąnaudų uždaviniu. Kaip alternatyva, buvo pasiūlyta įvairių nesudėtingų formalizavimo metodų, kuriuose akcentuojamas dalinis formalizmas ir dėmesys nukreipiamas į jų pritaikomumą. Tokių

supaprastintų metodų pavyzdžiais yra Alloy objektinių modelių notacija [13], IFAD VDM įrankių rinkinys [14].

Pasaulyje žinoma daugybė formalizavimo metodų ir notacijų, tačiau dažniausiai randami šie:

- Alloy,
- B-Metodas,
- Procesų algebra,
- Agentų modeliai,
- Esterela,
- Lustre,
- Petri tinklai,
- VDM,
- Z.

Alloy specifikacijos kalba paremta predikatų logika. Ji skirta nesudėtingos struktūros modeliams aprašyti. Alloy labiausiai pritaikyta programinės įrangos mikro modeliams aprašyti, tam kad būtų galima patikrinti jų teisingumą. Alloy specifikacijos kalba aprašytus modelius galima analizuoti ir patikrinti „Alloy Analyzer“ įrankiu. Šis įrankis generuoja modelio invariantinius atvejus, modeliuoja operacijų vykdymą ir gali patikrinti vartotojo nurodytas savybes. Žvelgiant grubiai, Alloy yra Z notacijos poaibis. Kompoziciniu požiūriu Alloy mechanizmas turi Z notacijos lankstumą, tačiau remiasi kiek kitokiomis idiomomis.

B-Metodas paremtas abstraktaus mechanizmo notacija, kuri dažnai naudojamas kuriant programinę įrangą. Metodo pradininkas – prancūzas Jean-Raymond Abrial. Kaip ir Alloy specifikacija, B yra susijęs su Z notacija ir gali būti naudojamas kuriant sistemas. Lyginant su Z – B-Metodas yra šiek tiek žemesnio lygmens ir daugiau dėmesio skiriama programinio kodo tikslumui, nei tiesiog formaliai specifikacijai. B-Metodas buvo naudojamas daugelyje kritinio saugumo reikalaujančiose sistemose (tokiose kaip Paryžiaus metro linijos) ir pramonėje susilaukė didelio dėmesio. Metodui taikyti sukurta galingų įrankių, padedančių specifikuoti, projektuoti, analizuoti ar generuoti programinį kodą.

Procesų algebra yra lygiagrečių sistemų formalaus modeliavimo metodų šeima. Nors egzistuoja įvairovė procesų algebros atmainų, tačiau visos jos turi keletą bendrų požymių:

- vietoj visiems prieinamų kintamųjų modifikavimo, nepriklausomų procesų sąveikai pavaizduoti naudojami komunikacijų kanalai (žinutė – perdavimas);
- procesams ir sistemoms aprašyti naudojamos nedidelės primityvų kolekcijos bei operatoriai šiems primityvams sujungti;
- procesų operatoriams apibrėžiamos algebrinės tiesos, kurios leidžia manipuliuoti procesų išraiškomis naudojantis lygtimis.

Kanalų naudojimas komunikacijoms aprašyti yra viena iš savybių, išskiriančių procesų algebrą iš kitų lygiagretumą aprašančių modelių, tokių kaip Petri tinklai ar agentų modeliai.

Agentų modeliai yra matematinis lygiagrečių skaičiavimų modeliai [15], sukurti 1973 m. Šiame modelyje agentu vadinamas lygiagrečių skaičiavimų primityvas: kaip atsaką į gautą žinutę, agentas gali atlikti vietinius sprendimus, sukurti daugiau agentų, išsiųsti daugiau žinučių ir nuspręsti, kaip atsakyti į kitą gautą žinutę. Agentų modelis gali būti naudojamas kaip karkasas lygiagretumo aiškinimui teoriniame lygmenyje bei kaip teorinis pagrindimas praktinių lygiagrečių sistemų realizacijoms.

Esterel yra vienalaikė programavimo kalba, skirta sudėtingoms sistemoms kurti. Imperatyvus Esterel programavimo stilius leidžia nesunkiai išreikšti lygiagretumą, dėl ko ji yra labai tinka valdymo modeliams projektuoti. Kalba pradėta kurti 1980 m. Prancūzijoje. Šiandien Esterel kompiliatoriai gali transliuoti šia kalba parašytas programas ir generuoti C programinį kodą arba techninės įrangos aprašus (VHDL arba Verilog). Kalba vis dar tobulinama, nors jau sukurta keletas kompiliatorių (SCADE, Esterel Studio).

Esterel kalbos privalumai – tikslus programinis valdymas laike; patikimas lygiagretumo specifikavimas valdymo sistemoms; visiškai deterministiniai modeliai; baigtinių būsenų kalba (įmanoma nustatyti vykdymo laiką, gerokai paprastesnis formalus verifikavimas); gali būti naudojama tiek programinei, tiek techninei įrangai aprašyti.

Esterel kalbos trūkumai – aprašinėjimas baigtinėmis būsenomis riboja lankstumą; sudėtingas duomenų apdorojimas; labiausiai tinkamas tik gana paprastiems sprendimus atliekantiems kontroleriams; sudėtingas kompiliavimo procesas.

Lustre – Esterel gimininga formalizavimo kalba, sukurta tų pačių kūrėjų, skirta sinchroniniais duomenų srautais valdomoms reaguojančioms sistemoms aprašyti. Pradėta kurti 1980 m., 1993 m. išsivystė į komercinį produktą, praktiškai naudojamą pramonėje. Dabar ji naudojama kritinėse valdymo sistemose – lėktuvuose, sraigtasparniuose ir atominėse elektrinėse.

Petri tinklai yra vienas iš keleto matematinių metodų diskrečiosioms paskirstytoms sistemoms aprašyti. Tai modeliavimo kalba, kuri orientuota grafo su paaiškinimais pavidalu, vaizduoja paskirstytos sistemos struktūrą. Petri tinklas turi viršūnes, perėjimo mazgus ir orientuotus lankus, jungiančius viršūnes su perėjimo mazgais. Petri tinklus 1962 m. išrado Carl Adam Petri.

Bet kuriuo Petri tinklo vykdymo laiko momentu, bet kuri viršūnė gali turėti tam tikrą kiekį žymių. Priešingai nei tradicinėse duomenų apdorojimo sistemose, kurios gali apdoroti tik vieną srautą įėjimo žymių, Petri tinklų perėjimai gali apimti žymes iš keleto įėjimo vietų, juos atitinkamai apdoroti ir atiduoti žymes į skirtingas išėjimo viršūnes.

Petri tinklai puikiai tinka lygiagrečiai paskirstytų sistemų modeliavimui.

VDM (Vienna Development Method) – sistemų kūrimo metodas, paremtas formalia specifikacija, parašyta VDM-SL specifikavimo kalba. Metodo naudojimui yra sukurti ir palaikomi įrankiai, taip pat sukurtas praplėtimas objektiškai orientuotam programavimui – VDM++. Metodas naudingas modeliais pagrįstoms sistemoms aprašinėti, tačiau netinkamas, jei sistema laikinė.

Z notacija yra formali specifikavimo kalba, naudojama kompiuterinių sistemų aprašymui ir modeliavimui. Pagrindinis Z tikslas – išsami ir aiški kompiuterinės programos specifikacija ir numatytos programos veiksenos formalus įrodymas.

Z išrasta 1970 m., Oksfordo universitete, ji remiasi aibių teorija ir predikatų logika. Z turi standartizuotą dažniausiai naudojamų matematinių funkcijų ir predikatų katalogą. Taip pat Z specifikacijose yra nemažai simbolių, nesančių ASCII kodų lentelėje, todėl specifikacijoje yra numatyti pasiūlymai, kaip koduoti Z notacijos simbolius ASCII kodų lentelės simboliais. Z notacija buvo praktiškai panaudota IBM CICS projekte.

1.3 Sudėtingų sistemų imitacinis modeliavimas

Viena svarbiausių matematinio eksperimentavimo krypčių yra imitacinis modeliavimas. Apie imitacinį modeliavimą galima kalbėti tiek plačiąja, tiek ir siaurąja prasme.

Imitacinis modeliavimas siaurąja prasme – tai eksperimentavimas su gatavais imitaciniais modeliais. Imitacinis modeliavimas plačiąja prasme – tai ne tik eksperimentai su imitaciniais modeliais, bet ir imitacinių modelių sudarymas, prieš pradedant eksperimentus su jais, ir tų modelių tobulinimas eksperimentų metu.

Imitacinis modeliavimas gali būti naudojamas ne tik kaip sistemų analizės, bet ir kaip jų veiklos optimizavimo priemonė, todėl galima kalbėti ne tik apie analitinius sisteminius modelius bei apie analitinius optimizacinius modelius, bet ir apie optimizavimo uždavinių sprendimą analitiniais metodais bei imitacinio modeliavimo priemonėmis.

Pagrindinis skirtumas tarp analitinių ir imitacinių sisteminių modelių yra tas, kad analitinių sisteminių modelių esmę sudaro matematinės formulės, išreiškiančios priklausomybę tarp objekto (sistemos) įėjimų, išėjimų ir būsenų vektorių koordinačių reikšmių – išėjimo ir perėjimo funkcijų matematinės išraiškos, o imitacinių sisteminių modelių esmę sudaro algoritmai, kuriais naudodamiesi galime imituoti objekto (sistemos) funkcionavimą, vidines ir išorines fazines, išėjimų bei įėjimų trajektorijas.

Iš analitinių sisteminių modelių visuomet galima gauti imitacinius modelius, nes, turėdami perėjimo ir išėjimo funkcijas bei žinodami objekto (sistemos) pradinę būseną ir pradines įėjimo trajektorijų atkarpas, mes visuomet galime sudaryti fazinių ir išėjimo trajektorijų pradinių atkarpų apskaičiavimo algoritmus, o kartais – ne tik algoritmiškai, bet ir analitiškai išreikšti išėjimų bei fazinių trajektorijų priklausomybę nuo pradinės būsenos ir įėjimo trajektorijų.

Iš imitacinių sisteminių modelių gauti analizinius modelius pavyksta gana retai, nes imitaciniai modeliai dažniausiai ir sudaromi kaip tik tais atvejais, kai mums nepavyksta nustatyti ir matematiškai išreikšti priklausomybių tarp įėjimų, išėjimų ir būsenų vektorių koordinačių reikšmių.

Ši aplinkybė sukelia ne tik imitacinių modelių trūkumus, lyginant su analitiniais, bet ir didelius jų pranašumus: imitacinius modelius galima kurti ir naudoti ten, kur nepavyksta sudaryti analitinių modelių. Mes, net nesugebėdami surasti ir matematiškai išreikšti priklausomybių tarp sudėtingos sistemos daugybės įėjimų, išėjimų ir būsenos vektoriaus koordinačių reikšmių, galime stebėti sistemos bei jos posistemių funkcionavimą ir, pasinaudodami pastebėtais to funkcionavimo dėsningumais, imituoti jį.

Galima kurti ne tik fizinius, bet ir matematinius imitacinius modelius, operuojant ne „fizinėje", o „matematinėje erdvėje" . Tai yra daug patogiau ir greičiau, nes patį imitavimo algoritmą galima programuoti ir realizuoti kompiuteryje.

Ypač plačiai matematiniai imitaciniai modeliai ir imitacinio modeliavimo metodai taikomi nagrinėjant stochastines sistemas ir stochastinius procesus.

Sistemos, kintančios laike, tokios kaip degalinės, kur automobiliai atvažiuoja ir išvažiuoja (kitaip vadinamos dinaminėmis sistemomis) yra susijusios su atsitiktiniais skaičiais. Niekas negali atspėti, kokiu tiksliu laiko momentu kitas automobilis atvyks į degalinę. Tokia sistema būtų puikus imitacinis modelis. Galime grafiškai apibrėžti tokios sistemos funkcionavimą, grafo viršūne žymėdami degalinėje esančių automobilių skaičių (sistemos būsena). Atvykus automobiliui bei jam išvykstant atitinkamai kinta grafo struktūra. Toks grafas dar vadinamas galimų funkcionavimo trajektorijų grafu ir gali būti gaunamas tiek stebint realiai veikiančią degalinę, tiek dirbtinai modeliuojant sistemą. Tokį dirbtiną grafo konstravimą ir analizę apima imitacinis modeliavimas.

Galimi du modelių tipai: įvykiais valdomi ir tolydaus laiko modeliai. Įvykiais valdomuose modeliuose sistemos būsena kinta tik įvykus tam tikriems įvykiams ir tokių įvykių skaičius yra baigtinis. Tuo tarpu kituose modeliuose sistemos būsena gali kisti visą laiką, ne tik tam tikrais diskrečiais laiko momentais. Pavyzdžiui, vandens lygis rezervuare, į kurį gali būti pilamas, bei išleidžiamas vanduo. Tokiems atvejams modeliuoti kur kas tinkamesni tolydaus laiko modeliai.

Abiejų aukščiau minėtų modelių savybes apima agregatinis metodas.

1.3.1 Įvykiais valdomi modeliai

Įvykiais valdomais modeliais vadinami tokie modeliai, kuriuose sistemos būsena kinta tik diskrečiais (gali būti ir atsitiktiniai) laiko momentais. Sistemos būsena kinta tik tada, kai vyksta įvykiai, o laiko tarpai tarp įvykių sistemos būsenai įtakos neturi. Puikiu įvykiais valdomos sistemos pavyzdžiu gali būti detalių gamykla. Individualios esybės (detalės) yra renkamos pagal atitinkamus

įvykius sistemoje (užsakymų numatymas ar priėmimas). Tarpai tarp įvykių šiuose modeliuose retai būna vienodi.

Gana dažnai, tuo pačiu laiko momentu, reikia manipuliuoti su dviem ar daugiau srauto elementais. Toks lygiagretus srauto valdymas galimas atliekant valdymą nuosekliai tuo pačiu laiko momentu. Dažniausiai tai sukelia logines problemas, kadangi iškyla elementų valdymo eiliškumo klausimas.

Dauguma kompiuterinių, loginių ir klaidų medžio modelių yra valdomi įvykiais. Čia nėra svarbus modeliavimas realiame laike. Kur kas svarbiau gauti modeliavimo rezultatus ir juos nagrinėjant aptikti logines projektavimo klaidas ar neteisingą įvykių nuoseklumą.

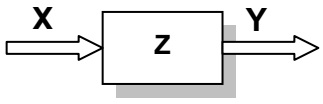
1.3.2 Tolydaus laiko modeliai

Tolydaus laiko modeliuose, laikui bėgant, sistemą apibūdinantys kintamieji gali kisti pastoviai, ne tik tam tikrais diskrečiais laiko momentais. Tokių kintamųjų reikšmės dažnai apibrėžiamos diferencialinėmis lygtimis. Diferencialinės lygtys čia gali būti suprantamos kaip lygtys, kurios reiškia ryšį tarp tolydžiojo kintamojo reikšmės ir jo kitimo koeficiento. Bendru atveju, tolydaus laiko modeliai susideda iš diferencialinių ir algebrinių lygčių rinkinio [16].

Bendru atveju agregatiniame metode tolydinės koordinatės gali būti aprašytos aukščiau minėtu būdu, o PLA atveju – tolydinė koordinatė $z_v(t)$ kinta pagal dėsnį $\frac{dz_v(t)}{dt} = -1$.

1.4 Agregatinio metodo naudojimas modelių imitavimui ir verifikavimui

Agregatinio sistemos specifikavimo požiūriu, sistema vaizduojama kaip tarpusavyje sąveikaujantys atkarpomis tiesiniai agregatai (angl. piece-linear agregate – PLA) aibė [17]. PLA suprantamas kaip objektas su apibrėžtų būsenų aibe Z, įėjimo signalų X ir išėjimo signalų Y aibėmis (5 pav.).



5 pav. Agregato struktūra

$X = \{x_1, x_2, \ldots\};\ Y = \{y_1, y_2, \ldots\};\ Z = \{z_1, z_2, \ldots\}.$

Šis metodas buvo sukurtas Rusijos mokslininkų Buslenkos ir Kovalenkos [18]. H. Pranevičius pasiūlė modifikaciją, t.y. papildė agregatinį aprašymą valdančiomis sekomis, tai sudarė prielaidas patogiam šių sistemų modelių realizavimui kompiuteriu.

Pasiekiamų būsenų metodą 1991 – 1992 m. sistemoje PRANAS – 2 realizavo V. Pilkauskas [19]. Šis metodas daugiausiai buvo taikomas analizuojant įvairius protokolus, pavyzdžiui,

N. Listopadskis, naudodamas šį metodą, tyrė transporto lygio protokolų savybes, tokio kaip X25; L. Sintonen ir H. Pranevičius [20] tyrė įvykiais valdomą telekomunikacinį protokolą.

Sistemai, specifikuotai agregatiniu metodu, galima sudaryti pasiekiamų būsenų grafą atliekant visų agregatų kompoziciją. Aprašyme lieka tik vidiniai įvykiai. Agregatinėje specifikacijoje visos tolydinės komponentės $w(e,t_m)$ keičiamos $w_e = 1$, jei operacija aktyvi, ir $w_e = 0$, jei operacija neaktyvi. Kadangi tolydinės komponentės nepriklauso nuo laiko, tai ir valdančios sekos tampa nereikalingos. Įvykdžius tokį pakeitimą sistemos būsena įgyja pavidalą: $z = \{z_v; w_{e_1}, w_{e_2}, \ldots, w_{e_n}\}$ ir prarandama laiko savybių analizės galimybė.

Pasiekiamų būsenų grafo, neįvertinančio laiko, trūkumas yra tas, kad dėl pašalintų valdančiųjų sekų, yra neįmanoma suskaičiuoti įvykio įvykimo tikimybės, nes yra prarasta informacija apie įvykių įvykimo laiko momentus (jų galimas pasiskirstymo funkcijas). D. Makackas pasiūlė sudarinėti grafą [1] PLA formalizmui, įvertinant valdančias sekas, kai jų nariai yra iš tam tikro intervalo.

1.4.1 PLA formalizmas

Agregato funkcionavimas analizuojamas laiko momentų aibėje $t \in T$. Būsena $z \in Z$, įėjimo signalas $x \in X$ ir išėjimo signalas $y \in Y$ laikomi laikinėmis funkcijomis. Be šių aibių taip pat turi būti apibrėžti perėjimo H ir išėjimo G operatoriai.

Atkarpomis tiesinio agregato būsena $z \in Z$ kiekvienu laiko momentu atitinka atkarpomis tiesinio Markovo proceso būseną $z(t) = (v(t), z_v(t))$, čia $v(t) = \{v_1(t), v_2(t), \ldots, v_p(t)\}$ yra diskretusis būsenos komponentė, įgyjanti reikšmes iš baigtinių reikšmių aibės; ir $z_v(t)$ yra tolydžioji komponentė, susidedanti iš $w(e_1,t_m), w(e_2,t_m), \ldots, w(e_k,t_m)$ koordinačių. Kiekviena iš šių komponenčių nusako, kada įvyks atitinkamas įvykis, galintis keisti agregato būseną.

Agregato būsena gali keistis tiktai dėl dviejų priežasčių: kai į agregatą ateina įėjimo signalas arba tolydinė komponentės reikšmė sutampa su einamu laiku. Įėjimo signalo priėmimo faktas vadinamas išoriniu įvykiu. Nesant įėjimo signalams, agregato būsena kinta pagal tokią priklausomybę $v(t) = const, \frac{dz_v(t)}{dt} = -1$. Kiekvienam įvykiui $e_i'' \in E''$ apibrėžiama valdymo seka $\xi_0^{(i)}, \xi_1^{(i)}, \xi_2^{(i)} \ldots$

Agregato funkcionavimas nagrinėjamas diskrečiais laiko momentais $T = \{t_1, t_2, \ldots, t_m, \ldots\}$, kuriais gali įvykti vienas ar keletas įvykių, iššaukiančių agregato būsenos pasikeitimą. Agregato būsenų aibė E susideda iš dviejų poaibių: $E = E' \cup E''$. Į poaibį $E' = \{e_1', e_2', \ldots, e_N'\}$ įeina įvykiai, sekantys iš įėjimo signalų aibės. E' įvykiai vadinami išoriniais įvykiais. Poaibio $E'' = \{e_1'', e_2'', \ldots, e_N''\}$ įvykiai vadinami vidiniais įvykiais.

Perėjimo operatorius $H(e_i)$ nusako naują agregato būseną $z(t_m) = H\lfloor z(t_m - 0), e_j \rfloor$, $e_i \in E' \cup E''$. Išvedimo operatorius $G(e_i)$ apibrėžia išėjimo signalų turinį $y = G[z(t_m), e_i]$, $e_i \in E' \cup E''$, $y \in Y$.

Paprastai PLA specifikacijos pateikiamos tokia forma:

1. Įėjimo aibė *X*.
2. Išėjimo aibė *Y*.
3. Išorinių įvykių aibė *E′*.
4. Vidinių įvykių aibė *E″*.
5. Valdančios sekos.
6. Diskreti būsenos dedamoji $v(t)$.
7. Tolydinė būsenos dedamoji $z_v(t)$.
8. Pradinė būsena $v(t_0)$, $z_v(t_0)$.
9. Perėjimo operatoriai *H, G*.

1.4.2 Lanksčios gamybinės sistemos specifikacija

Lanksti gamybinė sistema susideda iš penkių aptarnaujančių paraiškas įrenginių (A, B, C, D, E). A įrenginio paraiškos aptarnavimo trukmė $t_A \in (2; 4)$; atitinkamai B – $t_B \in (5; 7)$; C – $t_C \in (14; 16)$; D – $t_D \in (9; 11)$; E – $t_E \in (4; 6)$. Nauja paraiška į sistemą patenka laiko intervalais $e \in (3; 5)$. Visų aptarnavimo įrenginių darbo bei naujų paraiškų patekimo į sistemą tikimybių pasiskirstymas laiko intervaluose tolygus.

Patekusią į sistemą paraišką, gali aptarnauti du įrenginiai: A arba B. Paraiškų skirstymas įrenginiams atliekamas pagal du faktorius: įrenginių aptarnavimo spartą bei užimtumą. Jei sparčiau aptarnaujantis įrenginys A užimtas, paraiška aptarnaujama B įrenginyje. Jei užimtas ir B įrenginys – paraiška neaptarnaujama ir atmetama.

