KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Lina Cibulskienė

dynPLA panaudojimas mobilaus ryšio sistemų modeliavimui

Magistro darbas

Darbo vadovas
prof. habil. dr. H. Pranevičius

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Lina Cibulskienė

dynPLA panaudojimas mobilaus ryšio sistemų modeliavimui

Magistro darbas

Recenzentas

dr. Robertas Damaševičius
2007-05-24

Vadovas

prof. habil. dr. H. Pranevičius
2007-05-

Atliko

IFM-1/1 gr. stud.
Lina Cibulskienė
2007-05-24

Kaunas, 2007

Summary

The mobile computing system will be analyzed in this work. It is assumed that network consists of two types of devices: mobile and fixed. Fixed devices are always on and always connected to the network. These devices include, but are not limited to database servers and mobile support stations (MSS). Mobile devices access data from database. They connect and reconnect to different MSS depending on the signal strength. All of this is done seamlessly to the end user and the data transfer is not disturbed.

When planning to implement such system in programming language, the first step is analyzing and formalizing it. One of the formalization languages – PLA, was recently extended for the sole purpose of formalization of dynamic systems.

This work, "The Usage of dynPLA Formalism for Modeling of Mobile Computing System" covers the usage of new extended formalization language (dynPLA) for mobile computing system, analyzing if dynPLA extensions are sufficient for formalizing the said system. Also some improvements for the dynPLA formalization language will be suggested and the example of the mobile computing system will be given.

TURINYS

1. Įvadas

Kaip žinia, vis daugiau ir daugiau programinės įrangos kūrimu užsiimančių firmų prieš pradėdamos realizuoti produktą stengiasi kuo išsamiau aprašyti sistemos veikimo principus, galimybes, reikalavimus. Šiuo metu pagrindinę kuriamų sistemų dalį sudaro sistemos skirtos e-komercijai, e-darbui, e-informacijai ir t.t. Tokios sistemos privalo sugebėti prisitaikyti nuolat besikeičiančioje aplinkoje, mokėti keisti savo elgseną ir struktūrą. Šito nesugebančios sistemos tampa bevertėmis, o sparčiai keičiantis ir tobulėjant programinei bei aparatūrinei įrangai atsiranda vis didesnių ir sudėtingesnių sistemų, kurias išanalizuoti ir aprašyti tampa vis sunkiau. Kaip tik tam ir buvo sukurta PLA (*Piece Linear Aggregates*) formalizavimo kalba. Tačiau tokia formalizavimo kalba sugeba suspecifikuoti tik paprastas, laike nekintančias sistemas. Norint specifikuoti dinamines sistemas, PLA formalizavimo kalba buvo praplėsta įvedant papildomas aibes bei matricas [2], pritaikant ją besikeičiančių sistemos būsenų aprašymui.

Toks PLA formalizmo kalbos praplėtimas ir skirtas nagrinėti bei specifikuoti kintančias laike sistemas.

Sukurta dynPLA formalizavimo kalba turėtų leisti aprašyti dinamines sistemas, kuriose pagrindinis agregatas vykdo valdymo funkcijas, t.y. žino visus kitų agregatų įėjimus, išėjimus bei kontroliuoja jų veiksmus.

Iš pirmo žvilgsnio atrodytų, jog problema gali atsirasti kai sistemoje esantys agregatai yra visiškai individualūs. Tokiems agregatams sąveikaujant tarpusavyje egzistuoja atvejis kai nei vienas iš jų nėra valdantysis, t.y. tam tikra operacija sąlygojama išorinio įvykio arba inicijuojama tam tikro agregato, kuris nesugeba ar neturi teisės valdyti kito.

Nagrinėjant būtent tokias sistemas, galima susidurti su nemažais sunkumais bandant jas aprašyti. Po ilgo ir išsamaus sistemos, bei formalizavimo kalbos nagrinėjimo prieita prie išvados, jog yra būtina įvesti visus sistemos agregatus apgaubiantį agregatą, kuris reikalui esant gali pasidalinti turima informacija apie sistemą. Taip yra todėl, jog net labiausiai savarankiška ir niekieno nevaldoma sistema turi valdantį agregatą, tiesiog kartais jie yra nematomi paprastiems vartotojams, suteikiant klaidingą vaizdą, jog valdantysis agregatas neegzistuoja.

Dinaminės sistemos nagrinėjimas bei specifikavimas leidžia atskleisti visas dynPLA formalizavimo kalbos galimybes bei trūkumus, suteikiant progą ištaisyti egzistuojančius trūkumus, jei tokių yra.

1.1 Uždavinio formulavimas

Šiame darbe siekiama išsiaiškinti PLA (*Piece Linear Aggregates*) formalizavimo kalbos ypatybes bei galimybes. Kadangi ši kalba yra viena iš pagrindinių formalizavimo kalbų,

o taip pat laiko patikrinta ir patobulinta, savo magistriniame darbe nagrinėsiu kiek naujesnę šios kalbos atmainą – dynPLA. Ši formalizmo kalba skirta aprašyti dinaminėms, laike kintančioms sistemoms.

Darbe siekiama įsitikinti, jog dynPLA formalizavimo kalba yra tinkama dinaminėms sistemoms aprašyti, kadangi tai yra labai jauna ir neišbandyta kalba.

1.2 Tikslai

1. Išnagrinėti dynPLA formalizavimo kalbos ypatybes ir jos galimybes dinaminių sistemų aprašymui.
2. Susipažinti su šios formalizavimo kalbos savybėmis.
3. Sukurti mobilaus ryšio sistemos formalųjį modelį naudojant dynPLA.

1.3 Darbo originalumas ir naujumas

Dinaminių sistemų kūrimas, nagrinėjimas bei aprašymas egzistuoja dar visai neilgai. Taip pat tokiom sistemom aprašyti naudojama formalizavimo kalba yra gana jauna. Dar visai nesenai buvo pristatytas PLA formalizavimo kalbos praplėtimas skirtas laike kintančioms sistemoms aprašyti. Ir išskyrus keletą paprastų pavyzdžių, dynPLA nebuvo išbandyta stengiantis aprašyti pilną dinaminę sistemą, todėl yra būtinybė įsitikinti jos galimybėmis bei pataisyti trūkumus, jei tokie atsirastų, išsamiai nagrinėjant dynPLA.

2. Analitinė dalis

Formalizavimo kalba – tai yra įrankis sistemų specifikavimui. Savo darbe naudoju PLA, tiksliau dynPLA formalizavimo kalbą. Tačiau PLA nėra vienintelė formalizmo kalba, todėl supažindinsiu ir su alternatyva – DEVS formalizavimo kalba.

2.1 PLA formalizavimo kalba

PLA (*Piece Linear Aggregates*) [1] yra viena iš formalizavimo kalbų, leidžianti aprašyti sistemas. Bandant sistemas aprašyt agregatiniu būdu, sistema pateikiama kaip tarpusavyje sąveikaujančių atkarpomis – tiesinių agregatų rinkinys. PLA naudojamas kaip objektas, kurį apibūdina būsenų Z, įėjimo signalų X ir išėjimo signalų Y rinkinys. Agregato funkcionavimas stebimas laiko momentų $t \in T$ aibėje, t.y. agregato būsena $z \in Z$, įėjimo signalai $x \in X$ ir išėjimo signalai $y \in Y$ yra laiko funkcijos. Be minėto rinkinio dar naudojami perėjimo operatoriai H ir išėjimos operatoriai G.

Atkarpomis tiesinio agregato būsenos struktūra yra tokia pat kaip ir atkarpomis tiesinio Markovo proceso, t.y.

$$z(t) = (v(t), z_v(t)),$$

čia $v(t)$ - diskrečioji būsenų dedamoji,

$z_v(t)$ - tolydi būsenų dedamoji.

Kai nėra įėjimo signalų, agregato būsenos kinta taip:

$$v(t) = const, \quad \frac{dz_v(t)}{dt} = -a_v;$$

$a_v = (a_{v1}, a_{v2}, ..., a_{vk})$ – pastovus vektorius.

Agregato būsena gali kisti tik dviem atvejais: kai į agregatą siunčiamas įėjimo signalas arba kai viena iš tolydžiosios dedamosios koordinačių įgyja tam tikrą reikšmę.

Agregatų funkcionavimas stebimas laiko momentais $T = \{t_0, t_1, ..., t_m, ...\}$, kuriais įvyksta vienas ar keli įvykiai, sąlygojantys agregato būsenos pokyčius. Įvykių, kurie gali įvykti rinkinys yra dalinamas į du nepersikertančius poaibius $E = E^{'} \cup E^{''}$, $E^{'} \cap E^{''} = \varnothing$. $E^{'} = \{e_1^{'}, e_2^{'}, ..., e_N^{'}\}$ poaibis susideda iš įvykių $e_i^{'}$, $i = \overline{1, N}$, kurie įvyksta dėl įėjimo signalų $X = \{x_1, x_2, ...x_N\}$ atėjimo. Tarp aibių X ir $E^{'}$ elementų yra funkcinis ryšys. Įvykiai iš poaibio $E^{'}$ vadinami išoriniais. Poaibis $E^{''} = \{e_1^{''}, e_2^{''}, ..., e_f^{''}\}$ yra vadinamas vidinių įvykių poaibiu; čia $e_i^{''} = \{e_{ij}^{''}, j = 1,2,3,...\}$, $i = \overline{1, f}$, yra agregato vidiniai įvykiai, f - operacijų, kurios gali vykti agregate, skaičius. Aibės $E^{''}$ įvykiai fiksuoja operacijų pabaigą.

Kiekvienam vidiniam įvykiui $e_i^{"}$ iš poaibio $E^{"}$ priskiriama valdymo seka $\{\xi_j^{(i)}\}$,

čia $\xi_j^{(i)}$ - operacijos trukmė, kuriai pasibaigus įvyksta vidinis įvykis.

Taip pat nurodoma skaitiklių aibė:

$$\{r(e_i^{"}, t_m)\}, i = \overline{1, f};$$

čia $r(e_i^{"}, t_m)$ - $e_i^{"}$ įvykių skaičius laiko tarpe $[t_0, t_m]$.

Tam, kad būtų įmanoma nustatyti operacijų pradžios ir pabaigos momentus, naudojamos valdymo sumų aibės:

$$\{s(e_i^{"}, t_m)\},\ \{w(e_i^{"}, t_m)\}, i = \overline{1, f};$$

čia $s(e_i^{"}, t_m)$ - operacijos pradžios momentas, po kurio įvyksta $e_i^{"}$ įvykis;

Šis laiko momentas yra neapibrėžtas, jei operacija dar neprasidėjo.