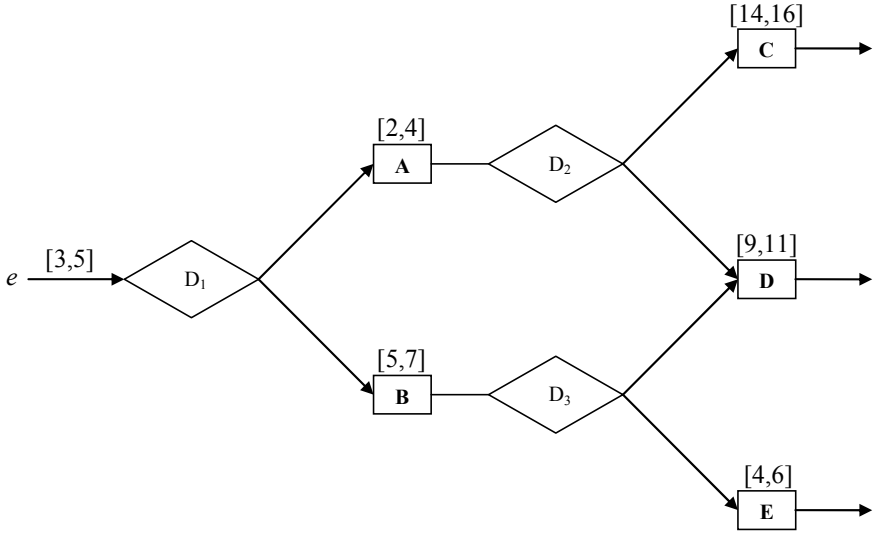
Kai paraiška baigiama aptarnauti A įrenginyje, ji perduodama į *C* arba *D* įrenginius. Jei abu įrenginiai laisvi – perduodama į *D*, priešingu atveju į *C*. Kai paraiška baigiama aptarnauti *B* įrenginyje, ji perduodama į *D* arba *E* įrenginius. Jei abu įrenginiai laisvi – perduodama į *D*, priešingu atveju į *E*.

Jei A arba B įrenginys baigęs paraiškos aptarnavimą neturi kam jos atiduoti – įrenginys blokuojamas, paraiška lieka jame iki tol, kol atsiranda laisvas kitas įrenginys, galintis pratęsti aptarnavimą. Tol, kol įrenginys blokuotas, naujos paraiškos aptarnavimas negali būti pradėtas. Eilių prieš įrenginius nėra.

C, D ir E įrenginiams baigus darbą, paraiška laikoma visiškai aptarnauta ir iš sistemos pašalinama.

Toliau nagrinėjami uždaviniai bus pagrįsti šia realiojo laiko sistema.

Lanksčios gamybinės sistemos schema pavaizduota 6 paveikslėlyje.



6 pav. Lanksčios gamybinės sistemos schema

Lanksčios gamybinės sistemos agregatinė specifikacija

1. **Įėjimo signalų aibė:** $X = \varnothing$.
2. **Išėjimo signalų aibė:** $Y = \varnothing$.
3. **Išorinių įvykių aibė:** $E' = \varnothing$.
4. **Vidinių įvykių aibė:** $E'' = \{e, e_A, e_B, e_C, e_D, e_E\}$,

   kur e – naujos paraiškos sistemoje atsiradimas;

   e_i – įrenginys i baigė paraiškos aptarnavimą.
5. **Valdančios sekos:**

   $e \mapsto S_0, \ldots, S_{n,} \ldots$, $e_A \mapsto S_0^A, \ldots, S_{n,}^A \ldots$, $e_B \mapsto S_0^B, \ldots, S_{n,}^B \ldots$, $e_C \mapsto S_0^C, \ldots, S_{n,}^C \ldots$,

   $e_D \mapsto S_0^D, \ldots, S_{n,}^D \ldots$, $e_E \mapsto S_0^E, \ldots, S_{n,}^E \ldots$.

   kur $S_i \in [3,5]$ – laiko tarpas, tarp naujų paraiškų atsiradimo sistemoje;

   $S_i^A \in [2,4]$ – A įrenginio paraiškos aptarnavimo laikas;

   $S_i^B \in [5,7]$ – B įrenginio paraiškos aptarnavimo laikas;

   $S_i^C \in [14,16]$ – C įrenginio paraiškos aptarnavimo laikas;

   $S_i^D \in [9,11]$ – D įrenginio paraiškos aptarnavimo laikas;

   $S_i^E \in [4,6]$ – E įrenginio paraiškos aptarnavimo laikas.

6. **Diskreti būsenos dedamoji:** $\nu(t) = \{n_A(t), n_B(t), n_C(t), n_D(t), n_E(t), b_A(t), b_B(t)\}$

   kur $n_i(t)$ – i įrenginio eilėje laukiančių paraiškų skaičius;

   $b_A(t) = \begin{cases} 1, \text{ kai A įrenginys blokuotas,} \\ 0, \text{ kai A įrenginys neblokuotas;} \end{cases}$

7. **Tolydinė būsenos dedamoji:** $z_\nu(t) = \{\omega(e,t), \omega(e_A,t), \omega(e_B,t), \omega(e_C,t), \omega(e_D,t), \omega(e_E,t)\}$

   kur $\omega(e,t)$ – laiko momentas, kada atvyks paraiška;

   $\omega(e_i,t)$ – laiko momentas, kada i įrenginys baigs paraiškos aptarnavimą.

8. **Parametrai:**

   l_i – i įrenginio eilės maksimalus ilgis;

9. **Pradinė būsena:** $z(t_0) = \{0,0,0,0,0,0,0,t_0 + S_0, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing\}$

Perėjimo ir išėjimo operatoriai:

Nauja paraiška pateko į sistemą

$H(e)$:

$\omega(e,t_m) = t_m + S_m$,

$$n_A(t_m) = \begin{cases} n_A(t_m)+1, & (\omega(\boldsymbol{e_A},t_m) > t_m \vee b_A(t_m-0)=1) \wedge (\omega(e_B,t_m) > t_m \vee b_B(t_m-0)=1) \wedge \\ & \wedge\, n_A(t_m) < l_A, \\ n_A(t_m-0), & \text{kitu atveju,} \end{cases}$$

$$n_B(t_m) = \begin{cases} n_B(t_m)+1, & (\omega(e_A,t_m) > t_m \vee b_A(t_m-0)=1) \wedge (\omega(e_B,t_m) > t_m \vee b_B(t_m-0)=1) \wedge \\ & \wedge\, n_A(t_m) = l_A \wedge n_B(t_m) < l_B, \\ n_B(t_m-0), & \text{kitu atveju,} \end{cases}$$

$n_C(t_m) = n_C(t_m-0)$, $n_D(t_m) = n_D(t_m-0)$, $n_E(t_m) = n_E(t_m-0)$,

$$\omega(e_A,t_m) = \begin{cases} t_m + S_m^A, & \omega(e_A,t_m-0) < t_m \wedge b_A(t_m-0)=0, \\ \omega(e_A,t_m-0), & \text{kitu atveju,} \end{cases}$$

$$\omega(e_B,t_m) = \begin{cases} t_m + S_m^B, & (\omega(e_B,t_m-0) < t_m \wedge b_B(t_m-0)=0) \wedge \\ & \wedge\, (\omega(e_A,t_m-0) > t_m \vee b_A(t_m-0)=1), \\ \omega(e_B,t_m-0), & \text{kitu atveju.} \end{cases}$$

Įrenginys A baigė paraiškos aptarnavimą:

$H(e_A)$:

$$\omega(e_A,t_m) = \begin{cases} t_m + S_m^A, & n_C(t_m-0) < l_C \vee n_D(t_m-0) < l_D \wedge n_A(t_m-0) > 0, \\ \omega(e_A,t_m-0), & \text{kitu atveju,} \end{cases}$$

$$b_A(t_m)=\begin{cases}1, & n_C(t_m-0)=l_c \wedge n_D(t_m-0)=l_D \wedge \omega(e_C,t_m)>t_m \wedge \omega(e_D,t_m)>t_m\\ 0, & \text{kitu atveju,}\end{cases}$$

$$n_A(t_m)=\begin{cases}n_A(t_m-0)-1, & n_C(t_m-0)<l_C \vee n_D(t_m-0)<l_D \wedge n_A(t_m-0)>0,\\ n_A(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$n_C(t_m)=\begin{cases}n_C(t_m)+1, & \omega(e_D,t_m)>t_m \wedge \omega(e_C,t_m)>t_m \wedge n_D(t_m)=l_D \wedge n_C(t_m)<l_C,\\ n_C(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$n_D(t_m)=\begin{cases}n_D(t_m)+1, & \omega(e_D,t_m)>t_m \wedge \omega(e_C,t_m)>t_m \wedge n_D(t_m)<l_D,\\ n_D(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_C,t_m)=\begin{cases}t_m+S_m^C, & \omega(e_C,t_m-0)<t_m \wedge \omega(e_D,t_m-0)>t_m,\\ \omega(e_C,t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_D,t_m)=\begin{cases}t_m+S_m^D, & \omega(e_D,t_m-0)<t_m,\\ \omega(e_D,t_m-0), & \text{kitu atveju.}\end{cases}$$

Įrenginys B baigė paraiškos aptarnavimą:

$H(e_B)$:

$$\omega(e_B,t_m)=\begin{cases}t_m+S_m^B, & n_D(t_m-0)<l_D \vee n_E(t_m-0)<l_E,\\ \omega(e_B,t_m-0), & \text{kitu atveju,}\end{cases}$$

$$b_B(t_m)=\begin{cases}1, & n_D(t_m-0)=l_D \wedge n_E(t_m-0)=l_E \wedge \omega(e_D'',t_m)>t_m \wedge \omega(e_E'',t_m)>t_m\\ 0, & \text{kitu atveju,}\end{cases}$$

$$n_D(t_m)=\begin{cases}n_D(t_m)+1, & \omega(e_E,t_m)>t_m \wedge \omega(e_D,t_m)>t_m \wedge n_E(t_m)=l_E \wedge n_D(t_m)<l_D,\\ n_D(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$n_E(t_m)=\begin{cases}n_E(t_m)+1, & \omega(e_E,t_m)>t_m \wedge \omega(e_D,t_m)>t_m \wedge n_E(t_m)<l_E,\\ n_E(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_D,t_m)=\begin{cases}t_m+S_m^D, & \omega(e_D,t_m-0)<t_m \wedge \omega(e_E,t_m-0)>t_m,\\ \omega(e_D,t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_E,t_m)=\begin{cases}t_m+S_m^E, & \omega(e_E,t_m-0)<t_m,\\ \omega(e_E,t_m-0), & \text{kitu atveju.}\end{cases}$$

Įrenginys C baigė paraiškos aptarnavimą:

$H(e_C)$:

$$n_C(t_m)=\begin{cases}n_C(t_m)-1, & \omega(e_C,t_m-0)<t_m,\\ n_C(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_C,t_m)=\begin{cases}t_m+S_m^C, & n(t_m-0)>0,\\ \omega(e_C,t_m-0), & \text{kitu atveju,}\end{cases}$$

$b_A(t_m)=0$,

$$n_A(t_m)=\begin{cases}n_A(t_m)-1, & b_A(t_m-0)=1\wedge n_A(t_m)>0,\\ n_A(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_A,t_m)=\begin{cases}t_m+S_m^A, & n_A(t_m)>0\wedge b_A(t_m-0)=1,\\ \omega(e_A,t_m-0), & \text{kitu atveju.}\end{cases}$$

Įrenginys E baigė paraiškos aptarnavimą:

$H(e_E)$:

$$n_E(t_m)=\begin{cases}n_E(t_m)-1, & \omega(e_E,t_m-0)<t_m,\\ n_E(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_E,t_m)=\begin{cases}t_m+S_m^E, & n(t_m-0)>0,\\ \omega(e_E,t_m-0), & \text{kitu atveju,}\end{cases}$$

$b_B(t_m)=0$,

$$n_B(t_m)=\begin{cases}n_B(t_m)-1, & b_B(t_m-0)=1\wedge n_B(t_m)>0,\\ n_B(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_B,t_m)=\begin{cases}t_m+S_m^B, & n_B(t_m)>0\wedge b_B(t_m-0)=1,\\ \omega(e_B,t_m-0), & \text{kitu atveju.}\end{cases}$$

Įrenginys D baigė paraiškos aptarnavimą:

$H(e_D)$:

$$n_D(t_m)=\begin{cases}n_D(t_m)-1, & \omega(e_D,t_m-0)<t_m,\\ n_D(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_D,t_m)=\begin{cases}t_m+S_m^D, & n(t_m-0)>0,\\ \omega(e_D,t_m-0), & \text{kitu atveju,}\end{cases}$$

$b_B(t_m)=0$,

$$n_A(t_m)=\begin{cases}n_A(t_m)-1, & b_A(t_m-0)=1\wedge n_A(t_m)>0,\\ n_A(t_m-0), & \text{kitu atveju,}\end{cases}$$

$$\omega(e_A,t_m)=\begin{cases}t_m+S_m^A, & n_A(t_m)>0\wedge b_A(t_m-0)=1,\\ \omega(e_A,t_m-0), & \text{kitu atveju,}\end{cases}$$

$$n_B(t_m)=\begin{cases} n_B(t_m)-1, & b_B(t_m-0)=1 \wedge n_B(t_m)>0, \\ n_B(t_m-0), & \text{kitu atveju,} \end{cases}$$

$$\omega(e_B,t_m)=\begin{cases} t_m+S_m^B, & n_B(t_m)>0 \wedge b_B(t_m-0)=1, \\ \omega(e_B,t_m-0), & \text{kitu atveju.} \end{cases}$$

2 SISTEMOS VEIKSENOS SAVYBĖS

2.1 Laiko momentų savybės

Laiko momentas aprašomas dviem dedamosiomis- atitinkamo atskaitos taško laiko koordinate t_i bei jo poslinkiu laiko ašyje α. Apibrėžkime laiko momentų aibę T:

$$T=\{t_i+\alpha \mid \alpha \in \mathsf{R}_i, i \in I\}, \text{ čia } i \in \mathsf{N},\ \mathsf{R}_i \in \mathsf{R}_0^+.$$

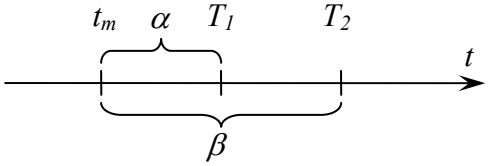
Be to, susitariame, kad jei $i<j$, tada ir $t_i<t_j$.

Esant daugiau kaip vienam laiko momentui, aprašytiems bendra atskaitos taško koordinate t_i, t.y. esant vienodiems indeksams i, galime vienareikšmiškai pasakyti, kuris elementas yra mažiausias.

Remiantis [5] darbe pateiktais teiginiais, iš laiko momentų aibės T pašaliname dalį elementų, tikrai nesančių aibės minimumais.

1 lema. $\min\limits_{\tau\in T}\tau=\min\limits_{i\in I}\tau_i$, čia $\tau_i=\min\limits_{\tau\in T_i}\tau$, $T_i=\{\tau \mid \tau=t_i+\alpha\in T\}$.

▶ Esant vienodiems laiko momentų atskaitos taškams, palyginimą tarp taškų atliekame atsižvelgdami tik į jo poslinkį laiko ašyje. T.y. jei $T_1=\{\tau \mid \tau=t_m+\alpha\}$ ir $T_2=\{\tau \mid \tau=t_m+\beta\}$, o $\alpha<\beta$, tada $T_1<T_2$.



◀

1 išvada. $\tau=t_i+\min\beta$, $\beta\in\{\alpha \mid t_i+\alpha\in T_i\}$.

▶ Iš 1 lemos įrodymo išplaukia, kad aibėje su vienodais koeficientais *i* ieškodami mažiausio elemento turime minimizuoti poslinkį laiko ašyje, t.y. elementas su mažiausiu poslinkiu iš visų galimų ir bus aibės minimumas. ◀

2 išvada. $T=\bigcup\limits_{i\in I}T_i$.

▶ Šios išvados įrodymas seka iš $T_i=\{\tau \mid \tau=t_i+\alpha\in T\}$ apibrėžimo. ◀

Iš aibės $\Theta = \{\tau_i \mid i \in I\}$ pašalinkime visus narius, kuriems egzistuoja laiko momentas su mažesniu t_i indeksu i ir mažesniu poslinkiu, t.y. sudarome aibę:

$$\widetilde{\Theta} = \{\tau_l = t_l + \alpha_l \in \Theta \mid \exists\, \tau_m = t_m + \alpha_m \in \Theta : m < l \wedge \alpha_m < \alpha_l\}.$$

2 lema. $\min_{\tau \in T} \tau = \min_{\tau \in \widetilde{\Theta}} \tau$.

▶ $\widetilde{T}_i = \{\tau = t_l + \alpha_l \mid l > i, \alpha_l > \alpha_i\}$ ◀

2.1.1 Laiko momentų palyginimas

Norint nustatyti, kuris iš aibės elementų $w_1, w_2 \in \widetilde{\Theta}$ yra mažesnis, reikia žinoti papildomas sąlygas apie laiko koordinates. Tarkime, kad $w_1 = t_m + \alpha_m$, $w_2 = t_n + \alpha_n$, $m > n$, $\alpha_m < \alpha_n$, o papildomas sąlygas sudaro tokios formos nelygybių aibė: $t_0 + \alpha < t_i < t_0 + \beta$, $t_{i_1} + \delta_{i_1,i} < t_i < t_{i_2} + \varepsilon_{i_2,i}$, $t_{i_1} + \gamma_{i_1,i_2} < t_{i_2} + \eta_{i_2,i_1}$ čia $i_1, i_2 < i < \max(m,n)$.

Tai galima nustatyti įvertinant skirtumą $w_1 - w_2 = t_m - t_n - (\alpha_n - \alpha_m)$. Jei šis skirtumas yra mažesnis už 0, tada $w_1 < w_2$, o tuo atveju, kai skirtumas didesnis už 0, tada $w_1 > w_2$. Reikia įvertinti skirtumą $t_m - t_n$, nes kitą sumos dedamąją $\alpha_n - \alpha_m$ galime nustatyti iš karto, nes $\alpha_m, \alpha_n \in \mathsf{R}_0^+$. Skirtumo $t_m - t_n$ įvertį iš viršaus ir apačios galima gauti sprendžiant tiesinio programavimo uždavinius:

- ieškant įverčio iš apačios, spręsime tokį tiesinio programavimo uždavinį $c_1 = \min(t_m - t_n)$, o apribojimų aibę sudarys tokia nelygybių sistema

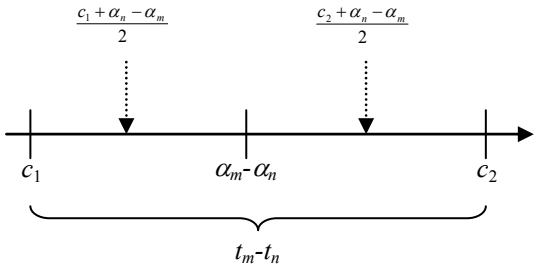
$$\begin{cases} t_1 > \alpha, \\ t_1 < \beta, \\ t_i - t_{i_1} > \delta_{i_1,i}, \\ t_i - t_{i_2} < \varepsilon_{i_2,i}, & \text{čia } i = 1,\dots l,\ i_1, i_2 < i, \\ t_{j_1} - t_{j_2} < \eta_{j_2} - \gamma_{j_2}, & \text{čia } j_1, j_2 < l, \\ 0 < t_1 < t_2 < \dots < t_l; \end{cases}$$

- ieškant įverčio iš viršaus, spręsime tokį tiesinio programavimo uždavinį $c_2 = \max(t_m - t_n)$, o apribojimų aibę sudarys tokia nelygybių sistema

$$\begin{cases} t_1 > \alpha, \\ t_1 < \beta, \\ t_i - t_{i_1} > \delta_{i_1,i}, \\ t_i - t_{i_2} < \varepsilon_{i_2,i}, & \text{čia } i = 1,\dots l,\ i_1, i_2 < i, \\ t_{j_1} - t_{j_2} < \eta_{j_2} - \gamma_{j_2}, & \text{čia } j_1, j_2 < l, \\ 0 < t_1 < t_2 < \dots < t_l. \end{cases}$$

Išsprendus šiuos uždavinius gausime, kad $c_1 < t_m - t_n < c_2$. Dabar galima atsakyti į klausimą kas mažiau: w_1 ar w_2. Galimi trys atvejai:

1. jei $\alpha_n - \alpha_m < c_1$, tada $w_1 - w_2 = t_m - t_n - (\alpha_n - \alpha_m) > t_m - t_n - c_1 > 0$, iš čia seka, kad $w_1 = t_m + \alpha_m > t_n + \alpha_n = w_2$;
2. jei $\alpha_n - \alpha_m > c_2$, tada $w_1 - w_2 = t_m - t_n - (\alpha_n - \alpha_m) < c_2 - (\alpha_n - \alpha_m) < 0$, iš čia seka, kad $w_1 = t_m + \alpha_m < t_n + \alpha_n = w_2$;
3. kai $c_1 < \alpha_n - \alpha_m < c_2$, vienareikšmiškai pasakyti kuris laiko momentas w_1 ar w_2 yra mažesnis – negalima. Tai parodysime taip: pirmu atveju laikysime, kad $t_m - t_n = \frac{c_1 + \alpha_n - \alpha_m}{2}$, o antru atveju – $t_m - t_n = \frac{c_2 + \alpha_n - \alpha_m}{2}$ (7 pav.).



7 pav. $t_m - t_n$ įvertinimo schema

Pirmuoju atveju $w_1 - w_2 = t_m - t_n - (\alpha_n - \alpha_m) = \frac{c_1 + \alpha_n - \alpha_m}{2} - (\alpha_n - \alpha_m) = \frac{c_1 - (\alpha_n - \alpha_m)}{2} < 0$, o antruoju atveju $w_1 - w_2 = t_m - t_n - (\alpha_n - \alpha_m) = \frac{c_2 + \alpha_n - \alpha_m}{2} - (\alpha_n - \alpha_m) = \frac{c_2 - (\alpha_n - \alpha_m)}{2} > 0$.

Galima suformuluoti teiginį apie dviejų momentų palyginimą, kurį tik ką įrodėme.

1 teorema. Tarkime, kad $c_1 < t_m - t_n < c_2$, čia $m > n$, o $\alpha_m < \alpha_n$, tada $t_m + \alpha_m > t_n + \alpha_n$, jei $\alpha_n - \alpha_m < c_1$; $t_m + \alpha_m < t_n + \alpha_n$, jei $\alpha_n - \alpha_m > c_2$, o kai $c_1 < \alpha_n - \alpha_m < c_2$ galimi abu atvejai: tiek $t_m + \alpha_m > t_n + \alpha_n$, tiek $t_m + \alpha_m < t_n + \alpha_n$.