$w(e_i^{"}, t_m)$ - operacijos pabaigos momentas. Neprioritetinių operacijų atveju valdymo suma $w(e_i^{"}, t_m)$ apibrėžiama taip:

$$w(e_i^{"}, t_m) = \begin{cases} s'(e_i^{"}, t_m) + \xi_{r(e_i^{"}, t_m)+1}, & \text{jei laiko momentu } t_m \text{ vyksta operacija, kuriai pasibaigus} \\ & \text{įvyks įvykis } e_i^{"} \\ \infty, & \text{kitu atveju.} \end{cases}$$

Begalybės simbolis (∞) vartojamas, kai reikia pažymėti, jog operacijos pabaigos momentas yra nežinomas.

Pateiktas valdymo sumos apibrėžimas vartojamas sudarant imitacinius modelius. Kai agregatinis modelis naudojamas sistemoms formalizuoti ir sistemos funkcionavimo teisingumui analizuoti, valdymo suma apibrėžiama supaprastintai:

$$w(e_i^{"}, t_m) = \begin{cases} < \infty, & \text{jei laiko momentu } t_m \text{ vyksta operacija, kuriai pasibaigus įvyks įvykis } e_i^{"}, \\ \infty, & \text{priešingu atveju.} \end{cases}$$

Valdymo sumos apsprendžia tik galimas įvykių padėtis po laiko momento t_m, kai įvykių atsitikimo momentai yra nenustatyti.

Apibrėžkime agregato būsenų dedamąsias koordinates. Diskrečioji būsenos dedamoji $v(t_m) = \{v_1(t_m), v_2(t_m), ..., v_p(t_m)\}$, apibrėžia sistemos būsenas, t.y. perduotų ir gautų informacijos paketų skaitikliai, pasirengimas informacijos perdavimui ir t.t.

Įvedus valdymo sumas, agregato būsenos tolydžioji dedamoji įgauna pavidalą:

$$z_v(t_m) = \{w(e_1^{"}, t_m), w(e_2^{"}, t_m), ..., w(e_f^{"}, t_m)\}.$$

Tolydžiosios dedamosios koordinatės apibrėžia laiko momentus, kada agregate galimi įvykiai. $w(e_1^{"}, t_m)$ reaguoja į kiekvieną įvykį $e_i^{"}$ iš įvykių aibės $E^{"}$, be to, visada $w(e_1^{"}, t_m) \geq t_m$.

Agregato būsena $z(t_m)$ kinta diskrečiais laiko momentais t_m, $m = 1,2,...,$ ir išlieka pastovi laiko tarpuose $[t_m, t_{m+1}), m = 0,1,2,...;$ čia t_0 – pradinis sistemos funkcionavimo momentas.

Žinant agregato būseną $z(t_m)$, $m = 0,1,2,...,$ laiko momentas t_{m+1}, kai įvyksta įvykis, apskaičiuojamas taip:

$$t_{m+1} = \min_i \{w(e_i^{''}, t_m)\}, 1 \le i \le f.$$

Operatorius H apibrėžia naują agregato būseną:

$$z(t_{m+1}) = H[z(t_m, e_i)], e_i \in E^{'} \cup E^{''}.$$

Operatorius G apibrėžia išėjimo signalus:

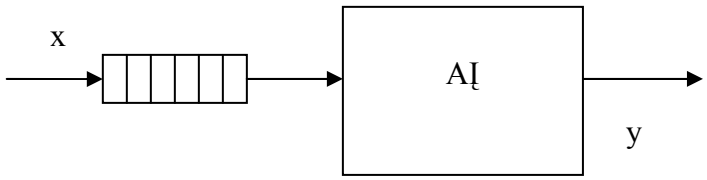
$$y = G[z(t_m, e_i)], e_i \in E^{'} \cup E^{''}, y \in Y.$$

Toliau perėjimo ir išėjimos operatoriai bus žymimi atitinkamai $H(e_i)$ ir $G(e_i)$.

2.1.1 paprastos agregatinės aptarnavimo sistemos specifikacija

Konceptualusis modelis

Į sistemą ateina n paraiškų. Kiekviena iš jų eilėje laukia aptarnavimo. Aptarnaujantis įrenginys AĮ, atėjusias paraiškas aptarnauja po vieną, iš eilės.



1 pav. *Aptarnavimo sistemos struktūra.*

Agregatinis modelis

1. Įėjimų signalų aibė $X = \{x_1\}$;

2. Išėjimo signalų aibė $Y = \{y\}$;

3. Išorinių įvykių aibė $E^{'} = \{e_1^{'}\}$;

   $e_1^{'}$ – atėjo signalas x_1.

4. Vidinių įvykių aibė $E^{''} = \{e_1^{''}\}$;

   $e_1^{''}$ – baigės paraiškos aptarnavimas;

5. Valdymo sekos $\{e_1^{''}\} \to \{\eta_j^{(1)}\}, j = \overline{1,\infty}$;

$\{\eta_j^{(1)}\}$ – j – osios paraiškos aptarnavimo trukmė.

6. Diskrečioji būsenos dedamoji

$v(t_m) = n(t_m)$;

$n(t_m)$ – paraiškų skaičius eilėje laiko momentu t_m.

7. Tolydžioji būsenos dedamoji

$z_v(t_m) = \{w(e_1^{''}, t_m)\}$;

$w(e_1^{''}, t_m)$ – laiko momentas, kada baigsis j – osios paraiškos aptarnavimas.

8. Pradinė būsena

Pradiniu laiko momentu $t_0 = 0$.

$v(t_0) = 0, \;\; z_v(t_0) = \infty.$

9. Perėjimo operatoriai:

$H(e_1^{'})$:

$$n(t_{m+1}) = n(t_m) + 1;$$

$$w(e_1^{''}, t_{m+1}) = \begin{cases} t_m + \eta_j^{(1)}, & jei \;\; n(t_m) = 0 \\ w(e_1^{''}, t_m), & jei \;\; n(t_m) > 0 \end{cases}$$

$H(e_1^{'})$:

$$n(t_{m+1}) = n(t_m) - 1;$$

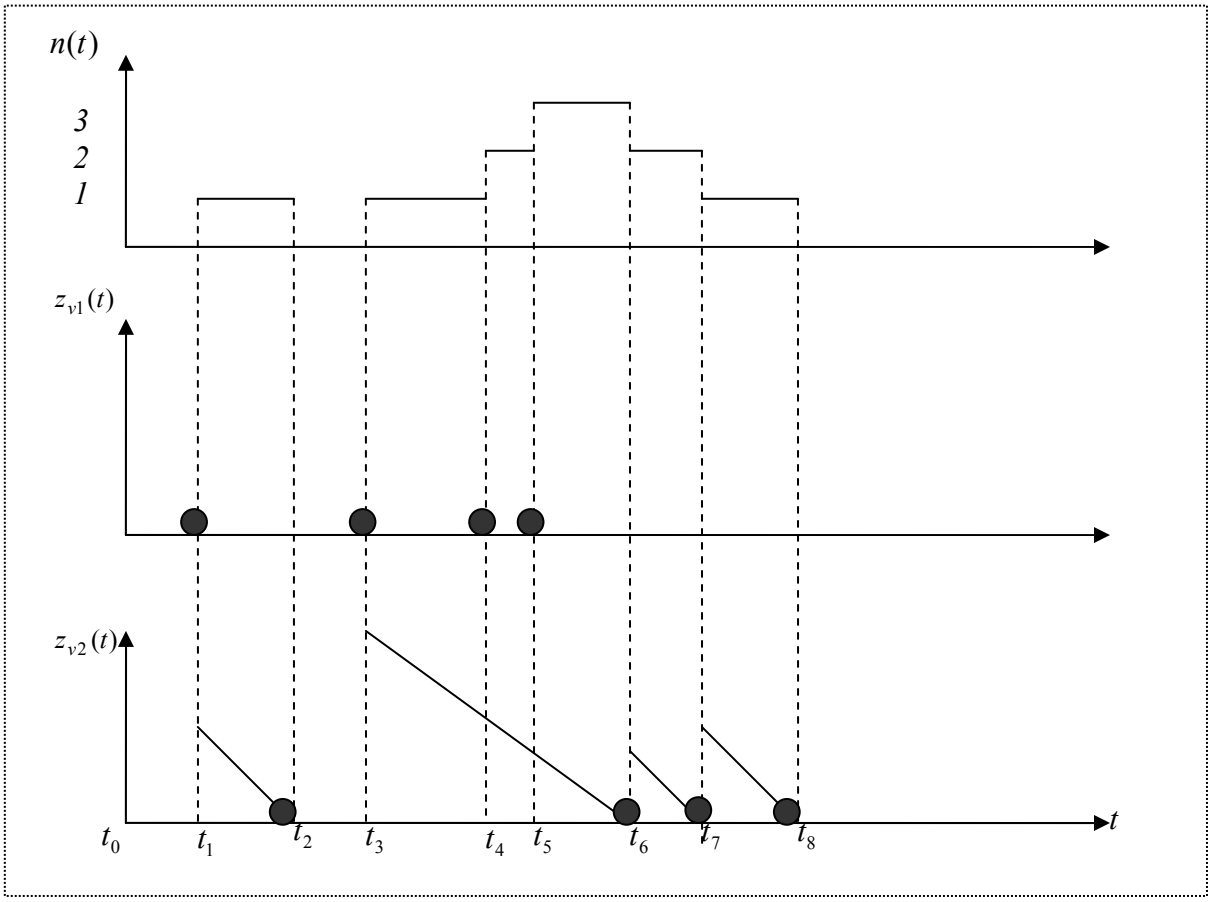
$$w(e_1^{''}, t_{m+1}) = \begin{cases} \infty, & jei \;\; n(t_{m+1}) = 0 \\ t_m + \eta_j^{(1)}, & jei \;\; n(t_{m+1}) > 0 \end{cases}$$

$G(e_1^{''})$:

$y = t_m$.

Kaip pavaizduota paveikslėlyje *1*, į sistemą atėjusi paraiška statoma į eilę, kurioje ji laukia kol bus aptarnauta.

Paveikslėlyje *2* pavaizduotas aptarnavimo sistemos būsenų kitimas bėgant laikui.