2.1.2 Perėjimo laiko nustatymo pavyzdys (1)

Tarkime, kad analizuojama sistemos būsena yra tokia:

$(0;0;(t_5 + 3, t_4 + 7);(t_5 + 2, t_5 + 12);\varnothing;(t_4 + 14, t_4 + 15);(t_2 + 9, t_2 + 11);\varnothing;R)$, o šios būsenos apribojimų aibė susideda iš tokių nelygybių:

$R = \{t_3 + 3 < t_5 < t_3 + 5;\ t_3 + 2 < t_4 < t_3 + 3;\ t_1 + 3 < t_3 < t_1 + 5;\ t_1 + 2 < t_2 < t_1 + 3;$

$t_0 + 3 < t_1 < t_0 + 5\}$.

Taigi analizuojamų laiko momentų aibės susideda iš $T_\alpha = \{t_5+3; t_5+2; t_4+14; t_2+9\}$ ir $T_\beta = \{t_4+7; t_5+12; t_4+15; t_2+11\}$. Uždavinio tikslas- $\min T_\alpha$ ir $\min T_\beta$.

Analizę pradėkim nuo aibės T_α. Pirmiausia, atmetam tuos aibės elementus, kurie akivaizdžiai didesni už likusius aibės elementus ir jų neanalizuosim. Pagal 1 lemą, $t_5+3 > t_5+2$ (nes atskaitos taškų laiko koordinatės sutampa, o pirmojo elemento poslinkis laiko ašyje 1 didesnis), o pagal 2 lemą gauname, kad $t_2+9 < t_4+14$ (nes pirmojo elemento atskaitos taško laiko koordinatės indeksas mažesnis už antrojo, bei 5 vienetais mažesnis laiko postūmis ašyje). Reiškias, $\min T_\alpha = \min\{t_5+2; t_2+9\}$.

Tam, kad palyginti 2 likusius laiko momentus, sudarysime ir išspręsime tiesinio programavimo uždavinį.

Remdamiesi apribojimų aibe R, darydami prielaidą, kad $t_0 = 0$, sudarome nelygybių sistemą:

$$\begin{cases} t_1 & & & & & & > 3 \\ t_1 & & & & & & < 5 \\ -t_1 & +t_2 & & & & & > 2 \\ -t_1 & +t_2 & & & & & < 3 \\ -t_1 & & +t_3 & & & & > 3 \\ -t_1 & & +t_3 & & & & < 5 \\ & & -t_3 & +t_4 & & & > 2 \\ & & -t_3 & +t_4 & & & < 3 \\ & & -t_3 & & +t_5 & & > 3 \\ & & -t_3 & & +t_5 & & < 5 \end{cases}$$

Pagal nagrinėjamus aibės elementus ($w_1 = t_5+2$ ir $w_2 = t_2+9$) sudarome tikslo funkciją – $t_5 - t_2$ ir simplekso metodu sprendžiame tiesinio programavimo uždavinį minimizuodami ir maksimizuodami sudarytą tikslo funkciją.

$\min(t_5 - t_2) = 3$; $\max(t_5 - t_2) = 8$, t.y. $c_1 = 3 < t_5 - t_2 < 8 = c_2$.

$\alpha_2 - \alpha_1 = 9 - 2 = 7$.

Tiesinio programavimo uždavinio sprendimo Simplekso metodu pavyzdys pateiktas 1 priede.

Kadangi $c_1 < \alpha_2 - \alpha_1 < c_2$, pagal aukščiau pateiktą laiko momentų palyginimo metodą, gauname du galimus laiko momentų įvertinimo atvejus:

- $w_1 < w_2$, kai $3 < t_5 - t_2 < 7$;
- $w_1 > w_2$, kai $7 < t_5 - t_2 < 8$.

Aibės T_β analizė atliekama analogiškai. Pagal 1 lemą gauname, jog $t_4+7<t_4+15$ (atskaitos taškų laiko koordinatės sutampa, o pirmojo elemento poslinkis laiko ašyje 8 vienetais mažesnis), o pagal 2 lemą gauname, kad $t_2+11<t_5+12$ (nes pirmojo elemento atskaitos taško laiko koordinatės indeksas mažesnis už antrojo, bei 1 mažesnis laiko postūmis ašyje). Reiškias, $\min T_\beta=\min\{t_4+7;t_2+11\}$.

Šių elementų palyginimui sudarome tikslo funkciją: t_4-t_2. Apribojimų aibė tokia pati kaip ir T_α aibės analizės atveju. Sprendžiame tiesinio programavimo uždavinį minimizuodami ir maksimizuodami tikslo funkciją:

$\min(t_4-t_2)=2$; $\max(t_4-t_2)=6$, t.y. $c_1=2<t_4-t_2<6=c_2$.

$\alpha_2-\alpha_1=11-7=4$.

Kadangi $c_1<\alpha_2-\alpha_1<c_2$, pagal aukščiau pateiktą laiko momentų palyginimo metodą, gauname du galimus laiko momentų įvertinimo atvejus:

- $w_1<w_2$, kai $2<t_4-t_2<4$;
- $w_1>w_2$, kai $4<t_4-t_2<6$.

Išsamesnė lanksčios gamybinės sistemos veiksenos analizė ir būsenų skaičiavimo pavyzdys pateiktas 2 priede.

2.1.3 Mažiausio laiko momento paieška

Tarkime, kad reikia palyginti ne du laiko momentus, o daugiau. Taikant tokį patį algoritmą galima palyginti bet kuriuos du laiko momentus. Sakykime, kad reikia rasti aibės $T=\left\{t_i+\alpha_i \mid i\in I,\alpha_i\in \mathsf{R}_0^+\right\}$ mažiausią elementą. Be to, visi aibės nariai tenkina sąlygą visiems $t'=t_{i'}+\alpha_{i'}\in T$ neegzistuoja $t''=t_{i''}+\alpha_{i''}$ toks, kad $i'<i''$, $\alpha_{i'}<\alpha_{i''}$, ar $i'>i''$, $\alpha_{i'}>\alpha_{i''}$ t.y. atitinka $\widetilde{\Theta}$ apibrėžimą.

Taip pat tarkime, kad yra duota apribojimų aibė R. Taikant tiesinio programavimo metodus, gausime kiekvienai aibės T elementų porai įvertinimus: $c_{i,j}^1<t_i-t_j<c_{i,j}^2$, $i>j$ ir $i,j\in I$.

Pritaikius 1 teoremą liks nepalyginti tik tie taškai t_i ir t_j, kurių postūmių skirtumas $\alpha_j-\alpha_i$ tenkina sąlygą $c_{i,j}^1<\alpha_j-\alpha_i<c_{i,j}^2$.

Paprastumo dėlei pernumeruokime likusius nepalygintus laiko momentus, kad indeksai eitų iš eilės. Didesniam buvusiam indeksui priskirkime didesnį indeksą. Tokiu būdu, gausime, kad turime n nenagrinėtų taškų: $t_1+\alpha_1$, $t_2+\alpha_2,\dots t_n+\alpha_n$, čia $t_1<t_2<\ldots<t_n$, bei $\alpha_1>\alpha_2>\ldots>\alpha_n$ ir visus įvertinimus $c_{i,j}^1<t_i-t_j<c_{i,j}^2$ ir $c_{i,j}^1<\alpha_j-\alpha_i<c_{i,j}^2$, $1\le i,j\le n$.

Tarkime, norime nustatyti, ar laiko momentas $t_k + \alpha_k$ gali būti mažiausias. Tuo tikslu imkime, kad $t_k - t_i = \frac{c_{ki}^1 + (\alpha_i - \alpha_k)}{2}$, $i \neq k$ ir įsitikinkime, kad $t_k + \alpha_k - (t_i + \alpha_i) < 0$.

$$t_k + \alpha_k - (t_i + \alpha_i) = t_k - t_i - (\alpha_i - \alpha_k) = \frac{c_{ki}^1 + \alpha_i - \alpha_k}{2} - (\alpha_i - \alpha_k) = \frac{c_{ik}^1 - (\alpha_i - \alpha_k)}{2} < 0.$$

Galime padaryti išvadą, kad taškas $t_k + \alpha_k$ prie šių sąlygų yra pats mažiausias. Tuo pačiu įrodėme 2 teoremą.

2 teorema. Tarkim $c_1 < t' - t'' < c_2$ ir neegzistuoja $t' + \alpha', t'' + \alpha'' \in \widetilde{\Theta}$, kad $c_1 < \alpha'' - \alpha' < c_2$. Tada $\min_{\tau \in \widetilde{\Theta}} \tau$ – nustatomas vienareikšmiškai. Kitu atveju, minimumu gali būti tiek $t' + \alpha'$, tiek $t'' + \alpha''$ laiko momentas.

2.1.4 Perėjimo laiko nustatymo pavyzdys (2)

Tarkime, kad analizuojama sistemos būsena yra tokia:

$(0;0;(t_{10}+3, t_{10}+5);(t_{10}+2, t_{10}+4);\varnothing;(t_4+14, t_4+16);(t_9+9, t_9+11);\varnothing;R)$, o šios būsenos apribojimų aibė susideda iš tokių nelygybių:

$R = \{t_8 + 3 < t_{10} < t_8 + 5;\quad t_8 + 2 < t_9 < t_8 + 4;\quad t_5 + 3 < t_8 < t_5 + 5;\quad t_5 + 2 < t_7 < t_5 + 3;$

$t_2 + 9 < t_6 < t_2 + 11;\quad t_2 + 7 < t_5 < t_2 + 8;\quad t_3 + 3 < t_5 < t_3 + 5;\quad t_3 + 2 < t_4 < t_3 + 3;\quad t_1 + 3 < t_3 < t_1 + 5;$

$t_1 + 2 < t_2 < t_1 + 3;\ t_0 + 3 < t_1 < t_0 + 5\}$.

Taigi analizuojamų laiko momentų aibės susideda iš $T_\alpha = \{t_{10}+3; t_{10}+2; t_4+14; t_9+9\}$, ir $T_\beta = \{t_{10}+5; t_{10}+4; t_4+16; t_9+11\}$. Uždavinio tikslas- $\min T_\alpha$ ir $\min T_\beta$.

Analizę pradedam nuo aibės T_α. Pirmiausia, atmetam tuos aibės elementus, kurie akivaizdžiai didesni už likusius aibės elementus ir jų neanalizuosim. Pagal 1 lemą, $t_{10} + 3 > t_{10} + 2$ (nes atskaitos taškų laiko koordinatės sutampa, o pirmojo elemento poslinkis laiko ašyje 1 didesnis). Likusių aibės elementų paprastai palyginti negalime, reiškias toliau analizuojama aibė yra $\min T_\alpha = \min\{t_{10}+2; t_4+14; t_9+9\}$.

Iš aibės T_α imame iš eilės 2 elementus $\{t_{10}+2; t_4+14\}$ ir spręsdami tiesinio programavimo uždavinį juos palyginame. Šiam tikslui pasiekti, darydami prielaidą, kad $t_0 = 0$, sudarome nelygybių sistemą:

$$
\begin{cases}
t_1 & & & & & & & & & & > 3 & (1)\\
t_1 & & & & & & & & & & < 5 & (2)\\
-t_1 & +t_2 & & & & & & & & & > 2 & (3)\\
-t_1 & +t_2 & & & & & & & & & < 3 & (4)\\
-t_1 & & +t_3 & & & & & & & & > 3 & (5)\\
-t_1 & & +t_3 & & & & & & & & < 5 & (6)\\
 & & -t_3 & +t_4 & & & & & & & > 2 & (7)\\
 & & -t_3 & +t_4 & & & & & & & < 3 & (8)\\
 & & -t_3 & & +t_5 & & & & & & > 3 & (9)\\
 & & -t_3 & & +t_5 & & & & & & < 5 & (10)\\
 & -t_2 & & & +t_5 & & & & & & > 7 & (11)\\
 & -t_2 & & & +t_5 & & & & & & < 8 & (12)\\
 & -t_2 & & & & +t_6 & & & & & > 9 & (13)\\
 & -t_2 & & & & +t_6 & & & & & < 11 & (14)\\
 & & & -t_5 & & & +t_7 & & & & > 2 & (15)\\
 & & & -t_5 & & & +t_7 & & & & < 3 & (16)\\
 & & & -t_5 & & & & +t_8 & & & > 3 & (17)\\
 & & & -t_5 & & & & +t_8 & & & < 5 & (18)\\
 & & & & & & & -t_8 & +t_9 & & > 2 & (19)\\
 & & & & & & & -t_8 & +t_9 & & < 4 & (20)\\
 & & & & & & & -t_8 & & +t_{10} & > 3 & (21)\\
 & & & & & & & -t_8 & & +t_{10} & < 5 & (22)
\end{cases}
$$

Pagal nagrinėjamus aibės elementus sudarome tikslo funkciją $-t_{10}-t_4$. Simplekso metodu sprendžiame tiesinio programavimo uždavinį minimizuodami ir maksimizuodami sudarytą tikslo funkciją:

$\min(t_{10}-t_4)=7$; $\max(t_{10}-t_4)=13$, t.y. $c_1=7<t_{10}-t_4<13=c_2$.

$\alpha_2-\alpha_1=14-2=12$.

Kadangi $c_1<\alpha_2-\alpha_1<c_2$, pagal 2.1.1 skyriuje pateiktą laiko momentų palyginimo metodą, gauname du galimus laiko momentų įvertinimo atvejus:

- $t_{10}+2<t_4+14$, kai $7<t_{10}-t_4<12$;
- $t_{10}+2>t_4+14$, kai $12<t_{10}-t_4<13$.

Iš to seka, jog toliau ieškant T_α aibės minimumo, analizuoti reikės du skirtingus atvejus: dvi aibes su papildytomis apribojimų aibėmis $\min T_\alpha=\min\{t_{10}+2;t_9+9\}, R_1=R\cup\{t_4+7<t_{10}<t_4+12\}$ ir $\min T_\alpha=\min\{t_4+14;t_9+9\}, R_1=R\cup\{t_4+12<t_{10}<t_4+13\}$.

Pirmuoju atveju sprendžiame tiesinio programavimo uždavinį, kurio tikslo funkcija $t_{10} - t_9$, o apribojimų aibė papildyta nauja nelygybe – $7 < t_{10} - t_4 < 12$:

$\min(t_{10} - t_9) = 0$; $\max(t_{10} - t_9) = 3$, t.y. $c_1 = 0 < t_{10} - t_9 < 3 = c_2$.

$\alpha_2 - \alpha_1 = 9 - 2 = 7$.

Kadangi $\alpha_2 - \alpha_1 > c_2$, pagal 1 teoremą gauname $t_{10} + 2 < t_9 + 9$. Vadinas, esant $7 < t_{10} - t_4 < 12$, $\min T_\alpha = \{t_{10} + 2\}$.

Antruoju atveju sprendžiame tiesinio programavimo uždavinį, kurio tikslo funkcija $t_9 - t_4$, o apribojimų aibė papildyta nauja nelygybe – $12 < t_{10} - t_4 < 13$:

$\min(t_9 - t_4) = 5$; $\max(t_9 - t_4) = 12$, t.y. $c_1 = 5 < t_9 - t_4 < 12 = c_2$.

$\alpha_2 - \alpha_1 = 14 - 9 = 5$.

Kadangi $\alpha_2 - \alpha_1 < c_1$, pagal 2.1.1 skyriuje pateiktą laiko momentų palyginimo metodą, gauname $t_4 + 14 < t_9 + 9$. Vadinas, esant $12 < t_{10} - t_4 < 13$, $\min T_\alpha = \{t_4 + 14\}$.

Atliekant aibės T_β minimumo paiešką, būtina atsižvelgti į analizuojant T_α aibę gautus apribojimus, t.y. T_β minimumo derėtų ieškoti visais galimais T_α minimumo atvejais. Šiuo atveju T_β minimumo ieškosime dviem atvejais:

- kai $\min T_\alpha = \{t_{10} + 2\}$ ir apribojimų aibė papildyta $7 < t_{10} - t_4 < 12$ nelygybe;
- kai $\min T_\alpha = \{t_4 + 14\}$ ir apribojimų aibė papildyta $12 < t_{10} - t_4 < 13$ nelygybe.

Analizė pradedama atmetant akivaizdžiai aiškius laiko momentus. Toks momentas yra $t_{10} + 5$ (lyginant su $t_{10} + 4$, pagal 1 lemą). Likusių aibės elementų paprastai palyginti negalime, reiškias toliau analizuojama aibė yra $\min T_\beta = \min\{t_{10} + 4; t_4 + 16; t_9 + 11\}$.

Ištirkime pirmąjį T_α minimumo atvejį, nelygybių sistemą papildydami $7 < t_{10} - t_4 < 12$ nelygybe.

Iš aibės T_β imame iš eilės 2 elementus $\{t_{10} + 4; t_4 + 16\}$ ir spręsdami tiesinio programavimo uždavinį juos palyginame. Tikslo funkcija – $t_{10} - t_4$:

$\min(t_{10} - t_4) = 7$; $\max(t_{10} - t_4) = 12$, t.y. $c_1 = 7 < t_{10} - t_4 < 12 = c_2$.

$\alpha_2 - \alpha_1 = 16 - 4 = 12$.

Kadangi $\alpha_2 - \alpha_1 > c_2$, pagal 2.1.1 skyriuje pateiktą laiko momentų palyginimo metodą, gauname $t_{10} + 4 < t_4 + 16$. Iš čia seka, kad $\min T_\beta = \min\{t_{10} + 4; t_9 + 11\}$. Toliau ieškant T_β aibės

minimumo, vėl sprendžiame tiesinio programavimo uždavinį, kurio tikslo funkcija $t_{10} - t_9$. Apribojimų aibė išlieka tokia pati.

$\min(t_{10} - t_9) = 0$; $\max(t_{10} - t_9) = 3$, t.y. $c_1 = 0 < t_{10} - t_9 < 3 = c_2$.

$\alpha_2 - \alpha_1 = 11 - 4 = 7$.

Kadangi $\alpha_2 - \alpha_1 > c_2$, pagal 2.1.1 skyriuje pateiktą laiko momentų palyginimo metodą, gauname $t_{10} + 4 < t_9 + 11$. Vadinas, esant $7 < t_{10} - t_4 < 12$, $\min T_\beta = \{t_{10} + 4\}$.

Antruoju T_α minimumo atveju, nelygybių sistemą papildom $12 < t_{10} - t_4 < 13$ nelygybe.

Iš aibės T_β imame iš eilės 2 elementus $\{t_{10} + 4; t_4 + 16\}$ ir spręsdami tiesinio programavimo uždavinį juos palyginame. Tikslo funkcija – $t_{10} - t_4$:

$\min(t_{10} - t_4) = 12$; $\max(t_{10} - t_4) = 13$, t.y. $c_1 = 12 < t_{10} - t_4 < 13 = c_2$.

$\alpha_2 - \alpha_1 = 16 - 4 = 12$.

Kadangi $\alpha_2 - \alpha_1 < c_1$, pagal 2.1.1 skyriuje pateiktą laiko momentų palyginimo metodą, gauname $t_{10} + 4 > t_4 + 16$. Iš čia seka, kad $\min T_\beta = \min\{t_4 + 16; t_9 + 11\}$. Toliau ieškant T_β aibės minimumo, vėl sprendžiame tiesinio programavimo uždavinį, kurio tikslo funkcija $t_9 - t_4$. Apribojimų aibė išlieka tokia pati.

$\min(t_9 - t_4) = 5$; $\max(t_9 - t_4) = 12$, t.y. $c_1 = 5 < t_9 - t_4 < 12 = c_2$.

$\alpha_2 - \alpha_1 = 16 - 11 = 5$.

Kadangi $\alpha_2 - \alpha_1 < c_1$, pagal 2.1.1 skyriuje pateiktą laiko momentų palyginimo metodą, gauname $t_4 + 16 < t_9 + 11$. Vadinas, esant $12 < t_{10} - t_4 < 13$, $\min T_\beta = \{t_4 + 16\}$.

2.2 Būsenos apribojimų aibės savybės

Realaus laiko sistemų analizės metode, kuris buvo pateiktas 2 skyriuje, yra įvesta būsenos laiko momento apribojimo aibės R sąvoka. Šioje būsenos apribojimų aibėje yra susiejami laiko momentai $t_0, t_1, t_2, \ldots$ Šios aibės apribojimai yra naudojami, kai reikia palyginti laiko momentus su atitinkamais postūmiais. Ankstesniame skyriuje buvo nagrinėtos dvi to paties uždavinio būsenos: 2.1.2 skyriuje yra nagrinėjama būsena po 4 įvykio, o 2.1.4 skyriuje – po 10 įvykio.

Pirmuoju atveju apribojimų aibę R sudarė 8 nelygybės, aprašančios 4 kintamųjų t_1, t_2, t_3, t_4 tarpusavio ryšį. Ieškant minimumų $\min T_\alpha$ ir $\min T_\beta$, nelygybių sistemą sudarydavo 10 nelygybių ir reikėdavo atlikti 8 iteracijas sprendžiant tiesinio programavimo uždavinius.

Antruoju atveju apribojimų aibę R sudarė 22 nelygybės, aprašančios 10 kintamųjų t_1, $t_2, \ldots t_{10}$ tarpusavio ryšį. Ieškant minimumų $\min T_\alpha$ ir $\min T_\beta$, nelygybių sistemą sudarydavo 32 nelygybės ir reikėdavo atlikti 19 iteracijų sprendžiant tiesinio programavimo uždavinius.

Nors tikslo funkcijoje yra tik du kintamieji, tačiau minimumo paieškos laikas priklauso nuo apribojimų nelygybių skaičiaus. Didėjant lygčių skaičiui, didėja daugiamačio briaunainio viršūnių skaičius. Kartu didėja tikimybė, kad reikės perrinkti didesnį skaičių viršūnių nustatant tikslo funkcijos minimumą ar maksimumą.

Kyla klausimas, ar galima sumažinti lygčių skaičių, nepaveikiant uždavinio sprendimo rezultato.

Tarkime, kad reikia palyginti laiko momentus $t_{10}+2$ ir t_9+9, bei žinome, kad $15 < t_8 < 20$. Šį t_8 įvertinimą galime gauti spręsdami tiesinio programavimo uždavinį, kurio nelygybių sistemą sudaro 2.1.4 skyriuje sudarytos nelygybių sistemos 1-18 lygtys, o tikslo funkcija tiesiog t_8. Gauname:

$\min(t_8)=15$; $\max(t_8)=20$, t.y. $c_1 = 10 < t_8 < 20 = c_2$.

Be to, galime pastebėti, kad laiko momentas t_{10} yra išreiškiamas per t_8, o taip pat ir t_9 išreiškiamas per t_8 (žiūr. 2.1.4 sk.), t.y. abu laiko momentai išreiškiami per tą patį laiko momentą. Todėl lygindami $t_{10}+2$ ir t_9+9, skaičiavimams naudosime tik tas lygtis, kurios susiję išreiškiant nagrinėjamuosius laiko momentus per bendrąjį, o nelygybių sistemą dar papildysime t_8 įvertinimu.

Taigi, šiuo atveju, apribojimų aibėje užteks 6 lygčių:

$$\begin{cases} -t_8 \quad\quad +t_{10} & > 3 \\ -t_8 \quad\quad +t_{10} & < 5 \\ -t_8 \quad +t_9 & > 2 \\ -t_8 \quad +t_9 & < 4 \\ t_8 & > 15 \\ t_8 & < 20 \end{cases}$$

Spręsime tiesinio programavimo uždavinį, kurio tikslo funkcija $-t_{10}-t_9$. Gauname:

$\min(t_{10}-t_9)=0$; $\max(t_{10}-t_9)=3$, t.y. $c_1 = 0 < t_{10}-t_9 < 3 = c_2$.

$\alpha_2 - \alpha_1 = 9-2 = 7$.

Kadangi $\alpha_2 - \alpha_1 > c_2$, pagal 2.1.1 skyriuje pateiktą laiko momentų palyginimo metodą, gauname $t_{10}+4 < t_9+11$. Gautas rezultatas tapačiai lygus analogiškam uždaviniui, spręstam 2.1.4 skyriuje.

Pateiktas pavyzdys rodo, kad, bandant palyginti laiko momentus, visiškai nebūtina spręsti sudėtingą tiesinio programavo uždavinį, kurio nelygybių sistemą sudaro visi tiriamą sistemos būseną aprašantys apribojimai. Galima gerokai suprastinti uždavinio sprendimą bei padidinti algoritmo veikimo našumą išreiškiant visus tiriamos aibės laiko momentus per visiems jiems bendrą laiko momentą ir sprendžiant tik susijusias nelygybes.

Aukščiau pateiktame pavyzdyje nesunkiai intuityviai nustatėme bendrą laiko momentą. Tačiau kyla klausimas, kaip elgtis bendru atveju. Kaip rasti laiko momentą, kurio atžvilgiu galima išreikšti visus nagrinėjamus laiko momentus.

Tarkime, turime k laiko momentų aibę $\{t_{i_1}+\alpha_{j_1}; t_{i_2}+\alpha_{j_2}; \dots t_{i_k}+\alpha_{j_k}\}$. Reikia rasti visiems elementams bendrą laiko momentą, kurio atžvilgiu būtų galima išreikšti visus šios aibės laiko momentus.

Šiam uždaviniui spręsti siūlomas algoritmas atrodo taip: sudaroma pradinė aibė K, kurioje saugomi analizuojamų laiko momentų atskaitos taškų indeksai i, t.y. $K=\{i_1,i_2,\dots,i_n\}$, čia $i\in\mathsf{N}$, o pasikartojantys aibės elementai atmetami. Pradedamas vykdyti ciklas: surandamas aibės K maksimumas $n=\max K$; apribojimų aibėje išrenkami laiko momentai, kuriais išreiškiamas laiko momentas t_n; šių išrinktų momentų indeksais papildoma aibė K ir pašalinamas iš jos elementas n. Taip ciklas kartojamas tol, kol aibėje K lieka vienas elementas.

Tarkime, kad reikia rasti aibės $T_\alpha=\{t_{10}+3; t_{10}+2; t_4+14; t_9+9\}$ minimumą, t.y. uždavinys identiškas 2.1.4 skyriuje aprašytam uždaviniui, tik šį kartą jį išspręsime suprastindami sprendimą ir išmesdami dalį nelygybių sistemos lygčių, tokiu būdu pademonstruodami aukščiau aprašyto algoritmo veikimą.

Skaičiavimo pradžia analogiška anksčiau naudotam metodui. Iš aibės T_α šaliname akivaizdžiai didesnius už likusius aibės elementus ir jų toliau neanalizuojam. Pagal 1 lemą, $t_{10}+3>t_{10}+2$ (nes atskaitos taškų laiko koordinatės sutampa, o pirmojo elemento poslinkis laiko ašyje 1 didesnis). Likusių aibės elementų paprastai palyginti negalime, reiškias toliau analizuojama aibė yra $\min T_\alpha=\min\{t_{10}+2; t_4+14; t_9+9\}$.

Sudarome aibę K, kurioje surašome analizuojamos aibės laiko momentų atskaitos taškų indeksus: $\max K_1=\{10, 9, 4\}$. Aibėje K_1 ieškome maksimumo $\max K_1=\{10\}$. Pagal apribojimų aibę nustatome, kokiais laiko momentais yra išreiškiamas t_{10}. Pagal apribojimų aibės nelygybes matome, kad t_{10} išreiškiamas t_8 laiko momentu (21), (22) nelygybės. $\max K_1$ elementą keičiame jį išreiškiančių laiko momentų indeksais ir gauname K_2 aibę: $K_2=\{8, 9, 4\}$, be to, (21) ir (22) nelygybes įtraukiame į sąrašą nelygybių, kurios bus reikalingos uždavinio sprendimui. Ciklą kartosime, tol kol aibėje K_i liks tik vienas elementas:

$K_2 = \{8, 9, 4\}$; $\max K_2 = \{9\}$; t_9 išreiškiamas per t_8 – pagal (20), (19) nelygybes;

$K_3 = \{8, 4\}$; $\max K_3 = \{8\}$; t_8 išreiškiamas per t_5 – pagal (18), (17) nelygybes;

$K_4 = \{5, 4\}$; $\max K_4 = \{5\}$; t_5 išreiškiamas per t_2 ir t_3 – pagal (12), (11), (10), (9) nelygybes;

$K_5 = \{2, 3, 4\}$; $\max K_5 = \{4\}$; t_4 išreiškiamas per t_3 – pagal (8), (7) nelygybes;

$K_6 = \{2, 3\}$; $\max K_6 = \{3\}$; t_3 išreiškiamas per t_1 – pagal (6), (5) nelygybes;

$K_7 = \{2, 1\}$; $\max K_7 = \{2\}$; t_2 išreiškiamas per t_1 – pagal (4), (3) nelygybes;

$K_8 = \{1\}$;

Kadangi aibė K_8 turi tik vieną elementą, skaičiavimą nutraukiame. Belieka įvertinti paskutinio gauto laiko momento (šiuo atveju t_1) apribojimus. Pagal (1) ir (2) nelygybes gauname, kad 1.

Tolesnis uždavinio sprendimas analogiškas pateiktam 2.1.4 skyriuje, išskyrus tai, jog spręsdami tiesinio programavimo uždavinį galėsime atmesti dalį nelygybių, taip suprastindami uždavinio sprendimą ir padidindami našumą, t.y. nelygybių sistemą sudarys šios nelygybės: 1-12, 17-20.

Įsitikinkime uždavinio sprendimo rezultatų teisingumu:

Nagrinėjami pirmieji du aibės T_α elementai. Tikslo funkcija – $t_{10} - t_4$.

$\min(t_{10} - t_4) = 7$; $\max(t_{10} - t_4) = 13$, t.y. $c_1 = 7 < t_{10} - t_4 < 13 = c_2$.

$\alpha_2 - \alpha_1 = 14 - 2 = 12$.

$c_1 < \alpha_2 - \alpha_1 < c_2$, todėl $t_{10} + 2 < t_4 + 14$, kai $7 < t_{10} - t_4 < 12$ ir $t_{10} + 2 > t_4 + 14$, kai $12 < t_{10} - t_4 < 13$. Toliau ieškant T_α aibės minimumo, analizuojam du skirtingus atvejus: dvi aibes su papildytomis apribojimų aibėmis $\min T_\alpha = \min\{t_{10} + 2; t_9 + 9\}$, su papildomu apribojimu $\{t_4 + 7 < t_{10} < t_4 + 12\}$ ir $\min T_\alpha = \min\{t_4 + 14; t_9 + 9\}$, su papildomu apribojimu $\{t_4 + 12 < t_{10} < t_4 + 13\}$.

Pirmuoju atveju sprendžiame tiesinio programavimo uždavinį, kurio tikslo funkcija $t_{10} - t_9$, o apribojimų aibė papildyta nauja nelygybe – $7 < t_{10} - t_4 < 12$:

$\min(t_{10} - t_9) = 0$; $\max(t_{10} - t_9) = 3$, t.y. $c_1 = 0 < t_{10} - t_9 < 3 = c_2$.

$\alpha_2 - \alpha_1 = 9 - 2 = 7$. Vadinas, esant $7 < t_{10} - t_4 < 12$, $\min T_\alpha = \{t_{10} + 2\}$.

Antruoju atveju sprendžiame tiesinio programavimo uždavinį, kurio tikslo funkcija $t_9 - t_4$, o apribojimų aibė papildyta nauja nelygybe – $12 < t_{10} - t_4 < 13$:

$\min(t_9 - t_4) = 5$; $\max(t_9 - t_4) = 12$, t.y. $c_1 = 5 < t_9 - t_4 < 12 = c_2$.

$\alpha_2 - \alpha_1 = 14 - 9 = 5$. Vadinas, esant $12 < t_{10} - t_4 < 13$, $\min T_\alpha = \{t_4 + 14\}$.

Palyginę gautus rezultatus su 2.1.4 skyriuje spręsto analogiško uždavinio rezultatais matome, kad apribojimų aibės nelygybių skaičiaus sumažinimas įtakos rezultato teisingumui neturėjo.

Akivaizdus kitas nelygybių atmetimo atvejis. Tarkime, kad turime nelygybių apribojimų aibę $t_i - t_j < \alpha$, čia $\alpha \in \mathsf{R}_\alpha \subset \mathsf{R}_0^+, i > j$. Tada galime pakeisti šias nelygybes viena, kur $t_i - t_j < \min \mathsf{R}_\alpha$. Analogiškai, jei nelygybių aibė $t_i - t_j > \beta$, čia $\beta \in \mathsf{R}_\beta \subset \mathsf{R}_0^+$ – galime pakeisti viena $t_i - t_j > \max \mathsf{R}_\beta$ nelygybe.

Nagrinėtas pavyzdys rodo, kad reikia tirti nelygybių aibės savybes, tai galėtų sumažinti būsenos nagrinėjimo laiką bei reikalingus resursus.

2.2.1 Minimali apribojimų aibės sritis

Nesunkiai įrodomas teiginys apie būsenos apribojimų aibės iškilumą.

3 lema. Būsenos apribojimų aibės R nelygybės aprašo iškilą daugiamatės erdvės sritį.

▶ Šios lemos įrodymas seka iš to, kad visos nelygybės būsenos apribojimų aibėje yra tiesiniai dariniai. Nesunkiai įrodoma, kad iškilų aibių sankirta yra taip pat iškila aibė ir tiesinis darinys yra taip pat iškiloji aibė. ◀

4 lema. Būsenos aibės *R* aprašytas daugiamatis daugiakampis yra aprėžtas t.y. visos kiekvienos daugiakampio viršūnės koordinatės yra baigtinės.

▶ Įrodymas seka iš aibėje *R* esančių nelygybių struktūros. Aibėje *R*, kiekvienam t_i egzistuoja nelygybė $t_l + a_l < t_i < t_m + b_m$, čia $0 < l, m < i$, $a_l, b_l \in \mathsf{R}_0^+$. Pradinis laiko momentas t_0 tenkina nelygybę $a < t_0 < b$, čia $a, b \in \mathsf{R}_0^+$. Reikia parodyti, kad visiems *i*, apibrėžtiems aibėje *R*, galioja nelygybės $a_i < t_i < b_i$, čia $a_i, b_i \in \mathsf{R}_0^+$. Tai įrodysime matematinės indukcijos būdu.

Tarkime, kad R_0 apibrėžia vieno matavimo erdvės kūną (atkarpą), tuomet jame yra tik viena nelygybė – $R \in \{a < t_0 < b\}$, t.y. taškas t_0 aprėžtas.

Kai R_1 apibrėžia dviejų matavimų erdvės kūną (plokštumoje), ją sudaro dvi nelygybės: $R_1 \in \{R_0 \cup t_0 + a_1 < t_1 < t_0 + b_1\}$. Iš čia seka, jog $a + a_1 < t_1 < b + b_1$. Vadinasi, taškai t_0 ir t_1 bus visada aprėžti.

Kai $R_{n\text{-}1}$ apibrėžia n matavimų erdvės kūną, jį sudaro tokios nelygybės: $R_{n-1} \in \{R_{n-2} \cup t_{n'} + a_{n'} < t_n < t_{n''} + b_{n''}\}$, čia $n', n'' < n$ ir $a_{n'}, b_{n''} \in \mathsf{R}_0^+$.

Parodysime, kad ši savybė galioja ir kito perėjimo metu, t.y. laiko momentu t_m. Šiuo atveju $R_n \in \{R_{n-1} \cup t_s + a_s < t_n < t_p + b_p\}$, kadangi sekančio įvykio įvykimo laiko momentas gali priklausyti tik nuo ankstesnių laiko momentų, tai $s, p < n$ $\left(a_s, b_p \in \mathsf{R}_0^+\right)$.

Iš padarytos indukcinės prielaidos seka, kad $c < t_s, t_p < d$, čia s ir $d \in \mathsf{R}_0^+$, dėl to $a_s + c < t_n < b_p + d$. ◀

2.2.2 Ekvivalenčios būsenos

Matematikoje ekvivalentumas suprantamas taip: t.y. sąryšis „~“ turi tenkinti tris savybes: $a \sim a$ (refleksyvumo), $a \sim b \Rightarrow b \sim a$ (simetriškumo) ir $a \sim b$, $b \sim c \Rightarrow a \sim c$ (tranzityvumo).

Mūsų atveju, ekvivalenčiomis būsenomis laikysime, kai sistemos tolimesnis funkcionavimas nuo ekvivalenčių būsenų yra vienodas, t.y. invariantiškas laiko atžvilgiu. Šiame skyrelyje bus pateiktas ekvivalenčių būsenų radimo būdas.

Nustatinėjant dviejų n-matės erdvės kūnų ekvivalentumą, reikia palyginti tuos kūnus aprašančias nelygybes. Sprendžiant šį uždavinį, pirmiausia reikia iš apribojimų aibės pašalinti tas nelygybes, kurios neturi įtakos aprašomajam kūnui kitų lygčių atžvilgiu:

- Pagal [2.2.1] skyriuje pateiktą algoritmą, minimizuojamas lygčių skaičius. Gaunamos nelygybės:

  $\begin{cases} t_i - t_j < b_{ij}^k \\ t_i - t_j > a_{ij}^l \end{cases}$, čia $0 \le i, j \le n$, $j < i$, $t \le k \le I_{ij}^b \in$.

  Kiekvienoje nelygybėje tik po du kintamuosius.

- Iš gautos nelygybių aibės šalinamos neturinčios įtakos nelygybės (pagal 1 lemą).

$$\begin{cases} t_i - t_j < \min\limits_{1 \le k \le I_{ij}^a} \left\{ b_{ij}^k \right\} = b_{ij} \\ t_i - t_j > \max\limits_{1 \le l \le I_{ij}^a} \left\{ a_{ij}^l \right\} = a_{ij} \end{cases} \qquad (*)$$

Tokiu būdu gaunamos daugiamatės erdvės kūną aprašančios nelygybės, kurios įtakoja tolimesnę sistemos veikseną.

Toliau atliekamas dviejų n-matės erdvės kūnų palyginimas, t.y. lyginamos dvi nelygybių aibės. Iš eilės imamos pirmosios aibės nelygybės ir lyginamos su antrosios aibės nelygybėmis. Vienodos nelygybės šalinamos iš aibių. Jei po šios procedūros aibės tampa tuščios – erdvės kūnai tapačiai lygūs. Priešingu atveju būtina analizuoti likusias nelygybes, patikrinant, ar jos kerta nagrinėjamą daugiakampį. Tai atliekama sprendžiant optimizavimo uždavinį, kurio apribojimų aibę sudaro pašalintos nelygybės, o tikslo funkcija sudaroma iš vienos likusių nelygybių. Išsprendus optimizavimo uždavinį gaunamas atitinkamų laiko momentų įvertinimas. Jei šis įvertis patenka į tiriamos nelygybės apribojimų intervalą – nelygybės aprašoma plokštuma kerta nagrinėjamąjį daugiakampį, tokiu būdu jį patikslindama, t.y. lyginami erdvės kūnai nėra tapačiai lygūs.

Tarkime, kad liko nelygybė $t_m - t_n < b_{mn}$, tuomet tikslo funkcija $z = t_m - t_n$. Įvertinę z gavome, kad $a < z < b$, tada jei $b_{ij} < a$ – nagrinėta sistema neturi sprendinio; jei $a < b_{ij} < b$ – nelygybė įtakoja daugiakampį; jei $b_{ij} > b$, nelygybę galima atmesti, kaip nedarančią įtakos.

2.2.3 Ekvivalenčių būsenų nustatymo pavyzdys

Norėdami pademonstruoti ekvivalenčių būsenų nustatymo metodiką, pasinaudosime [20] darbe pateiktu dvikanalės aptarnavimo sistemos pasiekiamų būsenų grafu tolydaus laiko atveju. Sistemos specifikacija ir visos pasiekiamų būsenų medžio viršūnės pateiktos 3 priede.

Bandysime patikrinti 17-os ir 44-os būsenos ekvivalentumą. Tiriamos būsenos:

17: $(0;(t_5+4,t_5+6),(t_5+3,t_5+5),\varnothing;R_{101})$, čia $R_{101} = R_{62} \cup \{t_4 < t_5 < t_3 + 6$, $t_3 + 4 < t_4 < t_3 + 5$, $t_1 + 4 < t_3 < t_1 + 6$, $t_1 + 3 < t_2 < t_1 + 4$, $t_0 + 4 < t_1 < t_0 + 6$, $4 < t_0 < 6$;

44: $(0;(t_9+4,t_9+6),(t_9+3,t_9+5),\varnothing;R_{431})$, čia $R_{431} = \{t_6 + 4 < t_9 < t_6 + 6$, $t_6 + 2 < t_8 < t_6 + 4$, $t_6 < t_7 < t_5 + 5$, $t_5 + 4 < t_6 < t_5 + 5$, $t_4 < t_5 < t_3 + 6$, $t_3 + 4 < t_4 < t_3 + 5$, $t_1 + 4 < t_3 < t_1 + 6$, $t_1 + 3 < t_2 < t_1 + 4$, $t_0 + 4 < t_1 < t_0 + 6$, $4 < t_0 < 6$.

Išskleista forma, nelygybės atrodo taip:

17 būsena:

	t_0	t_1	t_2	t_3	t_4	t_5		
$0 <$					-1	1		(1)
				-1		1	< 6	(2)
$4 <$				-1	1		< 5	(3)
$4 <$		-1		1			< 6	(4)
$3 <$		-1	1				< 4	(5)
$4 <$	-1	1					< 6	(6)
$4 <$	1						< 6	(7)

44 būsena:

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9		
$4 <$							-1			1	< 6	(1)
$2 <$							-1		1		< 4	(2)
$0 <$							-1	1				(3)
						-1		1			< 5	(4)
$4 <$						-1	1				< 5	(5)
$0 <$					-1	1						(6)
				-1		1					< 6	(7)
$4 <$				-1	1						< 5	(8)
$4 <$		-1		1							< 6	(9)
$3 <$		-1	1								< 4	(10)
$4 <$	-1	1									< 6	(11)
$4 <$	1										< 6	(12)

Pagal 2.2 skyriuje pateiktą algoritmą, atmetame nereikalingas lygtis. Gauname, jog t_5 laiko momentą 17-oje būsenoje išreiškia $\{1,2,3,6,7\}$ nelygybių aibė, o t_9 laiko momentą 44-oje – $\{1,5,6,7,8,9,11,12\}$. Laiko momentų skirtumas tarp būsenų – 9 - 5 = 4. Reiškias, 44-os būsenos laiko momentai atitiks 4 laiko vienetais mažesnį laiko momentą lyginant su 17-a būsena. Pagal būsenų struktūrą – būsenos ekvivalenčios, tačiau būtina atlikti ir nelygybių, aprašančių anksčiau įvykusius įvykius, palyginimą.

Kaip matome, 44-ta būsena išreiškiama per t_9 laiko momentą, o 17-ta būsena – per t_5. Spręsdami tiesinio programavimo uždavinį Simplekso metodu, įvertinsime šiuos du laiko momentus. Atlikus skaičiavimus, gavome, kad $24 < t_9 < 35$, ir $12 < t_5 < 18$, t.y. t_9 laiko momentą ribojantis intervalas 35 – 14 = 15, o t_5 – 18 – 12 = 6. Iš čia daroma išvada, kad įvykius ribojančios aibės nėra vienodos, todėl ir būsenos nėra ekvivalenčios.

.

3 SISTEMOS VEIKSENOS ANALIZĖ

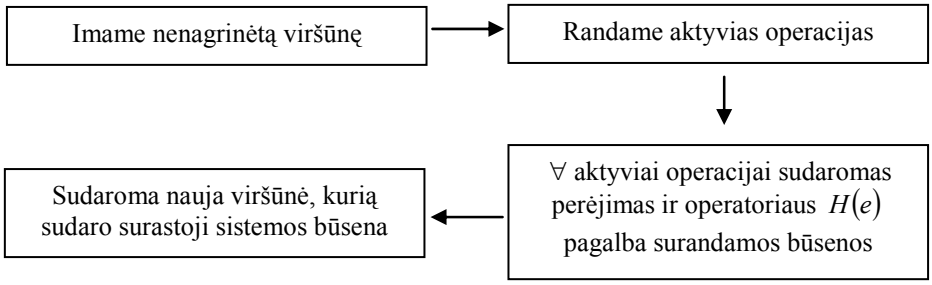
Sistemos veiksenos analizę atliksime sudarydami pasiekiamų būsenų grafą, kuriame būtų surasti visi galimi sistemos įvykiai ir nusakyta, kokiais laiko momentai jie įvyksta.