2 pav. *Aptarnavimo sistemos koordinačių kitimas laike.*

$n(t)$ - paraiškų skaičius sistemoje laiko momentu t.

$z_{v1}(t)$ - laiko momentas, kada į sistemą ateina nauja paraiška.

$z_{v2}(t)$ - laiko momentas, kada baigiamas apraiškos aptarnavimas.

Kaip matyti iš paveikslėlio *2*, į sistemą naujos paraiškos ateina laiko momentais t_1, t_3, t_4 ir t_5. Laiko momentais t_2, t_6, t_7 ir t_8 baigiasi paraiškų aptarnavimas. Žinant kada į sistemą ateina paraiška ir kada baigiasi jos aptarnavimas, lengvai galime nustatyti aptarnavimo trukmę.

Paraiškos aptarnavimo trukmė pažymėta pasvirusiu brūkšniuku nuo apraiškos atėjimo į sistemą iki jos aptarnavimo pabaigos momento.

2.2 Dinaminė PLA formalizavimo kalba (dynPLA)

Dinaminis PLA, tai yra standartinės PLA formalizavimo kalbos išplėtimas, kuris leidžia apibrėžti kintantį laike sistemos modelį. Norint iš paprastos PLA formalizavimo kalbos pereiti prie dinaminės reikia įvesti papildomas aibes [2]. Tokios aibės būtų naudojamos informacijos tam tikru momentu saugojimui. Taigi kintant laikui ir kintant sistemos struktūrai keičiasi ir atitinkamos aibės.

Viena iš tokių aibių yra aibė A su indeksu t. Šioje aibėje saugomi visi momentu t sistemos struktūrą sudarantys agregatai.

$A_{t+1} = A_t \cup \{A_{naujas}\}$ - naujo agregato į sistemos modelį įtraukimas

$A_{t+1} = A_t - \{A_{senas}\}$ - iš sistemos pašalinamas senas agregatas.

Bėgant laikui dinaminėje sistemoje gali įvykti naujos operacijos, susiformuoti nauji agregatai ar išnykti seni. Į sistemą įtraukus naują agregatą jam turi būti nustatomos būsenos, įėjimai, išėjimai. Lygiai taip pat prieš pašalinant senąjį agregatą reikia pašalinti jo įėjimus, išėjimus. Todėl aprašant dinamines sistemas taip pat modifikuojamos įėjimų, išėjimų, operatorių, vidinių bei išorinių įvykių aibės.

Agregato specifikacija praplečiama tokiais veiksmais:

1. Naujo įėjimo agregate sukūrimas

   $X_{t+1} = X_t \cup \{X_{naujas}\}$ - įtraukiamas X_{naujas} įėjimas į agregatą.

2. Įėjimo agregate pašalinimas

   $X_{t+1} = X_t - \{X_{senas}\}$ - pašalinamas X_{senas} įėjimas iš agregato.

3. Naujo išėjimo agregate sukūrimas

   $Y_{t+1} = Y_t \cup \{Y_{naujas}\}$ - įtraukiamas Y_{naujas} išėjimas į agregatą.

4. Išėjimos pašalinimas iš agregato

   $Y_{t+1} = Y_t - \{Y_{senas}\}$ - pašalinamas Y_{senas} išėjimas iš agregato.

5. Vidinių įvykių agregate sukūrimas

   $E''_{t+1} = E''_t \cup \{e''_{naujas}\}$ - įtraukiamas e''_{naujas} vidinis įvykis.

6. Vidinių įvykių pašalinimas iš agregato

   $E''_{t+1} = E''_t - \{e''_{senas}\}$ - pašalinamas e''_{senas} vidinis įvykis.

7. Išorinių įvykių agregate sukūrimas

   $E'_{t+1} = E'_t \cup \{e'_{naujas}\}$ - įtraukiamas e'_{naujas} išorinis įvykis.

8. Išorinių įvykių pašalinimas iš agregato

   $E'_{t+1} = E'_t - \{e'_{senas}\}$ - pašalinamas e'_{senas} išorinis įvykis.

Išorinių įvykių atsiradimas ar išnykimas taip pat įtakoja agregato įėjimų atsiradimus ar pašalinimus. Taigi pašalinus tam tikrą išorinį įvykį reikia pašalint ir atitinkamą agregato įėjimą.

H ir G operatoriai priklauso nuo išorinių ir vidinių įvykių. Egzistuojantys H, G operatoriai turi būti iš anksto aprašyti agregato specifikacijoje. Naujai susiformavę operatoriai, dėl išorinių ar vidinių įvykių atsiradimo, įtraukiami specifikacijos eigoje. Taigi galimybė

modifikuoti išorinių ar vidinių įvykių aibes įtakoja ir H, G operatorių aibes. T.y., įtraukus išorinį ar vidinį įvykį reikia įtraukti ir su tais įvykiais susijusius operatorius.

$H_{t+1} = H_t \cup \{H(e'_{naujas})\}$ - įtraukiamas H operatorius, kuris apdoroja e'_{naujas} įvykį.

$G_{t+1} = G_t \cup \{G(e'_{naujas})\}$ - įtraukiamas G operatorius, kuris apdoroja e'_{naujas} įvykį.

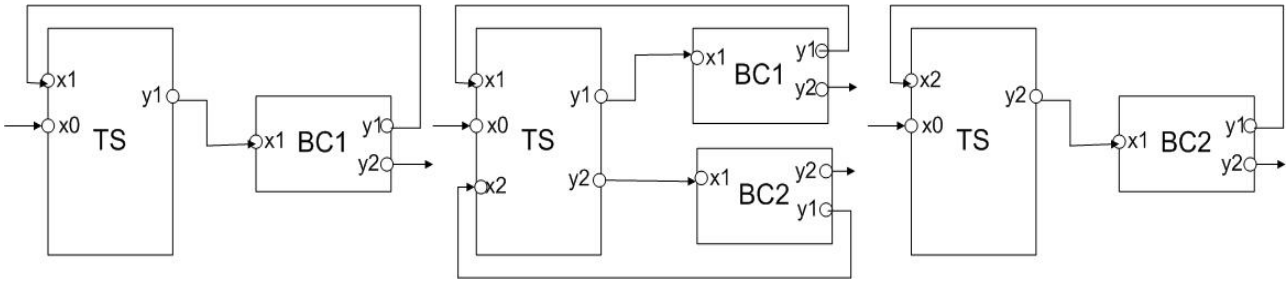
Tokia pati taisyklė galioja ir su išorinių ar vidinių įvykių pašalinimu. T.y., pašalinus išorinį ar vidinį įvykius, reikia pašalinti ir su tais įvykiais susijusius H, G operatorius.

$H_{t+1} = H_t - \{H(e'_{senas})\}$ - įtraukiamas H operatorius, kuris apdoroja e'_{senas} įvykį.

$G_{t+1} = G_t - \{G(e'_{senas})\}$ - įtraukiamas G operatorius, kuris apdoroja e'_{senas} įvykį.

Dinaminėje PLA formalizavimo kalboje taip pat naudojama sujungimų matrica M su indeksu t. Ši matrica atspindi sujungimus tarp sistemos agregatų laiko momentu t.

Iliustruosiu nedidelį pavyzdį, kuris išsamiau leis suprasti PLA formalizmo išplėtimą, naudojant kalbą dinaminėms sistemoms aprašyti. Sistema susideda iš Windows operacinės sistemos užduočių tvarkytojo (TS – task schedule), kuris valdo duomenų atsarginės kopijos kūrimo procesą (BC). Užduočių tvarkytojas gavęs darbą sukuria BC, kuris savo ruožtu atlieka duomenų dubliavimą ir baigęs darbą praneša TS. TS gavęs pranešimą apie darbo pabaigą sunaikina BC. Sistemos agregatinė struktūra pateikta 1 pav.



3 pav.*Dinaminės sistemos agregatinė struktūra skirtingais laiko momentais.*

Paveikslėlyje *3* pradiniu laiko momentu (t_0) sistema susideda iš TS ir vieno BC, kuris vykdo jam nustatytas užduotis. Laiko momentu t_m į sistemą ateina naujas signalas (reikalavimas sukurti naują tam tikrų duomenų kopiją). Tada TS sukuria naują BC (BC2), kuris atlieka reikiamus veiksmus. Tarkim, jog po dar kažkiek laiko BC1 baigia savo darbą, apie darbo užbaigimą jis praneša TS ir ši gavus darbo pabaigos patvirtinimą sunaikiną BC1. Tokiu būdu laikui bėgant kinta sistemos modelis, jis gali tiek plėstis, tiek mažėti.

Žemiau pateikta šios sistemos PLA specifikacija, kuri nuo standartinės skiriasi tuo, jog iš pradžių (laiko momentu t_0) yra apibrėžiamos visos pradinės aibės, tiek įėjimų ar išėjimų, tiek ir operatorių bei pačių agregatų. Sekančiu laiko momentu, į sistemą atėjus reikalavimui sukurti naują duomenų kopiją, TS sukuria papildomą BC. Naujo agregato sukūrimas reiškia, jog

sistemoje turi padidėti agregatų aibė A, visi BC agregato įėjimai ar išėjimai taip pat turi būti apibrėžti, juos priskiriant atitinkamai X ir Y aibėms.

Paveikslėlyje *4* aprašyta TS agregato specifikacija, kai jis sukuria dar vieną BC agregatą (BC2) ir sunaikiną baigusį darbą BC1 agregatą. Paveikslėlyje *5* aprašyta bendra BC agregato specifikacija.

$L1\ \ A_{t_0} = \{TS, BC1\}$

$L2\ \ TS:$

$L3\ \ H_{t_0} = \{H(e_0'), H(e_1')\}$

$L4\ \ G_{t_0} = \{G(e_0')\}$

$L5\ \ 1.X = \{x_0, x_1\}$

$L6\ \ 2.Y = \{y_0\}$

$L7\ \ 3.E' = \{e_0', e_1'\}$

$L8\ \ 4.E'' = \{\varnothing\}$

$L9\ \ 5.\varnothing$

$L10\ \ 6.W(t_0) = \{\varnothing\}$

$L11\ \ 7.v(t_0) = \{vnt(t_0)\}\ \ vnt(t_0) - BC\ \ agregat\ ų\ sk.$ mod *elyje*

$L12\ \ 8.vnt(t_0) = 1$

$L13\ \ 9.H(e_0') - ateina\ \ reikalavim\ \ as\ \ sukurti\ nauj ą\ \ duomen ų\ \ kopij ą$