3.1 Pasiekiamų būsenų grafo sudarymas

Realaus laiko sistemai, specifikuotai PLA, pasiekiamų būsenų grafas sudaromas tokiu būdu [5]:

- Atliekama visų agregatų kompozicija. Atlikus kompoziciją, gaunamas agregatas, kuris neturi nei įėjimo, nei išėjimo signalų. Aprašyme lieka tik vidiniai įvykiai. Agregatinėje specifikacijoje panaikinamas laikas taip: visos tolydinės komponentės $w(e,t_m)$ keičiamos $w_e = 1$, jei operacija aktyvi, ir $w_e = 0$, jei operacija neaktyvi. Kadangi tolydinės komponentės nepriklauso nuo laiko, tai ir valdančios sekos tampa nereikalingos. Įvykdžius tokį pakeitimą sistemos būsena įgyja pavidalą: $z = \{z_v; w_{e_1}, w_{e_2}, \ldots, w_{e_n}\}$.
- Nusakoma pradinių būsenų aibė.
- Apibrėžiama galutinių būsenų aibė.
- Pasiekiamų būsenų grafas susideda iš viršūnių aibės V ir lankų aibės L. Viršūnėje yra apibrėžiama sistemos būsena, o lanką sudaro: pradinė būsena v, įvykis, kuris sukelia perėjimą e, kita būsena v', t. y. (v,e,v'), čia $v' = H(e,v)$.
- Sudaroma nenagrinėjama būsenų aibė V_N, į kurią pradžioje patalpinamos visos pradinės būsenos. Paskui imama būsena $v \in V_N$ ir sudaroma tiek lankų, kiek yra aktyvių operacijų, t. y. gaunama aibė $L = \{v, e, H(e,v) | e \in E'', w(e) = 1\}$, kuri prijungiama prie lankų aibės L. Nenagrinėjamų viršūnių aibė V_N papildoma aibe $\{v' = H(e,v) | e \in E'', w(e) = 1, v' \notin V \cup V_N\}$.

Grafo generavimas baigsis, kai nenagrinėjamų būsenų aibė V_N taps tuščia, t. y. neliks nenagrinėtų viršūnių. Viršūnės analizės struktūrinė schema pateikiama 8 pav.



8 pav. Pasiekiamų būsenų grafo viršūnės analizės struktūrinė schema

Pasiekiamų būsenų grafas leidžia analizuoti šias sistemos savybes:

- pasiekiamumą, t. y. kokiu būdu galima iš pradinės būsenos pasiekti galinę būseną;
- būsenos koordinačių apribojimus, t. y. reikia nustatyti, ar būsenos koordinatės neišeina už iš anksto nustatytų ribų;
- tolydinių koordinačių reikalingumą, t. y. reikia nustatyti, ar specifikacijoje nėra aprašytų tolydinių komponenčių, kurios nė karto negeneruoja vidinių įvykių;
- aklaviečių paieška, t. y. reikia nustatyti, ar nėra būsenų, kuriose nėra aktyvių operacijų;
- ciklų paieška, reikia nustatyti, ar nepatenkame į uždarą ciklą, kuriame kartojasi tam tikra įvykių seka.

3.2 Pasiekiamų būsenų grafo sudarymo pavyzdys

Šiame skyriuje, pagal 2 skyriuje pateiktą metodiką, bus nagrinėjama konkreti viršūnė, skaičiuojami galimi būsenos perėjimai.

Tarkime, kad analizuojama būsena – $S(t) = (0; 0; (t_4 + 3; t_4 + 5); (t_4 + 2; t_4 + 4); (t_2 + 3; t_2 + 7); \varnothing; (t_3 + 9; t_3 + 11); \varnothing; R)$.

Operacijų pabaigos laiko momentų aibės yra pateiktos 9 pav.

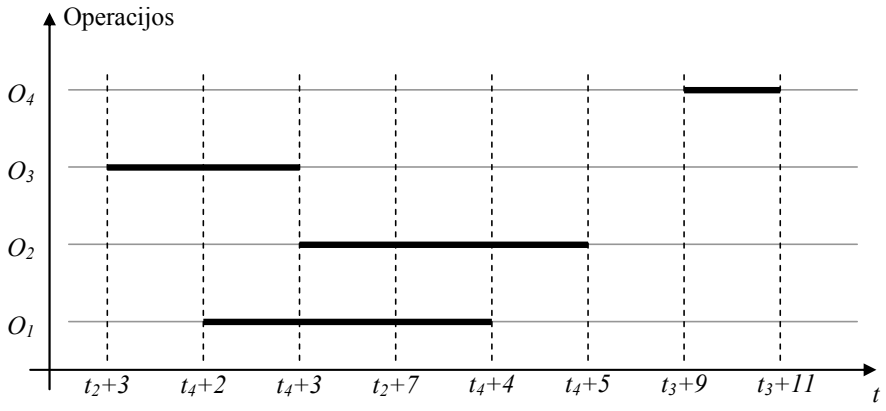
Apribojimų aibė R sudaryta iš šių nelygybių:

$$R = \{t_2 + 3 < t_4 < t_2 + 5; \quad (1)$$

$$t_2 < t_3 < t_1 + 4; \quad (2)$$

$$t_1 + 3 < t_2 < t_1 + 4; \quad (3)$$

$$t_0 + 3 < t_1 < t_0 + 5;\}. \quad (4)$$



9 pav. Operacijų pabaigos laiko momentų aibės

Būtina pastebėti, kad kai kurių laiko momentų eiliškumą (pvz. $t_4 + 3$ ir $t_2 + 7$ ar pan.) kol kas vienareikšmiškai nustatyti neįmanoma. Būtent dėl šios priežasties privaloma toliau vykdyti išsamią laiko momentų analizę, kad galėtume tiksliai nustatyti 9 pav. vaizduojamų laiko momentų eiliškumą.

Sudarant pasiekiamų būsenų grafą, pirmame žingsnyje reikia nustatyti nagrinėjamą laiko intervalą, t.y. rasti $I = (\alpha, \beta)$, kur α – operacijų pradžios laiko momentų minimumas, o β – operacijų pabaigos laiko momentų minimumas, t.y. $\alpha = \min(t_2 + 3; t_4 + 2; t_4 + 3; t_3 + 9)$, $\beta = \min(t_2 + 7; t_4 + 4; t_4 + 5; t_3 + 11)$.

Operacijų pradžios laiko momentų minimumo α paieška

Atlikdami veiksmus apribojimų aibėje ir tardami, kad $t_0 = 0$, lyginame iš eilės α aibės elementus ieškodami minimumo:

- $t_4 + 2 \,?\, t_2 + 3$: pertvarkę lyginamus laiko momentus gauname $t_4 - t_2 \,?\, 1$. Iš (1) nelygybės atėmę t_2 gauname $3 < t_4 - t_2 < 5$, iš ko seka, kad $t_4 - t_2 > 1$, o taip pat $t_4 + 2 > t_2 + 5$. Gautą mažesnįjį laiko momentą lyginame su kitu aibės laiko momentu.
- $t_4 + 3 \,?\, t_2 + 3$: pagal 1 lemą $t_4 + 2 < t_4 + 3$, nes atskaitos taškų laiko koordinatės sutampa, o pirmojo elemento poslinkis laiko ašyje 1 mažesnis. O taip pat jau žinome, kad $t_4 + 2 > t_2 + 3$, todėl išplaukia, kad $t_4 + 3 > t_4 + 2 > t_2 + 3$, t.y. $t_4 + 3 > t_2 + 3$.
- $t_3 + 9 \,?\, t_2 + 3$: pagal 2 lemą $t_3 + 9 > t_2 + 3$, nes pirmojo laiko momento atskaitos taško laiko koordinatės indeksas didesnis už antrojo, bei 4 vienetais didesnis poslinkis laiko ašyje.

Tokiu būdu, radome α aibės minimumą – $\alpha = t_2 + 3$.

Operacijų pabaigos laiko momentų minimumo β paieška

Analogiškai ieškome β aibės minimumo, lygindami iš eilės ją sudarančius elementus:

- $t_4 + 4 \,?\, t_2 + 7$: pertvarkę lyginamus laiko momentus gauname $t_4 - t_2 \,?\, 3$. Iš (1) nelygybės atėmę t_2 gauname $3 < t_4 - t_2 < 5$, iš ko seka, kad $t_4 - t_2 > 3$, o taip pat $t_4 + 4 > t_2 + 7$.
- $t_4 + 5 \,?\, t_2 + 7$: pagal 1 lemą $t_4 + 5 < t_4 + 4$, nes atskaitos taškų laiko koordinatės sutampa, o pirmojo elemento poslinkis laiko ašyje 1 mažesnis. O taip pat jau žinome, kad $t_4 + 4 > t_2 + 7$, todėl išplaukia, kad $t_4 + 5 > t_4 + 4 > t_2 + 7$, t.y. $t_4 + 5 > t_2 + 7$.
- $t_3 + 9 \,?\, t_2 + 7$: pagal 2 lemą $t_3 + 9 > t_2 + 7$, nes pirmojo laiko momento atskaitos taško laiko koordinatės indeksas didesnis už antrojo, bei 4 vienetais didesnis poslinkis laiko ašyje.

Taigi nagrinėjama laiko sritis apibrėžiama intervalu $\alpha, \beta = (t_2 + 3; t_2 + 7)$.

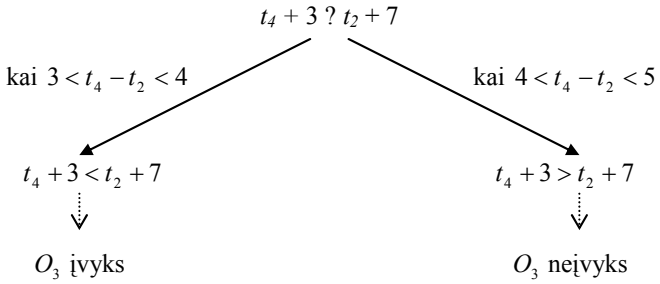
Galimų būsenos perėjimo operacijų analizė

Skaičiuojant galimus perėjimus, reikia išskirti operacijas, kurių laiko momentai patenka į rastą intervalą. Tam lyginame laiko momentą β su visų operacijų pradžios laiko momentais. Jei β didesnis už lyginamos operacijos pradžios laiko momentą, atitinkama operacija gali įvykti analizuojamame laiko intervale.

Tikriname, ar gali įvykti operacija O_1: kadangi nagrinėjamas intervalas $\alpha, \beta = (t_2 + 3; t_2 + 7)$, akivaizdu, kad operacija O_1 ir su juo susijęs įvykis e_B gali įvykti visame intervale $(t_2 + 3; t_2 + 7)$.

Tikriname, ar gali įvykti operacija O_2: lyginame laiko momentus $t_4 + 2$ ir $t_2 + 7$. Anksčiau gavome $t_4 - t_2$ įvertį $-3 < t_4 - t_2 < 5$, iš kurio seka, kad $t_4 + 4 < t_2 + 7$. Vadinasi, operacija O_2 ir su ja susijęs įvykis e_B gali įvykti laiko intervale $(t_2 + 3; t_2 + 7)$.

Tikriname, ar gali įvykti operacija O_3: lyginame laiko momentus $t_4 + 3$ ir $t_2 + 7$. Šiuos laiko momentus, pagal turimą $t_4 - t_2$ įvertį, neįmanoma palyginti vienareikšmiškai, todėl $t_4 - t_2$ įvertis dalinamas į du intervalus ir nagrinėjami du skirtingi atvejai: kai $3 < t_4 - t_2 < 4$ ir $4 < t_4 - t_2 < 5$.

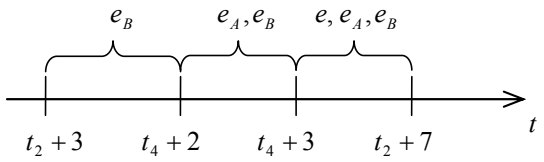


Taigi galimi du naujų būsenų apribojimų atvejai, reiškias ir galimi būsenų perėjimai turi būti nagrinėjami abiem galimais atvejais.

Operacija O_4 ir su ja susijęs įvykis e_D nagrinėjamame laiko intervale įvykti negali, nes pagal 2 lemą $\beta = t_2 + 7 < t_3 + 9$.

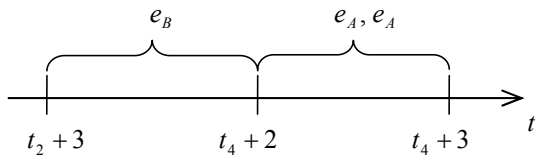
Kitame etape nagrinėjame aukščiau gautus apribojimų aibės papildymus. Vienu atveju naujos sistemos būsenos bus apribotos $R_n = R \cup \{t_2 + 3 < t_5 < t_2 + 7; 3 < t_4 - t_2 < 4\}$, kitu atveju $R_n = R \cup \{t_2 + 3 < t_5 < t_2 + 7; 4 < t_4 - t_2 < 5\}$.

Pirmuoju atveju gali įvykti trys operacijos: O_1, O_2, O_3, su kuriomis atitinkamai susiję įvykiai e_B, e_A ir e. Galimi operacijų įvykimo pabaigos laiko intervalai pavaizduoti 10 paveikslėlyje.



10 pav. Pirmuoju atveju galimi operacijų įvykimo pabaigos laiko intervalai

Antruoju atveju gali įvykti dvi operacijos: O_1, O_2, su kuriomis atitinkamai susiję įvykiai e_B, e_A ir pabaigos laiko intervalai pavaizduoti 11 paveikslėlyje.



11 pav. Antruoju atveju galimi operacijų įvykimo pabaigos laiko intervalai

Perėjimo operatorių generuojamų naujų būsenų skaičiavimas

Nagrinėkime pirmąjį apribojimų aibės atvejį laiko intervale $(t_2+3; t_4+2)$. Kaip matome 10 pav., šiame intervale naują būseną gali sužadinti tik vienas įvykis – e_B. Reiškias, naujoji būsena bus tokia:

$l_{S,S_1} = (e_B, t_5 \in (t_2+3, t_4+2))$

$S_1: (0;0;(t_4+3, t_4+5);(t_4+2, t_4+4);\varnothing;\varnothing;(t_3+9, t_3+11);(t_5+4, t_5+6);R_{S_1})$

čia $R_{S_1} = R \cup \{3 < t_4 - t_2 < 4;\ t_2+3 < t_5 < t_4+2\}$.

Antrajame laiko intervale $(t_4+2; t_4+3)$ aktyvios dvi operacijos – O_1, O_2. Akivaizdu, kad viena iš jų negali įvykti tol, kol neįvyko antroji. Reiškias, skaičiuojant atitinkamos operacijos sužadinamą perėjimą, būtina apriboti antrosios aktyvios operacijos pradžios laiko momentą:

$l_{S,S_2} = (e_A, t_5 \in (t_4+2, t_4+3))$

$S_2: (0;0;(t_4+3, t_4+5);(t_5, t_4+4);\varnothing;\varnothing;(t_3+9, t_3+11);(t_5+4, t_5+6);R_{S_2})$

čia $R_{S_2} = R \cup \{3 < t_4 - t_2 < 4;\ t_4+3 < t_5 < t_4+2\}$;

$l_{S,S_3} = (e_B, t_5 \in (t_4+2, t_4+3))$

$S_3: (0;0;(t_4+3, t_4+5);\varnothing;(t_5, t_2+7);(t_5+14, t_5+16);(t_3+9, t_3+11);\varnothing;R_{S_3})$

čia $R_{S_3} = R \cup \{3 < t_4 - t_2 < 4;\ t_4+2 < t_5 < t_4+3\}$.

Trečiajame laiko intervale $(t_4+3; t_2+7)$ aktyvios trys operacijos – O_1, O_2, O_3. Analogiškai, kaip ir prieš tai nagrinėtame intervale, įvykus vienam iš šių operacijų įvykių, būtina apriboti likusių aktyvių operacijų pradžios laiko momentus. Taigi gauname:

$l_{S,S_4} = (e, t_5 \in (t_4+3, t_2+7))$

$S_4: (0;0;(t_5+3, t_5+5);(t_5, t_4+4);(t_5, t_2+7);\varnothing;(t_3+9, t_3+11);\varnothing;R_{S_4})$

čia $R_{S_4} = R \cup \{3 < t_4 - t_2 < 4;\ t_4+3 < t_5 < t_2+7\}$;

$l_{S,S_5} = (e_A, t_5 \in (t_4+3, t_2+7))$

$S_5: \left(0;0;(t_5, t_4+5);\varnothing;(t_5, t_2+7);(t_5+14, t_5+16);(t_3+9, t_3+11);\varnothing;R_{S_4}\right)$

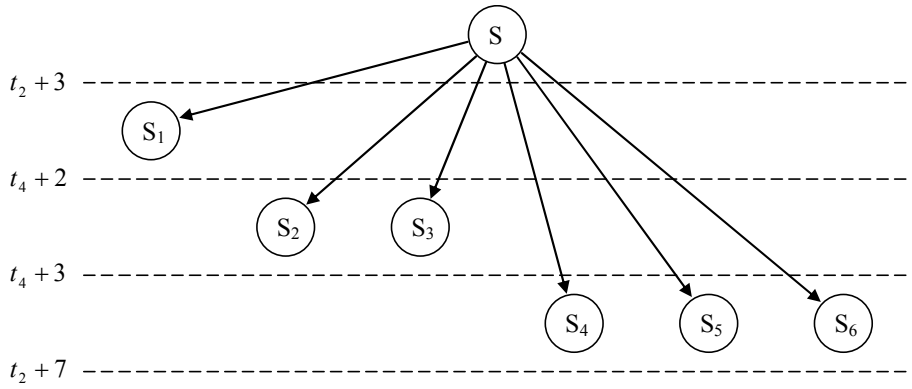
čia $R_{S_5} = R \cup \{3 < t_4 - t_2 < 4;\ t_4 + 3 < t_5 < t_2 + 7\}$;

$l_{S,S_6} = (e_B, t_5 \in (t_4+3, t_2+7))$

$S_6: \left(0;0;(t_5, t_4+5);(t_5, t_4+4);\varnothing;\varnothing;(t_3+9, t_3+11);(t_5+4, t_5+6);R_{S_4}\right)$

čia $R_{S_6} = R \cup \{3 < t_4 - t_2 < 4;\ t_4 + 3 < t_5 < t_2 + 7\}$.

Galimi perėjimai iš būsenos S, su pirmuoju apribojimų aibės atveju, kur $3 < t_4 - t_2 < 4$, pavaizduoti 12 paveikslėlyje.



12 pav. Galimų perėjimų iš būsenos S grafo poaibis

Lygiai taip nagrinėkime antrąjį apribojimų aibės atvejį, kai $4 < t_4 - t_2 < 5$. Pirmasis nagrinėjamas laiko intervalas – $(t_2+3;\ t_4+2)$. Kaip matome 10 pav., šiame intervale naują būseną gali sužadinti tik vienas įvykis – e_B. Reiškias, naujoji būsena bus tokia:

$l_{S,S_7} = (e_B, t_5 \in (t_2+3, t_4+2))$

$S_7: \left(0;0;(t_4+3, t_4+5);(t_4+2, t_4+4);\varnothing;\varnothing;(t_3+9, t_3+11);(t_5+4, t_5+6);R_{S_8}\right)$

čia $R_{S_8} = R \cup \{4 < t_4 - t_2 < 5;\ t_2 + 3 < t_5 < t_4 + 2\}$.

Antrajame nagrinėjamo laiko intervale $(t_4+2;\ t_4+3)$ aktyvios dvi operacijos – O_1, O_2:

$l_{S,S_8} = (e_A, t_5 \in (t_4+2, t_4+3))$

$S_8: \left(0;0;(t_4+3, t_4+5);(t_5, t_4+4);\varnothing;\varnothing;(t_3+9, t_3+11);(t_5+4, t_5+6);R_{S_8}\right)$

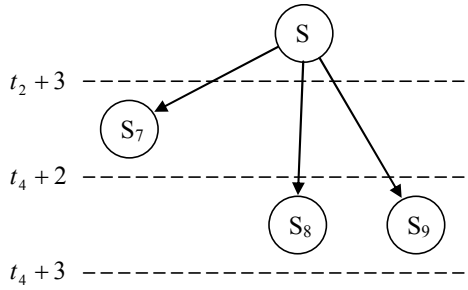
čia $R_{S_8} = R \cup \{4 < t_4 - t_2 < 5;\ t_4 + 2 < t_5 < t_4 + 3\}$;

$l_{S,S_9} = (e_B, t_5 \in (t_4+2, t_4+3))$

$S_9: \left(0;0;(t_4+3, t_4+5);\varnothing;(t_5, t_2+7);(t_5+14, t_5+16);(t_3+9, t_3+11);\varnothing;R_{S_9}\right)$

čia $R_{S_9} = R \cup \{4 < t_4 - t_2 < 5;\ t_4 + 2 < t_5 < t_4 + 3\}$.

Galimi perėjimai iš būsenos S, su antruoju apribojimų aibės atveju, kur $4 < t_4 - t_2 < 5$, pavaizduoti 13 paveikslėlyje.



13 pav. Galimų perėjimų iš būsenos S grafo poaibis

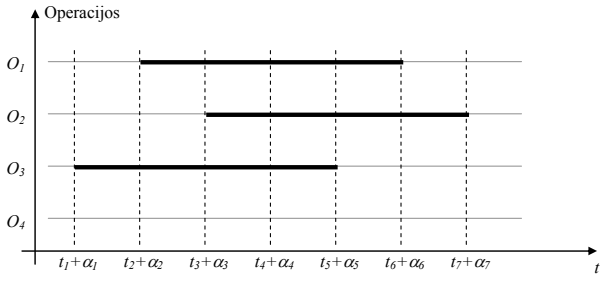
Apjungę 12 ir 13 paveikslėliuose pateiktus sistemos būsenų grafo poaibius, gautume visiškai išnagrinėtą S būsenos perėjimų grafą. Tolesniuose analizės etapuose reikėtų imti visas naujai sugeneruotas būsenas $(S_1, S_2, \dots S_8)$ iš eilės ir, atsižvelgiant į sudarytus apribojimų aibės papildymus, jas išanalizuoti.

3.3 Būsenos skaičiavimo atskiras atvejis

Sistemos būsenos nagrinėjimo metu gali susidaryti tokia situacija, kad aktyvios operacijos sužadinamas įvykis prasidės ir baigsis anksčiau nei prieš tai buvusių aktyvių operacijų pabaigos laiko momentas. Tokiu būdu, toliau analizuojant sistemą toks įvykis būtų tariamai išanalizuotas, o sistemos veiksena apibrėžiama netiksliai. Todėl atliekant sistemos analizę būtina atlikti patikrinimą, ar generuojamas naujas įvykis nėra būtent toks; esant tokiam atvejui, reikalingas patikslinimų įvedimas. Pamėginkime detaliau išanalizuoti tokią situaciją, tokio įvykio atsiradimo aplinkybes ir veiksmus, kurių reikia imtis tokiu atveju.

Tarkime, kad turime būseną T_1:

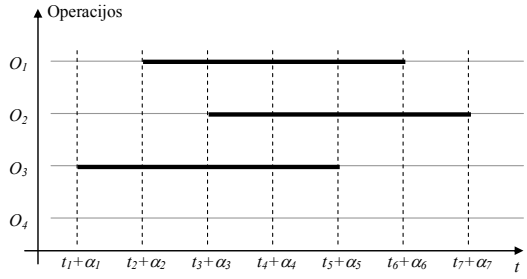
$T_1 = (0;0;(t_2+\alpha_2, t_6+\alpha_6);(t_3+\alpha_3, t_7+\alpha_7);(t_1+\alpha_1, t_5+\alpha_5);\varnothing;\varnothing;\varnothing;R_1)$. Grafiškai ši būsena pavaizduota 14 paveikslėlyje.



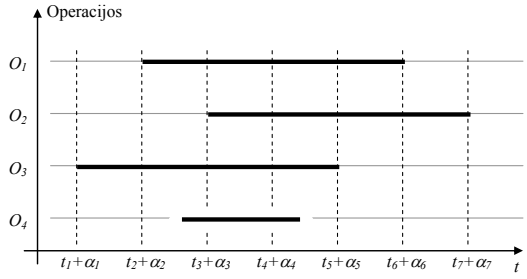
14 pav. T_1 būsenos grafinis atvaizdas

Kaip minėta anksčiau, būsenos analizė pradedama įvykių pradžios ir pabaigos laiko momentų minimumo paieška. Tarkime, kad laiko momentų eiliškumas toks, koks pateiktas 14 pav. Taigi, šiuo atveju, $\alpha = \min(t_2+\alpha_2, t_3+\alpha_3, t_1+\alpha_1) = t_1+\alpha_1$ ir atitinkamai

$\beta = \min(t_6 + \alpha_6, t_7 + \alpha_7, t_5 + \alpha_5) = t_5 + \alpha_5$. Dėl to nagrinėjamas intervalas – $t_8 \in (t_1 + \alpha_1; t_5 + \alpha_5)$ (15 pav.).



15 pav. Nagrinėjamas būsenos T_1 laiko intervalas



16 pav. Įvykio *e* generuojama operacija O_4

Aktyvios operacijos – O_1, O_2, O_3. Nagrinėkime tik vieną operaciją iš galimų – O_1 ir tarkime, kad ši operacija susijusi su įvykiu e. Taip pat, įvykis e generuoja naują operaciją O_4, kuri įvyks laiko intervale $(t_8; t_8 + \alpha_8)$. Čia, jei $t_8 > t_2 + \alpha_2$ bei $t_8 + \alpha_8 < \beta = t_5 + \alpha_5$, gauname, kad operacija O_4 būtinai baigsis anksčiau nei nagrinėjamas laiko intervalas (16 pav.). To pasekoje, kitame analizuojamame intervale, įvykio e sugeneruota būsena O_4 nebegalės būti aktyvi. Tai veda prie klaidingos sistemos veiksena analizės.

Taigi sugeneravus naują operaciją, visada būtina patikrinti, ar šios operacijos įvykimo intervalo pabaigos laiko momentas nėra mažesnis už analizuojamo laiko intervalo pabaigą β. Susidarius tokiai situacijai, nagrinėjamą intervalą reikia sumažinti iki naujosios operacijos pradžios laiko momento.

Pratęsiant aukščiau nagrinėtos būsenos analizę, tiriamas laiko intervalas siaurinamas iki $t_9 \in (t_1 + \alpha_1; t_8)$, o nagrinėta operacija O_4 keičia sistemos būseną į $T_2 = (0;0;(t_8, t_6 + \alpha_6);(t_3 + \alpha_3, t_7 + \alpha_7);\varnothing;(t_8, t_8 + \alpha_8);\varnothing;\varnothing)$. Kitame analizuojamame laiko intervale, operacija O_4 bus aktyvi ir „nepasimes“ tolimesnės analizės metu.

4 PASIEKIAMŲ BŪSENŲ GRAFO SUDARYMO ALGORITMAI

Šiame skyriuje pateikiami darbe aprašytų metodų algoritmų žodiniai aprašai, bei jų realizacijų pseudo kodai. Aprašomi minimumo paieškos, minimalios apribojimų aibės srities radimo bei ekvivalenčių būsenų nustatymo algoritmai.

4.1 Apibendrintas minimumo paieškos algoritmas

Šiame darbe sprendžiama problema praktiškai susiveda į vieną užduotį: laiko momentų aibės elementų rikiavimas didėjimo tvarka. Kitaip tariant, mūsų užduotis- sugebėti įvertinti du laiko momentus, kiekvienas kurių aprašytas dvejomis dedamosiomis: atitinkamo atskaitos taško laiko koordinate t_i bei jo poslinkiu laiko ašyje α.

Taigi apibendrintas minimalios laiko koordinatės paieškos algoritmas aibėje yra toks:

1. Aiškių laiko momentų išmetimas iš aibės.
2. Jei laiko momentų aibėje yra daugiau kaip vienas elementas, pasirenkame pirmuosius du neanalizuotus elementus ir sprendžiam optimizavimo uždavinį:
    - surandam įmanomai didesnį pradinį atskaitos tašką, bendrą visiems aibės elementams;
    - pagal turimus aibės elementų apribojimus sudarome nelygybių sistemą;
    - sudarom tikslo funkciją;
    - simplekso metodu sprendžiame tiesinio programavimo uždavinį, minimizuojame ir maksimizuojame tikslo funkciją.
3. Pagal antrame žingsnyje gautus rezultatus palyginame du tiriamus laiko momentus.
4. Jei laiko momentų aibėje dar yra neanalizuotų elementų- kartojame antrą žingsnį, papildydami nelygybių sistemą prieš tai padarytomis prielaidomis (apribojimais).

Tam, kad supaprastintume minimalios laiko koordinatės aibėje paieškos procedūrą, išskaidysime ją į keletą smulkesnių algoritmų ir pateiksime šių algoritmų aprašymus ir pseudo kodus.

4.1.1 Aiškių laiko momentų paieška

Laiko momentų įvertinimas, sprendžiant tiesinio programavimo uždavinį- daug laiko resursų reikalaujanti procedūra, todėl efektyviam sprendimui labai svarbus aibės elementų skaičius.

Tačiau dažnai, laiko momentų įvertinimui, nebūtina spręsti tiesinio programavimo uždavinį. Atlikę pora laiko momentų dedamųjų palyginimo veiksmų, galime iš karto įvertinti jų eiliškumą laiko ašyje, išvengdami sudėtingų skaičiavimų. Momentai, kaip minėta anksčiau, aprašomi dvejomis dedamosiomis: atskaitos taško laiko koordinate t_i bei jo poslinkiu laiko ašyje α.

Trivialiai didesni laiko momentai pasižymi šiomis savybėmis:

- atskaitos taško laiko koordinatės indeksai sutampa, arba yra didesni, ir
- poslinkis laiko ašyje sutampa arba yra didesnis, ir
- laiko momentai nėra tapačiai lygūs.

Jei lyginant laiko momentus tenkinamos šios sąlygos, tariame, jog laiko momentai trivialiai didesni ir toliau jų nenagrinėjame t.y. šaliname iš aibės.

Aiškių laiko momentų paieškos algoritmo pseudo kodas. Pagal aprašytą aiškių laiko momentų paieškos algoritmą, pateikiamas jo realizacijos pseudo kodą.

```
1: dim a[] ← set (t, α)
2: for i ← 1 to a.length
3:  for j ← 1 to a.length
4:      if (a[i].t >= a[j].t) & (a[i]. α >= a[j]. α) & (a[i] != a[j])
5:            remove a[i] from a
6:      if (a[i].t <= a[j].t) & (a[i]. α <= a[j]. α) & (a[i] != a[j])
7:            remove a[j] from a
8: return a[]
```

4.1.2 Dviejų laiko momentų palyginimas

Vienas svarbiausių ir dažniausiai atliekamų veiksmų, bandant įvertinti realiojo laiko sistemos veikseną – dviejų laiko momentų palyginimas. Šiam uždaviniui spręsti sudaromas tiesinio programavimo uždavinys, kuris sprendžiamas Simplekso metodu. Uždavinio nelygybių sistemą sudaro būseną aprašančių apribojimų aibė, o tikslo funkcija – nagrinėjamų laiko momentų skirtumas. Išsprendus sudarytą tiesinio programavimo uždavinį, gaunamas šių dviejų laiko momentų skirtumo minimumo ir maksimumo įvertis, kuris, pagal 2.1.1 skyriuje pateiktą metodiką lyginamas su nagrinėjamų laiko momentų atitinkamų poslinkių laiko ašyje skirtumu. Remiantis 1 teorema, nustatoma, kuris laiko momentas didesnis.

Dviejų laiko momentų palyginimo pseudo kodas. Pagal aprašytą laiko momentų palyginimo algoritmą, pateikiamas realizacijos pseudo kodas. Jame naudojama *Simplex* procedūra atlieka tiesinio programavimo uždavinio sprendimą Simplekso metodu.

```
 1: dim tm,αm ← (t_m,α_m)
 2: dim tn,αn ← (t_n,α_n)
 3: tf ← t_m - t_n                  // tikslo funkcija
 4: aa ← a1<t1<b1, a2<t2<b2, ...    // apribojimų aibė
 5: c1 ← simplex(tf, aa, min)      // tikslo f-jos minimumas
 6: c2 ← simplex(tf, aa, max)      // tikslo f-jos maksimumas
 7: if (αn - αm < c1)
 8:       return (t_m,α_m > t_n,α_n)
 9: if (αn - αm > c2)
10:       return (t_m,α_m < t_n,α_n)
11: if (c1 < αn - αm < c2)
12:       return (t_m,α_m < t_n,α_n, c1<tn-tm<αn-αm; t_m,α_m > t_n,α_n, αn-αm <tn-tm<c2)
```

4.2 Minimalios apribojimų srities radimo algoritmas

Didėjant sistemos funkcionavimo trajektorijų medžiui, kiekviena nauja būsena aprašoma vis didesne apribojimų aibe, t.y. apribojimų aibę sudaro vis daugiau nelygybių. Todėl tolesnė būsenų analizė tampa daug resursų reikalaujančiu uždaviniu. 2.2 skyriuje teigiama, kad ne visos apribojimų aibės nelygybės turi įtakos tolesnei būsenos elgsenai, reiškias jas galima atmesti. Nereikalingų nelygybių paieškos uždaviniui spręsti siūlomas algoritmas atrodo taip: sudaroma pradinė aibė K, kurioje saugomi analizuojamų laiko momentų atskaitos taškų indeksai i, t.y. $K = \{i_1, i_2, \ldots, i_n\}$, čia $i \in \mathsf{N}$, o pasikartojantys aibės elementai atmetami. Pradedamas vykdyti ciklas: surandamas aibės K maksimumas $n = \max K$; apribojimų aibėje išrenkami laiko momentai, kuriais išreiškiamas laiko momentas t_n; šių išrinktų momentų indeksais papildoma aibė K ir pašalinamas iš jos elementas n. Taip ciklas kartojamas tol, kol aibėje K lieka vienas elementas.

Minimalios apribojimų srities radimo algoritmo pseudo kodas. Pagal aprašytą minimalios apribojimų srities radimo algoritmą, pateikiamas jo pseudo kodas.

```
1:  dim K[] ← set (t_i, t_j, t_k,...)          // laiko momentų masyvas
2:  dim R[] ← set ()                           // reikšmingų nelygybių indeksai
3:  while length(K) > 0
4:       Kmax ← max(K)
5:       for i=1 to length(RR)
6:            if RR[i] ∈ t_Kmax
7:               add i to R
8:  return R
```

Algoritmo vykdymo rezultatas – masyvas, kurį sudaro reikalingų būseną apribojančių nelygybių eilės numeriai.

4.3 Ekvivalenčių būsenų nustatymo algoritmas

Atliekant realiojo laiko sistemos veiksenos analizę, gaunami atitinkamos būsenos galimi perėjimai tam tikrais laiko momentais. Sistemai funkcionuojant, būsena gali pereiti į tokią būseną, kuri sistemos funkcionavimo istorijoje jau kartojasi. Tolimesnis funkcionavimas iš tokių būsenų visiškai vienodas, o būsenų dedamąsias ir apribojimus aprašančių laiko momentų skirtumas – pastovus dydis. Šios būsenos yra ekvivalenčios, o grafe perėjimas į ekvivalenčią būseną vaizduojamas lanku.

Nustatant dviejų būsenų ekvivalentumą, lyginamos kūnus aprašančios nelygybės. Pirmiausia atmetamos nelygybės, neturinčios įtakos kitų nelygybių atžvilgiu. Likusios nelygybių aibės lyginamos tarpusavyje, vienodos išbraukiamos iš abiejų aibių. Jei po palyginimo abi aibės tampa tuščios – lyginamos būsenos ekvivalenčios. Jei bent viena aibė netuščia, t.y. lieka nelygybių – lyginamos būsenos nėra ekvivalenčios.

```
 1:  R1[] ← (amj<tm-tn, tm-tn<bnj,...)        // pirmos būsenos apribojimų aibė
 2:  R2[] ← (akn<tk-tl, tk-tl<bln,...)        // antros būsenos apribojimų aibė
 3:  // šalinamos įtakos neturinčios nelygybės
 4:  for i=1 to length(R1)
 5:       for j=i to length(R1)
 6:            if aj < ai
 7:                 remove j from R1
 8:            if bj > bi
 9:               remove j from R1
10:  for i=1 to length(R2)
11:       for j=i to length(R2)
12:            if aj < ai
13:                 remove j from R2
14:            if bj > bi
15:                 remove j from R2
16:  // aibių R1 ir R2 palyginimas
17:  for i=1 to length(R1)
18:       for j=i to length(R2)
19:            if ((ami = anj) and (tmi = tkj) and (tni = tlj)) or
20:            ((bi = bj) and (tmi = tkj) and (tni = tlj))
21:                 remove i from R1
22:                 Remove j from R2
23:  if length(R1)>0 or length(R2)>0
24:       return 0
25:  else
26:       return 1
```

4.4 Apibendrintas būsenos analizės algoritmas

Ketvirtajame skyriuje pateiktų algoritmų užtenka pilnos būsenos analizės pagrindiniams žingsniams atlikti.

Atliekant pilną būsenos analizę, pirmiausiai atmetamos įtakos neturinčios būseną ribojančios nelygybės (pagal 4.2 skyriuje pateiktą algoritmą). Sekančiame analizės etape, randami α ir β koeficientai, nusakantys nagrinėjamą laiko intervalą. Šiame etape susidariusio uždavinio sprendimo metodas aprašytas 4.1 skyriuje, kur pateiktas apibendrintas dviejų laiko momentų palyginimo algoritmas. Taikant šį palyginimo algoritmą visiems būseną sudarantiems laiko momentams, bus gaunami įvykių pradžios ir pabaigos laiko momentų minimumai.

Suradus α ir β koeficientus, reikia patikrinti, kurios aktyvios būsenos operacijos gali įvykti laiko intervale (α , β). Šiam tikslui, lyginami aktyvių operacijų pradžių laiko momentai ir koeficientas β. Jei pradžios laiko momentas didesnis už β – operacija nagrinėjamame intervale įvykti negali. Priešingu atveju operacija įvykti gali ir toks atvejis privalo būti išnagrinėtas.

Nagrinėjamas intervalas (α , β) dalomas į intervalus pagal galinčių šiame intervale įvykti įvykių pradžių laiko momentus. Kiekvienas šių intervalų turi būti išnagrinėtas atsižvelgiant į to intervalo pradžios bei pabaigos laiko momentus ir įvykius, kurie gali nagrinėjamame intervale įvykti. Šiame etape gaunamos naujos sistemos būsenos ir apribojimų aibės papildymai.

Naujos būsenos turi būti lyginamos su kitomis sistemos funkcionavimo trajektorijos būsenomis. Jei ekvivalenčios būsenos nėra – sistemos funkcionavimo grafe būsena atvaizduojama kaip nauja šaka. Priešingu atveju – lanku į ekvivalenčią būseną. Ekvivalenčių būsenų nustatymo algoritmas pateiktas 4.3 skyriuje.

Jei sistemos veiksena grafe baigiasi būsena, neturinčia sau ekvivalenčios anksčiau buvusios būsenos, sistemos veiksenos (funkcionavimo) analizę reikia tęsti. Sistema nagrinėjama tol, kol visos funkcionavimo trajektorijų pabaigos būsenos turi sau ekvivalenčią būseną. Tokiu atveju galima laikyti, kad sistemos veiksena išnagrinėta pilnai.

DARBO REZULTATAI IR IŠVADOS

Sukurta ir ištirta realiojo laiko sistemų, aprašytų agregatais, pasiekiamų būsenų grafo, kuris įvertina įvykių įvykimo laikus, sudarymo metodologija.

Parodyta, kad sekančio įvykio įvykimo laiko momento nustatymui, pakanka lyginti dviejų operacijų pabaigos laiko momentus.

Parodyta, kad dviejų operacijų pabaigos momentų palyginimas gali būti atliktas naudojant Simplekso optimizavimo metodą, kuris nustato skirtumą tarp operacijų įvykimo laiko momentų įvertinant ankstesnę sistemos veikseną.

Sudaryti minimumo paieškos, dviejų laiko momentų palyginimo, minimalios apribojimų aibės bei ekvivalenčių būsenų nustatymo algoritmai.

Pasikartojančių būsenų paieškai veiksenoje (trajektorijų šeimoje) siūloma lyginti ne tik būsenų struktūrą ir nelygybių formą, bet ir jų apribojimų aibes, kurios aprašo anksčiau įvykusius įvykius. Šios apribojimų aibės yra daugiamačiai briaunainiai, kuriuos reikia palyginti norint įsitikinti, ar tolimesnė realiojo laiko sistemos veiksenos sutampa.

Sudaryta metodika gali būti taikoma realiojo laiko sistemų veiksenos analizės programinių priemonių sudarymui.

Sukurtus algoritmus galima pritaikyti realiojo laiko sistemų analizei, kai yra naudojami kito tipo modeliai (pvz. laiko automatai, laiko Petri tinklai, Q-modeliai ir panašiai).

LITERATŪRA

1. RODD M. G., MOTUS L. *Timing analysis of real - time software*. 1994.
2. LAMPORT, L.; LYNCH N. *Distributed computing: modes and methods. Handbook of theoretical computer science*. 1990.
3. ALUR, R.; HENZINGER T. A. *A Really Temporal Logic. Symposium on Foundations of Computer Science*. 1989, p. 164 – 169.
4. SCHNEIDER S. *Timed CSP: Theory and Practice. – G.* 1992. p. 640 – 675
5. MAKACKAS, D. *Agregatinio metodo taikymas realiojo laiko sistemų funkcionavimo charakteristikoms įvertinti*: daktaro disertacijos darbas. KTU, informatikos fakultetas. [Kaunas], 2003.
6. AWERILL, M.; KELTON, D. *Simulation Modeling and Analysis*. Third Edition. New York, 2000. 17 p. ISBN 0070-59-292-6.
7. BOCCARA, N. *Modeling Complex Systems*. New York, 2004. 38 p. ISBN 0-387-40462-7.
8. LORIN, H.; DEITEL H. M. *Operating Systems*. Third Edition. Upper Saddle River. 2004. ISBN 0-13-182827-4.
9. SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Operating System Concepts*. 7th Edition, January 2005. ISBN 0-471-69466-5.
10. *FOLDOC – Free On-Line Dictionary Of Computing* [žiūrėta 2006 m. kovas 14 d.], prieiga internete http://foldoc.org/?formalmethods
11. *Wikipedia, the free encyclopedia* [žiūrėta 2006 vasaris 17 d.], prieiga internete http://en.wikipedia.org/wiki/Formal_methods
12. DUNSTAN, M.; KELSEY, T.; LINTON, S. *Lightweight formal methods for computer algebra systems*. Germany, 1998. ISBN 1-58113-003-3.
13. *The Alloy Analyzer* [žiūrėta 2006 vasaris 17 d.], prieiga internete http://alloy.mit.edu
14. AGERHOLM, S.; LARSEN, P. *A Lightweight Approach to Formal Methods*. Proceedings of the International Workshop on Current Trends in Applied Formal Method: Applied Formal Methods. London, 1998. ISBN 3-540-66462-9.
15. HEWITT, C.; BISHOP, P.; STEIGEI, R. *A Universal Modular Actor Formalism for Artificial Intelligence*. 3rd International Joint Conference on Artificial Intelligence. Standford, California, 1973, pp. 235-245.

16. WANG, K.; PRANEVIČIUS, H. *Application of AI to Production Engineering Nordic-Baltic Summer School '97 Lecturer Notes*. Kaunas, 1997. 275-302 p. ISBN 9986-13-564-8.

17. PRANEVIČIUS, H.; PILKAUSKAS, V.; ir CHMIELIAUSKAS A. *Aggregate approach for specification and analysis of computer network protocols*. Kaunas, 1994. 152 p. ISBN 9986-13-019-0.

18. BUSLENKO, N.; KOVALENKO I. *Lectures on Complex Systems Theory*. Moscow, 1973.

19. PILKAUSKAS, V.; CHMIELIAUSKAS, A. *Automatized system or validation and simulation of computer network protocols PRANAS – 2*. Kaunas, 1991.

20. SINTONEN, L.; PRANEVIČIUS, H. *Simulation of high – speed ring access with slot reuse*. Informacinės technologijos ’97. Konferencijos pranešimų medžiaga. Kaunas, 1997. p. 237 – 244.

21. LEONAVIČIŪTĖ, Ū. *Agregatinių specifikacijų analizė pasiekiamų būsenų metodu:* magistro darbas. Kauno Technologijos Universitetas. [Kaunas], 2003.

“Evaluation of Real Time System Behavior”

SANTRAUKA ANGLŲ KALBA (SUMMARY)

Complexity and variety of systems that are working in real time mode need to be specified regarding all behavior conditions. The correctness of the specification, determines whether implemented system will supply conditions that were set. To ensure that specification of the described real-time system is correct, we have to do verification and validation of the specification.

Traditional verification methods do not assure full real time system inspection. The main drawback, talking about them, is impossibility of system evaluation according time.