$L14\ \ M_{t_0} = \{vnt(t_0) + 0, TS, y_1, BC1, x_1\}$

$L15\ \ M_{t_0} = \{vnt(t_0) + 1, BC1, y_1, TS, x_1\}$

$L16\ \ M_{t_0} = \{vnt(t_0) + 2, Agg\ 0, y_1, TS, x_0\}$

$L17\ \ M_{t_0} = \{vnt(t_0) + 3, BC1, y_2, Agg0, x_1\}$

$L18\ \ A_{t+1} = A_t \cup \{BCvnt(t) + 1\}$

$L19\ \ vnt(t+1) = vnt(t) + 1$

$L20\ \ X_{t+1} = X_t \cup \{x_{vnt(t+1)}\}$

$L21\ \ Y_{t+1} = Y_t \cup \{y_{vnt(t+1)}\}$

$L22\ \ M_{t+1} = M_t \cup \{vnt(t+1) + 3, TS, y_{vnt(t+1)}, BCvnt(t+1), x_1\}$

$L23\ \ M_{t+1} = M_t \cup \{vnt(t+1) + 4, BCvnt(t+1), y_1, TS, x_{vnt(t+1)}\}$

$L24\ \ M_{t_0} = M_t \cup \{vnt(t+1) + 5, BCvnt(t+1), y_2, Agg0, x_{vnt(t+1)}\}$

$L25\ \ H(e_1') - pranešimas\ \ apie\ \ darbo\ \ pabaigą$

$L26\ \ A_{t+1} = A_t - \{BCvnt(t+1)\}$

$L27\ \ X_{t+1} = X_t - \{x_{vnt(t+1)}\}$

$L28\ \ Y_{t+1} = Y_t - \{y_{vnt(t+1)}\}$

$L29\ \ M_{t+1} = M_t - \{vnt(t+1) + 3, TS, y_{vnt(t+1)}, BCvnt(t+1), x_1\}$

$L30\ \ M_{t+1} = M_t - \{vnt(t+1) + 4, BCvnt(t+1), y_1, TS, x_{vnt(t+1)}\}$

$L31\ \ M_{t_0} = M_t - \{vnt(t+1) + 5, BCvnt(t+1), y_2, Agg0, x_{vnt(t+1)}\}$

$L32\ \ vnt(t+1) = vnt(t) - 1$

$L33\ \ G(e_0'):$

$L34\ \ Y = \{y_{vnt(t+1)}\}$

4 pav. *TS agregato specifikacija.*

$L35 \quad BC:$

$L36 \quad H_{t_0} = \{H(e_1^{'})\}$

$L37 \quad G_{t_0} = \{G(e_1^{'})\}$

$L38 \quad 1.X = \{x_1\}$

$L39 \quad 2.Y = \{y_1, y_2\}$

$L40 \quad 3.E^{'} = \{e_1^{'}\}$

$L41 \quad 4.E^{''} = \{e_1^{''}\}$

$L42 \quad 5.\{e_1^{''}\} \mapsto \{\eta_k^{(1)}\}, k = \overline{1,\infty}$

$L43 \quad 6.W(t_m) = \{w(e_1^{''}, t_m)\}$

$L44 \quad 7.v(t_m) = \varnothing$

$L45 \quad 8.\varnothing$

$L46 \quad 9.H(e_1^{'}):$

$L47 \quad //\ veiksmai$

$L48 \quad G(e_1^{'}):$

$L49 \quad Y = \{y_1, y_2\}$

5 pav. *BC agregato specifikacija.*

L14-L17 eilutėse yra aprašyti pirminės sistemos sujungimų kanalai. Ryšiams tarp agregatų įėjimų ir išėjimų aprašyti naudojama M sujungimų matrica.

$M_{t+1} = M_t \cup \{1, A_{iš}, y_{iš}, A_{į}, x_{į}\}$ - sukuriamas sujungimas (1-as kanalas) tarp $A_{iš}$ agregato $y_{iš}$ išėjimo ir $A_{į}$ agregato $x_{į}$ įėjimo.

$M_{t+1} = M_t - \{1, A_{iš}, y_{iš}, A_{į}, x_{į}\}$ - pašalinamas sujungimas (1-as kanalas) tarp $A_{iš}$ agregato $y_{iš}$ išėjimo ir $A_{į}$ agregato $x_{į}$ įėjimo.

1 lentelė. Agregatų sujungimo matrica.

Kanalo Nr.	Iš agregato	Išėjimas	Į agregatą	Įėjimas
1.	TS	y_1	BC1	x_1
2.	BC1	y_1	TS	x_1
3.	Agg0	y_1	TS	x_0
4.	BC1	y_2	Agg0	x_1
5.	TS	y_2	BC2	x_1
6.	BC2	y_1	TS	x_2
7.	BC2	y_2	Agg0	x_1

Sujungimų matricą galima užrašyti taip kaip pavaizduota *3* pav. L14-L17 ir L22-L24 eilutėse, tačiau galimą užrašyti ir taip kaip pavaizduota Lentelėje 1.

Toks PLA formalizavimo kalbos išplėtimas leidžia aprašinėti modelius, kurių nesugebėtų aprašyti paprasta PLA kalba.