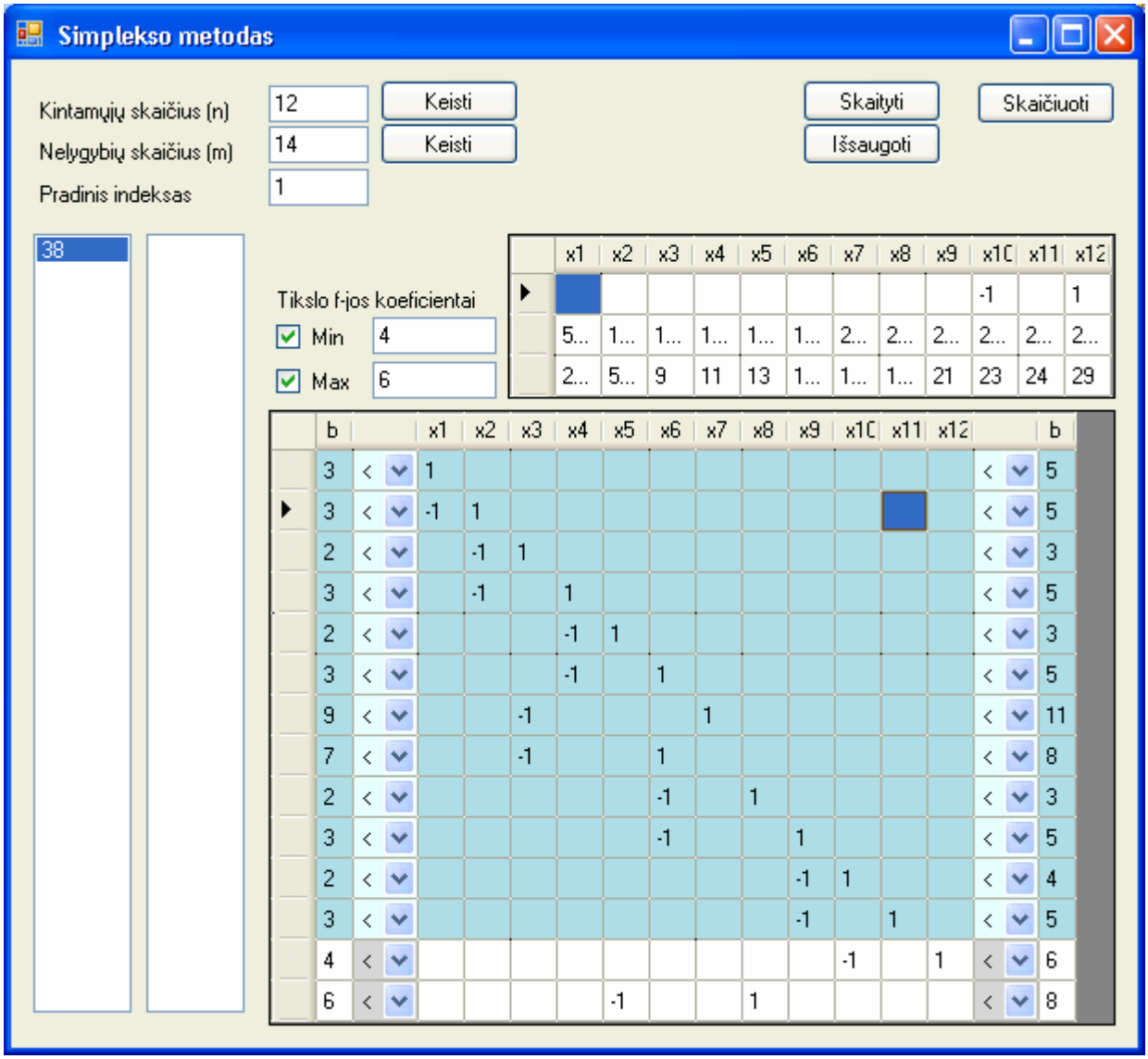
In this paper a reachable states graph and its generating algorithms are described, while two time moments are compared using linear programming, Simplex method. Therefore, method for checking equivalent state in system behavioral pathway was suggested.

Also, theoretical reasoning for creating computer applications, which automates generation of reachable states graph, is given. Reachable states graph fully evaluates real time system behavior.

PRIEDAI

1 priedas. Tiesinio programavimo uždavinio sprendimo Simplekso metodu pavyzdys.

Tiesinio programavimo uždaviniams Simplekso metodu spręsti, buvo naudojama programa „Simplekso metodas“, sukurta pagal Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest ir Clifford Stein „Introduction To Algorithms“ knygoje pateiktą Simplekso metodo algoritmą. Programos vaizdas pateiktas 17 paveikslėlyje.



17 pav. Programos "Simplekso metodas" vartotojo sąsaja

2 priedas. Lanksčios gamybinės sistemos veiksenos ir būsenų trajektorijų grafo fragmentas.

Pradinė būsena

0: $(0;0;(t_0+3,t_0+5);0;0;0;0;0;R_0)$, čia $R_0=\varnothing$

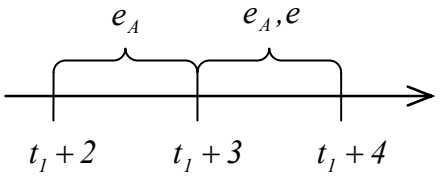
$\alpha=min\{t_0+3\}\ \beta=min\{t_0+5\}\ (\alpha,\beta)=(t_0+3,t_0+5)$

Gali įvykti tik vienas įvykis- e.

Nagrinėjam 0 būseną, pagal perėjimo operatorių $H(e)$

$l_{0,1}=(e,t_1\in(t_0+3,t_0+5))$

1: $(0;0;(t_1+3,t_1+5);(t_1+2,t_1+4);\varnothing;\varnothing;\varnothing;\varnothing;R_1)$, čia $R_1=R_0\cup\{t_0+3<t_1<t_0+5\}$



$\alpha=min\{t_1+2,t_1+3\}\ \beta=min\{t_1+4,t_1+5\}\ (\alpha,\beta)=(t_1+2,t_1+4)$

Per šį laiko tarpą gali įvykti trys įvykiai:

- $e_A,(t_1+2,t_1+3)$
- $e_A,(t_1+3,t_1+4)$
- $e,(t_1+3,t_1+4)$

Nagrinėjam 1 būseną, 1 atvejį, įvyksta įvykis e_A laiko intervale (t_1+2,t_1+3)

$l_{1,2}=(e_A,t_2\in(t_1+2,t_1+3))$

2: $(0;0;(t_1+3,t_1+5);\varnothing;\varnothing;\varnothing;(t_2+9,t_2+11);\varnothing;R_2)$, čia $R_2=R_1\cup\{t_1+2<t_2<t_1+3\}$

Nagrinėjam 1 būseną, 2 atvejį, įvyksta įvykis e_A laiko intervale (t_1+3,t_1+4)

$l_{1,3}=(e_A,t_2\in(t_1+3,t_1+4))$

3: $(0;0;(t_2,t_1+5);\varnothing;\varnothing;\varnothing;(t_2+9,t_2+11);\varnothing;R_3)$, čia $R_3=R_1\cup\{t_1+3<t_2<t_1+4\}$

Nagrinėjam 1 būseną, 3 atvejį, įvyksta įvykis e laiko intervale (t_1+3,t_1+4)

$l_{1,4}=(e,t_2\in(t_1+3,t_1+4))$

4: $(0;0;(t_2+3,t_2+5);(t_2,t_1+4);(t_2+5,t_2+7);\varnothing;\varnothing;\varnothing;R_4)$, čia $R_4=R_1\cup\{t_1+3<t_2<t_1+4\}$

Nagrinėjam 2 būseną:

$\alpha = min\{t_1 + 3, t_2 + 9\}\ \beta = min\{t_1 + 5, t_2 + 11\}\ (\alpha, \beta) = (t_1 + 3, t_1 + 5)$

Intervale $(t_1 + 3, t_1 + 5)$ gali įvykti įvykiai tik e įvykis, todėl pagal perėjimo operatorių $H(e)$ gauname:

$l_{2,5} = (e, t_3 \in (t_1 + 3, t_1 + 5))$

5: $(0;0;(t_3 + 3, t_3 + 5);(t_3 + 2, t_3 + 4);\varnothing;\varnothing;(t_2 + 9, t_2 + 11);\varnothing;R_5)$, čia $R_5 = R_2 \cup \{t_1 + 3 < t_3 < t_1 + 5\}$

Nagrinėjam 3 būseną:

$\alpha = min\{t_2, t_2 + 9\}\ \beta = min\{t_1 + 5, t_2 + 11\}\ (\alpha, \beta) = (t_2, t_1 + 5)$

Įvykis e_D įvyks tik tuo atveju, jei $t_2 + 9 \overset{?}{<} t_1 + 5$. Kadangi nelygybė netenkinama, įvykis neįvyks.

Reiškias, intervale $(t_2, t_1 + 5)$ gali įvykti tik įvykis e, todėl pagal perėjimo operatorių $H(e)$ gauname:

$l_{3,6} = (e, t_3 \in (t_2, t_1 + 5))$

6: $(0;0;(t_3 + 3, t_3 + 5);(t_3 + 2, t_3 + 4);\varnothing;\varnothing;(t_2 + 9, t_2 + 11);\varnothing;R_6)$, čia $R_6 = R_3 \cup \{t_2 < t_3 < t_1 + 5\}$

Nagrinėjam 4 būseną

$\alpha = min\{t_2 + 3, t_2, t_2 + 5\}$

$\beta = min\{t_2 + 5, t_1 + 4, t_2 + 7\}$

Kadangi $t_2 > t_1\ ir\ 5 > 4, tai\ t_2 + 5 > t_1 + 4$

$(\alpha, \beta) = (t_2, t_1 + 4)$

Kokie įvykiai gali įvykti šiame intervale? (žiūrim pagal 4 būseną)

1) $e_A, (t_2, t_1 + 4)$

2) e, jei $t_2 + 3 < t_1 + 4$:

$t_2 + 3 \overset{?}{<} t_1 + 4$, iš čia $t_2 - t_1 \overset{?}{<} 1$

Iš R_4: $t_1 + 3 < t_2 < t_1 + 4$, iš čia $t_2 - t_1 > 3$, reiškias $t_2 + 3 > t_1 + 4$, e neįvyks.

Gali įvykti tik vienas įvykis - e_A.

$l_{4,7} = (e_A, t_3 \in (t_2, t_1 + 4))$

7: $(0;0;(t_2 + 3, t_2 + 5);\varnothing;(t_2 + 5, t_2 + 7);\varnothing;(t_3 + 9, t_3 + 11);\varnothing;R_7)$, čia $R_7 = R_4 \cup \{t_2 < t_3 < t_1 + 4\}$

Nagrinėjam 5 būseną:

$\alpha = min\{t_3 + 3, t_3 + 2, t_2 + 9\}\ \beta = min\{t_3 + 5, t_3 + 4, t_2 + 11\}$

Ieškome minimumo: tikriname nelygybę $t_2 + 9 \overset{?}{<} t_3 + 2$, t.y. $t_3 - t_2 \overset{?}{>} 7$.

Iš R_5: $t_1 + 3 < t_3 < t_1 + 5$

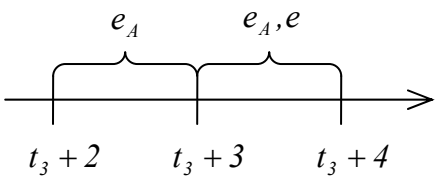
Iš $R_2: t_1 + 2 < t_2 < t_1 + 3$ tapačiai lygu $-t_1 - 3 < -t_2 < -t_1 - 2$. Sudėję nelygybes gauname: $0 < t_3 - t_2 < 3$. Iš šios sąlygos matome, kad tikrinamoji lygybė netenkinama. Vadinasi: $(\alpha, \beta) = (t_3 + 2, t_3 + 4)$.

Ieškome įvykių, kurie gali įvykti šiame laiko intervale.

Įvykis e_D įvyks tik tuo atveju, jei $t_3 + 4 \overset{?}{>} t_2 + 9$ (iš čia $t_3 - t_2 \overset{?}{>} 5$). Kadangi nelygybė netenkinama, įvykis neįvyks.

Reiškias, laiko intervale $(t_3 + 2, t_3 + 4)$ gali įvykti tik trys įvykiai:

- $e_A, (t_3 + 2, t_3 + 3)$
- $e_A, (t_3 + 3, t_3 + 4)$
- $e, (t_3 + 3, t_3 + 4)$



Nagrinėjam 5 būsenos pirmą atvejį- $e_A, (t_3 + 2, t_3 + 3)$

$l_{5,8} = (e_A, t_4 \in (t_3 + 2, t_3 + 3))$

8: $(0;0;(t_3 + 3, t_3 + 5);\varnothing;\varnothing;(t_4 + 14, t_4 + 16);(t_2 + 9, t_2 + 11);\varnothing; R_8)$, čia

$R_8 = R_5 \cup \{t_3 + 2 < t_4 < t_3 + 3\}$

Nagrinėjam 5 būsenos antrą atvejį- $e_A, (t_3 + 3, t_3 + 4)$

$l_{5,9} = (e_A, t_4 \in (t_3 + 3, t_3 + 4))$

9: $(0;0;(t_4, t_3 + 5);\varnothing;\varnothing;(t_4 + 14, t_4 + 16);(t_2 + 9, t_2 + 11);\varnothing; R_9)$, čia $R_9 = R_5 \cup \{t_3 + 3 < t_4 < t_3 + 4\}$

Nagrinėjam 5 būsenos trečią atvejį- $e, (t_3 + 3, t_3 + 4)$

$l_{5,10} = (e, t_4 \in (t_3 + 3, t_3 + 4))$

10: $(0;0;(t_4 + 3, t_4 + 5);(t_4, t_3 + 4);(t_4 + 5, t_4 + 7);\varnothing;(t_2 + 9, t_2 + 11);\varnothing; R_{10})$, čia

$R_{10} = R_5 \cup \{t_3 + 3 < t_4 < t_3 + 4\}$

Nagrinėjam 6 būseną:

$\alpha = min\{t_3 + 3, t_3 + 2, t_2 + 9\}$

$\beta = min\{t_3 + 5, t_3 + 4, t_2 + 11\}$

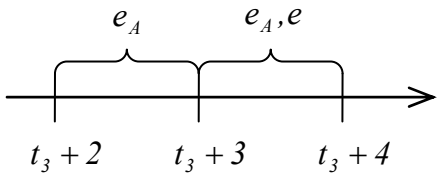
$(\alpha, \beta) = (t_3 + 2, t_3 + 4)$

Ieškome įvykių, kurie gali įvykti šiame laiko intervale.

Įvykis e_D įvyks tik tuo atveju, jei $t_3 + 4 \overset{?}{>} t_2 + 9$ (iš čia $t_3 - t_2 \overset{?}{>} 5$). Nelygybė netenkinama, įvykis neįvyks.

Reiškias, laiko intervale $(t_3 + 2, t_3 + 4)$ gali įvykti tik trys įvykiai:

- $e_A,(t_3+2,t_3+3)$
- $e_A,(t_3+3,t_3+4)$
- $e,(t_3+3,t_3+4)$



Nagrinėjam 6 būsenos pirmą atvejį- $e_A,(t_3+2,t_3+3)$

$l_{6,11}=(e_A,t_4\in(t_3+2,t_3+3))$

11: $(0;0;(t_3+3,t_3+5);\varnothing;\varnothing;(t_4+14,t_4+16);(t_2+9,t_2+11);\varnothing;R_{11})$, čia

$R_{11}=R_6\cup\{t_3+2<t_4<t_3+3\}$

Nagrinėjam 6 būsenos antrą atvejį- $e_A,(t_3+3,t_3+4)$

$l_{6,12}=(e_A,t_4\in(t_3+3,t_3+4))$

12: $(0;0;(t_4,t_3+5);\varnothing;\varnothing;(t_4+14,t_4+16);(t_2+9,t_2+11);\varnothing;R_{12})$, čia

$R_{12}=R_6\cup\{t_3+3<t_4<t_3+4\}$

Nagrinėjam 6 būsenos trečią atvejį- $e,(t_3+3,t_3+4)$

$l_{6,13}=(e,t_4\in(t_3+3,t_3+4))$

13: $(0;0;(t_4+3,t_4+5);(t_4,t_3+4);(t_4+5,t_4+7);\varnothing;(t_2+9,t_2+11);\varnothing;R_{13})$, čia

$R_{13}=R_6\cup\{t_3+3<t_4<t_3+4\}$

Nagrinėjam 7 būseną:

$\alpha=min\{t_2+3,t_2+5,t_3+9\}\ \beta=min\{t_2+5,t_2+7,t_3+11\}\ (\alpha,\beta)=(t_2+3,t_2+5)$

Gali įvykti tik vienas įvykis - e.

$l_{7,14}=(e,t_4\in(t_2+3,t_2+5))$

14: $(0;0;(t_4+3,t_4+5);(t_4+2,t_4+4);(t_2+5,t_2+7);\varnothing;(t_3+9,t_3+11);\varnothing;R_{14})$, čia

$R_{14}=R_7\cup\{t_2+3<t_4<t_2+5\}$

$l_{8,15}=(e,t_5\in(t_3+3,t_3+5))$

15: $(0;0;(t_5+3,t_5+5);(t_5+2,t_5+4);\varnothing;(t_4+14,t_4+16);(t_2+9,t_2+11);\varnothing;R_{15})$, čia

$R_{15}=R_8\cup\{t_3+3<t_5<t_3+5\}$

$l_{9,16}=(e,t_5\in(t_4,t_3+5))$

16: $(0;0;(t_5+3,t_5+5);(t_5+2,t_5+4);\varnothing;(t_4+14,t_4+16);(t_2+9,t_2+11);\varnothing;R_{16})$, čia

$R_{16} = R_9 \cup \{t_4 < t_5 < t_3+5\}$

$l_{10,17} = (e_A, t_5 \in (t_4, t_3+4))$

17: $(0;0;(t_4+3,t_4+5);\varnothing;(t_4+5,t_4+7);(t_5+14,t_5+16);(t_2+9,t_2+11);\varnothing;R_{17})$, čia

$R_{17} = R_{10} \cup \{t_4 < t_5 < t_3+4\}$

$l_{11,18} = (e, t_5 \in (t_3+3, t_3+5))$

18: $(0;0;(t_5+3,t_5+5);(t_5+4,t_5+6);\varnothing;(t_4+14,t_4+16);(t_2+9,t_2+11);\varnothing;R_{18})$, čia

$R_{18} = R_{11} \cup \{t_3+3 < t_5 < t_3+5\}$

$l_{12,19} = (e_D, t_5 \in (t_2+9, t_4))$

19: $(0;0;(t_4,t_3+5);\varnothing;\varnothing;(t_4+14,t_4+16);\varnothing;\varnothing;R_{19})$, čia $R_{19} = R_{12} \cup \{t_2+9 < t_5 < t_4\}$

$l_{12,20} = (e_D, t_5 \in (t_4, t_3+5))$

20: $(0;0;(t_5,t_3+5);\varnothing;\varnothing;(t_4+14,t_4+16);\varnothing;\varnothing;R_{20})$, čia $R_{20} = R_{12} \cup \{t_4 < t_5 < t_3+5\}$

$l_{12,21} = (e, t_5 \in (t_4, t_3+5))$

21: $(0;0;(t_5+3,t_5+5);(t_5+2,t_5+4);\varnothing;(t_4+14,t_4+16);(t_5,t_2+11);\varnothing;R_{21})$, čia

$R_{21} = R_{12} \cup \{t_4 < t_5 < t_3+5\}$

$l_{13,22} = (e_A, t_5 \in (t_4, t_3+4))$

22: $(0;0;(t_4+3,t_4+5);\varnothing;(t_4+5,t_4+7);(t_5+14,t_5+16);(t_2+9,t_2+11);\varnothing;R_{22})$, čia

$R_{22} = R_{13} \cup \{t_4 < t_5 < t_3+4\}$

$l_{14,23} = (e_B, t_5 \in (t_2+5, t_4+2))$

23: $(0;0;(t_4+3,t_4+5);(t_4+2,t_4+4);\varnothing;\varnothing;(t_3+9,t_3+11);(t_5+4,t_5+6);R_{23})$, čia

$R_{23} = R_{14} \cup \{t_2+5 < t_5 < t_4+2 \; ; \; 3 < t_4 - t_2 < 4\}$

$l_{14,24} = (e_B, t_5 \in (t_4+2, t_4+3))$

24: $(0;0;(t_4+3,t_4+5);(t_5,t_4+4);\varnothing;\varnothing;(t_3+9,t_3+11);(t_5+4,t_5+6);R_{24})$, čia

$R_{24} = R_{14} \cup \{t_4+2 < t_5 < t_4+3 \; ; \; 3 < t_4 - t_2 < 4\}$

$l_{14,25} = (e_A, t_5 \in (t_4+2, t_4+3))$

25: $(0;0;(t_4+3,t_4+5);\varnothing;(t_5,t_2+7);(t_5+14,t_5+16);(t_3+9,t_3+11);\varnothing;R_{25})$, čia

$R_{25} = R_{14} \cup \{t_4+2 < t_5 < t_4+3 \; ; \; 3 < t_4 - t_2 < 4\}$

$l_{14,26} = (e_B, t_5 \in (t_4+3, t_2+7))$

26: $(0;0;(t_5,t_4+5);(t_5,t_4+4);\varnothing;\varnothing;(t_3+9,t_3+11);(t_5+4,t_5+6);R_{26})$, čia

$R_{26} = R_{14} \cup \{t_4+3 < t_5 < t_2+7\ ;\ 3 < t_4 - t_2 < 4\}$

$l_{14,27} = (e_A, t_5 \in (t_4+3, t_2+7))$

27: $(0;0;(t_5,t_4+5);\varnothing;(t_5,t_2+7);\varnothing;(t_3+9,t_3+11);(t_5+4,t_5+6);R_{27})$, čia

$R_{27} = R_{14} \cup \{t_4+3 < t_5 < t_2+7\ ;\ 3 < t_4 - t_2 < 4\}$

$l_{14,28} = (e, t_5 \in (t_4+3, t_2+7))$

28: $(0;0;(t_5+3,t_5+5);(t_5,t_4+4);(t_5,t_2+7);\varnothing;(t_3+9,t_3+11);\varnothing;R_{28})$, čia

$R_{28} = R_{14} \cup \{t_4+3 < t_5 < t_2+7\ ;\ 3 < t_4 - t_2 < 4\}$

$l_{14,29} = (e_B, t_5 \in (t_2+5, t_4+2))$

29: $(0;0;(t_4+3,t_4+5);(t_4+2,t_4+4);\varnothing;\varnothing;(t_3+9,t_3+11);(t_5+4,t_5+6);R_{29})$, čia

$R_{29} = R_{14} \cup \{t_2+5 < t_5 < t_4+2\ ;\ 4 < t_4 - t_2 < 5\}$

$l_{14,30} = (e_B, t_5 \in (t_4+2, t_2+7))$

30: $(0;0;(t_4+3,t_4+5);(t_5,t_4+4);\varnothing;\varnothing;(t_3+9,t_3+11);(t_5+4,t_5+6);R_{30})$, čia

$R_{30} = R_{14} \cup \{t_4+2 < t_5 < t_2+7\ ;\ 4 < t_4 - t_2 < 5\}$

$l_{14,31} = (e_A, t_5 \in (t_4+2, t_2+7))$

31: $(0;0;(t_4+3,t_4+5);\varnothing;(t_5,t_2+7);(t_5+14,t_5+16);(t_3+9,t_3+11);\varnothing;R_{31})$, čia

$R_{31} = R_{14} \cup \{t_4+2 < t_5 < t_2+7\ ;\ 4 < t_4 - t_2 < 5\}$

$l_{15,32} = (e_D, t_6 \in (t_2+9, t_2+11))$

32: $(0;0;(t_5+3,t_5+5);(t_5+2,t_5+4);\varnothing;(t_4+14,t_4+16);\varnothing;\varnothing;R_{32})$, čia

$R_{32} = R_{15} \cup \{t_2+9 < t_6 < t_2+11\ ;\ 7 < t_5 - t_2 < 8\}$

$l_{32,33} = (e_A, t_7 \in (t_5+2, t_5+3))$

33: $(0;0;(t_5+3,t_5+5);\varnothing;\varnothing;(t_4+14,t_4+16);(t_7+9,t_7+11);\varnothing;R_{33})$, čia

$R_{33} = R_{32} \cup \{t_5+2 < t_7 < t_5+3\}$

$l_{32,34} = (e_A, t_7 \in (t_5+3, t_5+4))$

34: $(0;0;(t_7,t_5+5);\varnothing;\varnothing;(t_4+14,t_4+16);(t_7+9,t_7+11);\varnothing;R_{34})$, čia

$R_{33} = R_{32} \cup \{t_5+3 < t_7 < t_5+4\}$

$l_{32,35} = (e, t_7 \in (t_5+3, t_5+4))$

35: $(0;0;(t_7+3,t_7+5);(t_7,t_5+4);(t_7+5,t_7+7);(t_4+14,t_4+16);\varnothing;\varnothing;R_{35})$, čia

$R_{35} = R_{32} \cup \{t_5+3 < t_7 < t_5+4\}$

$l_{33,36} = (e, t_8 \in (t_5 + 3, t_5 + 5))$

36: $(0;0;(t_8 + 3, t_8 + 5);(t_8 + 2, t_8 + 4);\varnothing;(t_4 + 14, t_4 + 16);\varnothing;\varnothing;R_{36})$, čia

$R_{36} = R_{33} \cup \{t_5 + 3 < t_8 < t_5 + 5\}$

$l_{36,37} = (e_A, t_9 \in (t_8 + 2, t_8 + 4))$

37: $(0;0;(t_8 + 3, t_8 + 5);\varnothing;\varnothing;(t_4 + 14, t_4 + 16);(t_9 + 9, t_9 + 11);\varnothing;R_{37})$, čia

$R_{37} = R_{36} \cup \{t_8 + 2 < t_9 < t_8 + 4\}$

$l_{37,38} = (e, t_{10} \in (t_8 + 3, t_8 + 5))$

38: $(0;0;(t_{10} + 3, t_{10} + 5);(t_{10} + 2, t_{10} + 4);\varnothing;(t_4 + 14, t_4 + 16);(t_9 + 9, t_9 + 11);\varnothing;R_{38})$, čia

$R_{38} = R_{37} \cup \{t_8 + 3 < t_{10} < t_8 + 5\}$

$l_{37,38} = (e, t_{10} \in (t_8 + 3, t_8 + 5))$

38: $(0;0;(t_{10} + 3, t_{10} + 5);(t_{10} + 2, t_{10} + 4);\varnothing;(t_4 + 14, t_4 + 16);(t_9 + 9, t_9 + 11);\varnothing;R_{38})$, čia

$R_{38} = R_{37} \cup \{t_8 + 3 < t_{10} < t_8 + 5\}$

$l_{38,39} = (e_A, t_{11} \in (t_{10} + 2, t_{10} + 3))$

39: $(1;0;(t_{10} + 3, t_{10} + 5);\varnothing;\varnothing;(t_4 + 14, t_4 + 16);(t_9 + 9, t_9 + 11);\varnothing;R_{39})$, čia

$R_{39} = R_{38} \cup \{t_{10} + 2 < t_{11} < t_{10} + 5 \; ; \; 6 < t_{10} - t_4 < 10\}$

$l_{38,40} = (e_A, t_{11} \in (t_{10} + 3, t_{10} + 4))$

$l_{38,41} = (e, t_{11} \in (t_{10} + 3, t_{10} + 4))$

$l_{39,42} = (e, t_{12} \in (t_{10} + 3, t_{10} + 5))$

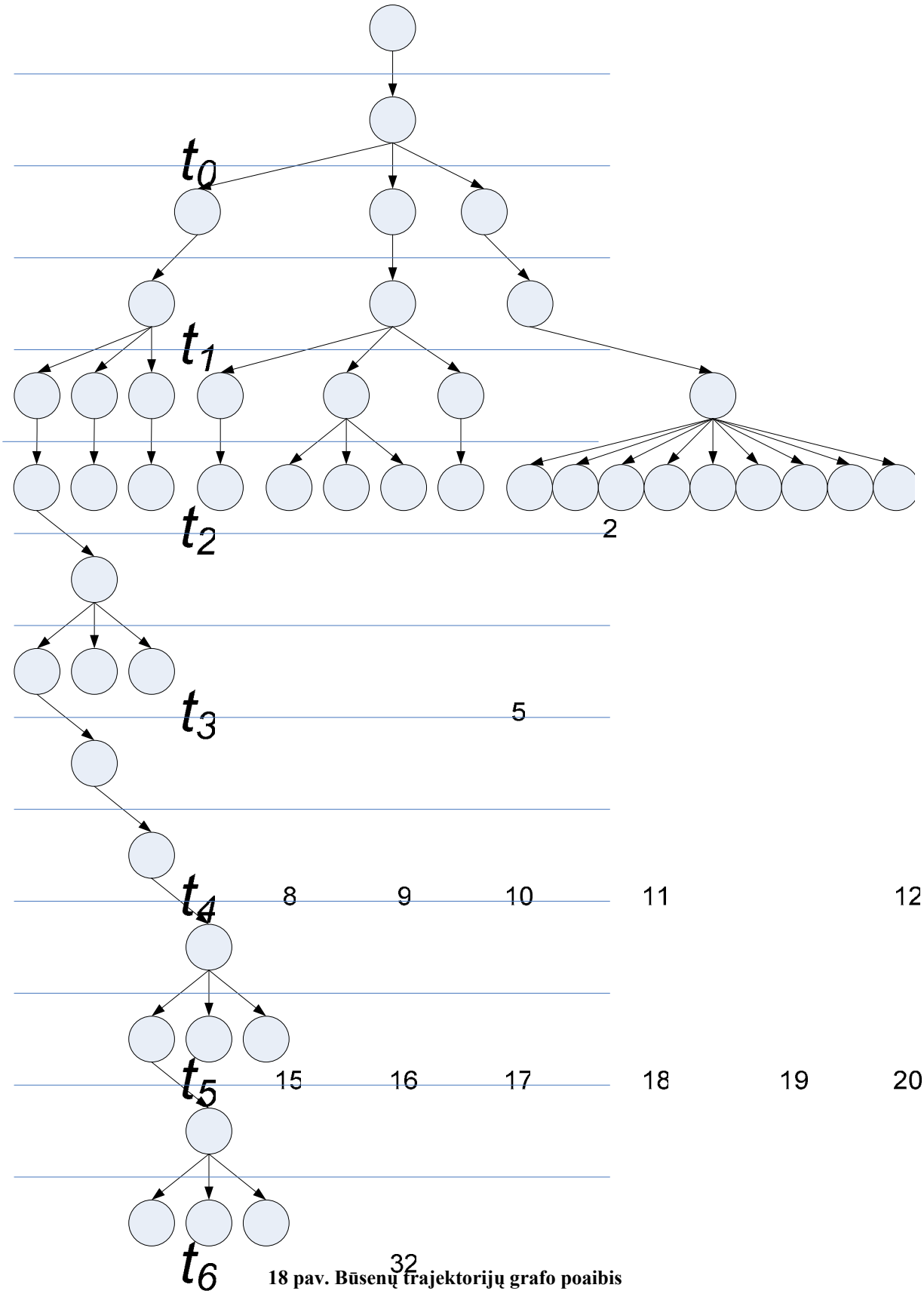
42: $(1;0;(t_{12} + 3, t_{12} + 5);\varnothing;(t_{12} + 5, t_{12} + 7);(t_4 + 14, t_4 + 16);(t_9 + 9, t_9 + 11);\varnothing;R_{42})$, čia

$R_{42} = R_{39} \cup \{t_{10} + 3 < t_{12} < t_{10} + 5 \; ; \; 6 < t_{10} - t_4 < 9\}$

$l_{42,43} = (e, t_{13} \in (t_{12} + 3, t_4 + 14))$

43: $(1;0;(t_{13} + 3, t_{13} + 5);(t_{13} + 2, t_{13} + 4);(t_{12} + 5, t_{12} + 7);(t_4 + 14, t_4 + 16);(t_9 + 9, t_9 + 11);\varnothing;R_{43})$,

čia $R_{43} = R_{42} \cup \{t_{12} + 3 < t_{13} < t_4 + 14 \; ; \; 9 < t_2 - t_4 < 11 \; ; \; 2 < t_{12} - t_9 < 4\}$.



18 pav. Būsenų trajektorijų grafo poaibis

Page Break

3 priedas. Dvikanalės aptarnavimo sistemos specifikacija bei pasiekiamų būsenų grafas.

Pateikiamas dvikanalės aptarnavimo sistemos pasiekiamų būsenų grafas tolydaus laiko atveju, jo viršūnės ir perėjimai.

Remiantis 2.4.2 skyriuje pateikta pasiekiamų būsenų grafo, įvertinančio laiką, sudarymo metodika, sudarytas pasiekiamų būsenų grafas dvikanalei aptarnavimo sistemai, kurios aprašymas ir agregatinė specifikacija pateikta 2.2.1 skyriuje, įvertinant laikinius operacijų trukmių apribojimus ir surandant ekvivalenčias būsenas. Skaičiavimai nėra pateikiami, nes jie vykdomi analogiškai, kaip pateikta 2 priede. Ekvivalenčių būsenų apibrėžimas yra pateikiamas 3.2 skyriuje. Žymėjimas: ties viršūne per tašką yra parašoma viršūnė, kuriai ši yra ekvivalenti.

Pasiekiamų būsenų medžio viršūnės:

$1:(0;(t_0+4,t_0+6),\varnothing,\varnothing;R_0)$, čia $R_0=\varnothing$;

$2:(0;(t_1+4,t_1+6),(t_1+3,t_1+5),\varnothing;R_{11})$, čia $R_{11}=R_0\cup\{t_0+4<t_1<t_0+6\}$;

$3:(0;(t_1+4,t_1+6),\varnothing,\varnothing;R_{21})$, čia $R_{21}=R_{11}\cup\{t_1+3<t_2<t_1+4\}$;

$4:(0;(t_2,t_1+6),\varnothing,\varnothing;R_{22})$, čia $R_{22}=R_{11}\cup\{t_1+4<t_2<t_1+5\}$;

$5:(0;(t_2+4,t_2+6),(t_2,t_1+5),(t_2+2,t_2+4);R_{23})$, čia $R_{23}=R_{11}\cup\{t_1+4<t_2<t_1+5\}$;

$6:(0;(t_3+4,t_3+6),(t_3+3,t_3+5),\varnothing;R_{31})$, čia $R_{31}=R_{21}\cup\{t_1+4<t_3<t_1+6\}$;

$9:(0;(t_3+4,t_3+6),\varnothing,\varnothing;R_{61})$, čia $R_{61}=R_{31}\cup\{t_3+3<t_4<t_3+4\}$;

$10:(0;(t_4,t_3+6),\varnothing,\varnothing;R_{62})$, čia $R_{62}=R_{31}\cup\{t_3+4<t_4<t_3+5\}$;

$11:(0;(t_4+4,t_4+6),(t_4,t_3+5),(t_4+2,t_4+4);R_{63})$, čia $R_{63}=R_{31}\cup\{t_3+4<t_4<t_3+5\}$;

$17:(0;(t_5+4,t_5+6),(t_5+3,t_5+5),\varnothing;R_{101})$, čia $R_{101}=R_{62}\cup\{t_4<t_5<t_3+6\}$;

$18:(0;(t_4+4,t_4+6),\varnothing,(t_4+2,t_4+4);R_{111})$, čia $R_{111}=R_{63}\cup\{t_4<t_5<t_3+5\}$;

$26:(0;(t_5+4,t_5+6),\varnothing,\varnothing;R_{171})$, čia $R_{171}=R_{101}\cup\{t_5+3<t_6<t_5+4\}$;

$27:(0;(t_6,t_5+6),\varnothing,\varnothing;R_{172})$, čia $R_{172}=R_{101}\cup\{t_5+4<t_6<t_5+5\}$;

$28:(0;(t_6+4,t_6+6),(t_6,t_5+5),(t_6+2,t_6+4);R_{173})$, čia $R_{173}=R_{101}\cup\{t_5+4<t_6<t_5+5\}$;

$29:(0;(t_4+4,t_4+6),\varnothing,\varnothing;R_{181})$, čia $R_{181}=R_{111}\cup\{t_4+2<t_6<t_4+4\}$;

$37:(0;(t_6+4,t_6+6),\varnothing,(t_6+2,t_6+4);R_{281})$, čia $R_{281}=R_{173}\cup\{t_6<t_7<t_5+5\}$;

$38:(0;(t_7+4,t_7+6),(t_7+3,t_7+5),\varnothing;R_{291})$, čia $R_{291}=R_{181}\cup\{t_4+4<t_7<t_4+6\}$;

$43:(0;(t_6+4,t_6+6),\varnothing,\varnothing;R_{371})$, čia $R_{371}=R_{281}\cup\{t_6+2<t_8<t_6+4\}$;

$44:(0;(t_9+4,t_9+6),(t_9+3,t_9+5),\varnothing;R_{431})$, čia $R_{431}=R_{371}\cup\{t_6+4<t_9<t_6+6\}$;

$59:(0;(t_9+4,t_9+6),\varnothing,\varnothing;R_{441})$, čia $R_{441}=R_{431}\cup\{t_9+3<t_{10}<t_9+4\}$;

$60:(0;(t_{10},t_9+6),\varnothing,\varnothing;R_{442})$, čia $R_{442}=R_{431}\cup\{t_9+4<t_{10}<t_9+5\}$;

$61:(0;(t_{10}+4,t_{10}+6),(t_{10},t_9+5),(t_{10}+2,t_{10}+4);R_{443})$, čia $R_{443}=R_{431}\cup\{t_9+4<t_{10}<t_9+5\}$;

$62:(0;(t_7+4,t_7+6),\varnothing,\varnothing;R_{381})$, čia $R_{381}=R_{291}\cup\{t_7+3<t_8<t_7+4\}$;

$63:(0;(t_8,t_7+6),\varnothing,\varnothing;R_{382})$, čia $R_{382}=R_{291}\cup\{t_7+4<t_8<t_7+5\}$;

$64:(0;(t_8+4,t_8+6),(t_8,t_7+5),(t_8+2,t_8+4);R_{383})$, čia $R_{383}=R_{291}\cup\{t_7+4<t_8<t_7+5\}$;

$65:(0;(t_9+4,t_9+6),(t_9+3,t_9+5),\varnothing;R_{631})$, čia $R_{631}=R_{382}\cup\{t_8<t_9<t_7+6\}$;

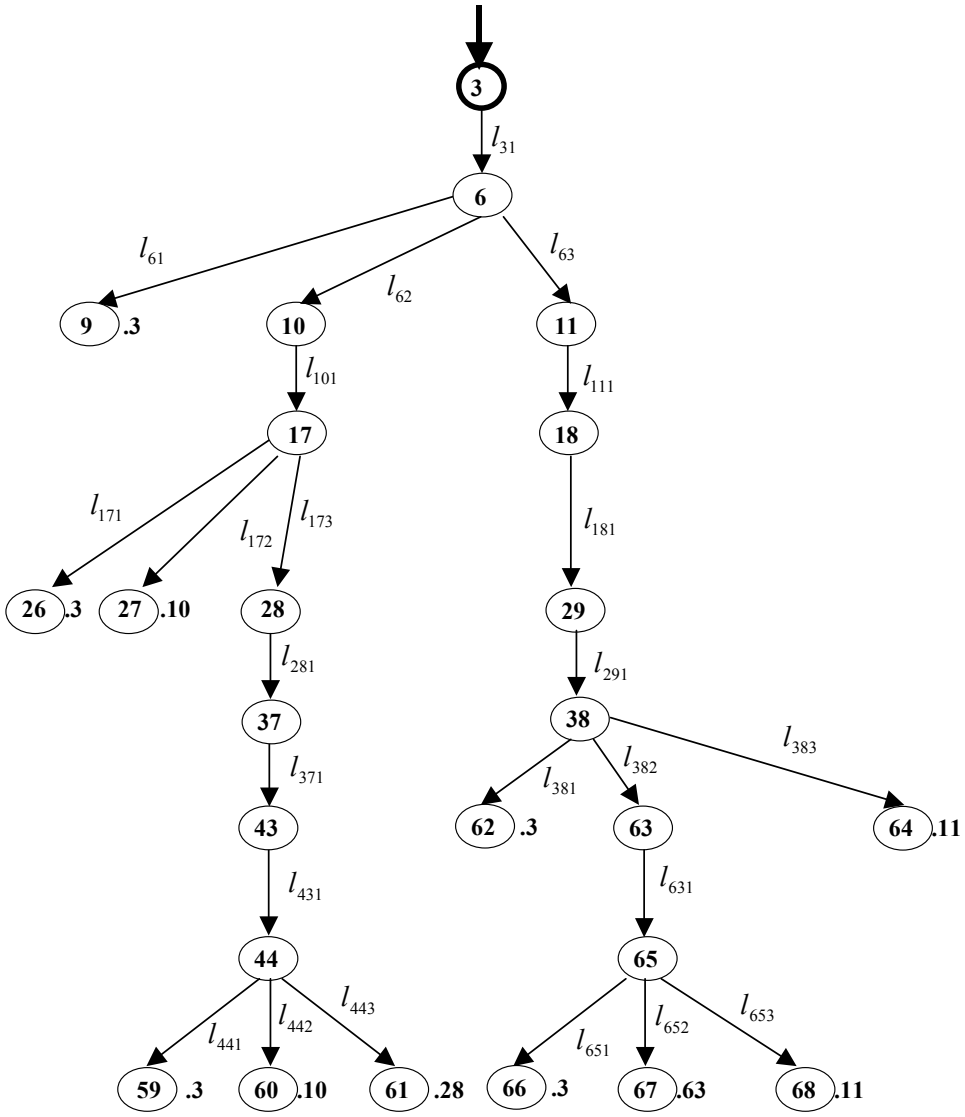
$66:(0;(t_9+4,t_9+6),\varnothing,\varnothing;R_{651})$, čia $R_{651}=R_{631}\cup\{t_9+3<t_{10}<t_9+4\}$;

$67:(0;(t_{10},t_9+6),\varnothing,\varnothing;R_{652})$, čia $R_{652}=R_{631}\cup\{t_9+4<t_{10}<t_9+5\}$;

$68:(0;(t_{10}+4,t_{10}+6),(t_{10},t_9+5),(t_{10}+2,t_{10}+4);R_{653})$, čia $R_{653}=R_{631}\cup\{t_9+4<t_{10}<t_9+5\}$;

Pasiekiamų būsenų grafo perėjimai:

$l_{11}=(e_1,t_1\in(t_0+4,t_0+6))$;
$l_{21}=(e_2,t_2\in(t_1+3,t_1+4))$;
$l_{22}=(e_2,t_2\in(t_1+4,t_1+5))$;
$l_{23}=(e_1,t_2\in(t_1+4,t_1+5))$;
$l_{31}=(e_1,t_3\in(t_1+4,t_1+6))$;
$l_{61}=(e_2,t_4\in(t_3+3,t_3+4))$;
$l_{62}=(e_2,t_4\in(t_3+4,t_3+5))$;
$l_{63}=(e_1,t_4\in(t_3+4,t_3+5))$;
$l_{101}=(e_1,t_5\in(t_4,t_3+6))$;
$l_{111}=(e_2,t_5\in(t_4,t_3+5))$;
$l_{171}=(e_2,t_6\in(t_5+3,t_5+4))$;
$l_{172}=(e_2,t_6\in(t_5+4,t_5+5))$;
$l_{173}=(e_1,t_6\in(t_5+4,t_5+5))$;
$l_{181}=(e_3,t_6\in(t_4+2,t_4+4))$;
$l_{281}=(e_2,t_7\in(t_6,t_5+5))$;
$l_{291}=(e_1,t_7\in(t_4+4,t_4+6))$;
$l_{371}=(e_3,t_8\in(t_6+2,t_6+4))$;
$l_{381}=(e_2,t_8\in(t_7+3,t_7+4))$;
$l_{382}=(e_2,t_8\in(t_7+4,t_7+5))$;
$l_{383}=(e_1,t_8\in(t_7+4,t_7+5))$;
$l_{431}=(e_1,t_9\in(t_6+4,t_6+6))$;
$l_{441}=(e_2,t_{10}\in(t_9+3,t_9+4))$;
$l_{442}=(e_2,t_{10}\in(t_9+4,t_9+5))$;
$l_{443}=(e_1,t_{10}\in(t_9+4,t_9+5))$;
$l_{631}=(e_1,t_9\in(t_8,t_7+6))$;
$l_{651}=(e_2,t_{10}\in(t_9+3,t_9+4))$;
$l_{652}=(e_2,t_{10}\in(t_9+4,t_9+5))$;
$l_{653}=(e_1,t_{10}\in(t_9+4,t_9+5))$;



19 pav. Dvikanalės aptarnavimo sistemos pasiekiamų būsenų grafas nuo 3 viršūnės