Tačiau kadangi dynPLA yra naujai sukurta formalizavimo kalba, jai dar nėra sukurtų verifikavimo įrankių.

2.3 DEVS formalizavimo kalba

DEVS (Discrete EVent System specification) – tai formalizavimo kalba skirta aprašyti sistemas, kurių įėjimo/išėjimo parametrų elgesį apibūdina įvykių seka su sąlyga, jog būsenų skaičius gali kisti baigtiniame laiko intervale, baigtiniu skaičiumi.

DEVS aprašo statinę sistemos stuktūrą, jos elgseną, kuri vykdo struktūrinius pasikeitimus. Nors modelis, kurio iėjimų, išėjimų ir būsenų elgsena ekvivalenti kintamos struktūros modeliui gali būti aprašyta DEVS, tačiau DEVS formalizavimo metodas neaprašo struktūrinių pasikeitimų.

Būtent struktūriniams pasikeitimams aprašyti ir buvo įvestas dynDEVS.

Norint suprasti dynDEVS formalizmo kalbos ypatumus reikia apibrėžti standartinį DEVS modelį.

2.3.1 Atominis DEVS modelis

Atominis DEVS modelis pagal [3] aprašomas tokia struktūra:

$DEVS = < X, Y, S, \delta_{\text{int}}, \delta_{ext}, \lambda, ta >$

čia:

X - įėjimų aibė,

Y - išėjimų aibė,

S - būsenų aibė,

$\delta_{\text{int}} : S \to S$ - vidinė perėjimo funkcija,

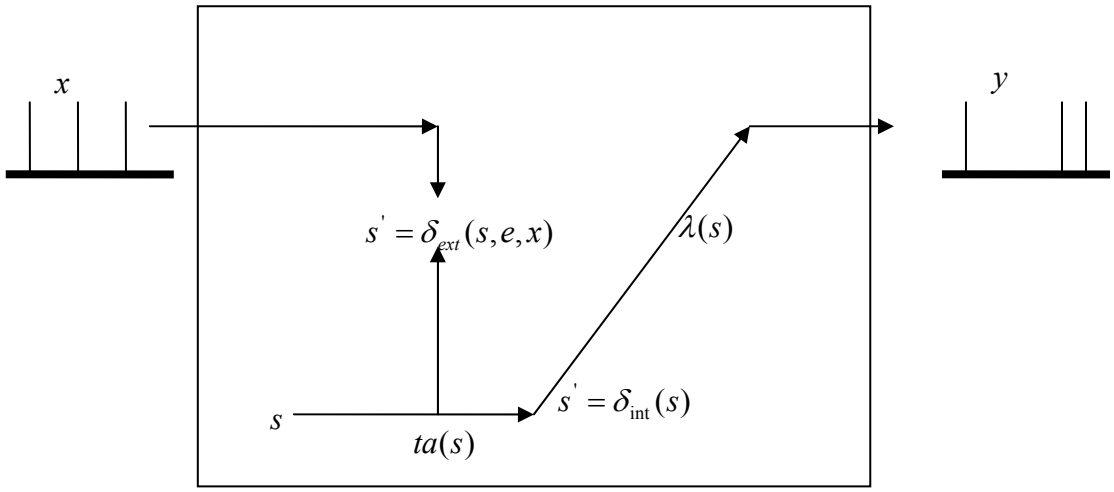
$\delta_{ext} : Q \ x \ X \to S$ - išorinė perėjimo funkcija, kaip $Q = \{(s,e) | s \in S \ ir \ 0 \le e \le ta(s)\}$, kur e – praėjęs laikas.

$\lambda : S \to Y$ - išėjimo funkcija,

$ta : S \to R^{+}_{0,\infty}$ - laiko eigos funkcija.

Atominis DEVS modelis aprašomas kaip įėjimų, būsenų, išėjimų aibėmis bei vidine ir išorine perėjimo funkcija, išėjimo ir laiko eigos funkcija. Vidinė perėjimo funkcija žymi būsenų perėjimus priklausomai nuo vidinių įvykių, kurių veikimo laikas aprašytas laiko eigos funkcija.

Išorinio įvykio metu, modelis pateikia išėjimą. Išorinė perėjimo funkcija sužadinama išorinių įėjimų.



6 pav. *DEVS funkcija.*

DEVS funkcija galima paprastai pavaizduoti kaip tai parodyta paveikslėlyje *6*.

Tarkim sistema yra būsenoje *s*. Laiko kitimo funkcija *ta* (time advanced), nustato kaip ilgai egzistuos esama būsena iki kol bus sugeneruotas vidinis įvykis.

Sistema laiko tarpą *ta(s)* bus būsenoje *s,* kol nepasirodys joks išorinis įvykis. Praėjęs laikas *e* žymi kaip ilgai sistema stovi dabartinėje būsenoje. Jei esamoj būsenoj praleistas laikas lygus *ta*, tai yra $e = ta(s)$, tai sugeneruojamas išėjimas $\lambda(s)$ ir būsena *s* pereina į būsena *s'*, kuri aprašyta $\delta_{\text{int}}(s)$ funkcija. Išėjimas sugeneruojamas prieš vidinės būsenos pasikeitimą.

Jei įvyksta išorinis įvykis, kol $e < ta(s)$, pasikeičia būsena. Nauja būsena apskaičiuojama pagal $\delta_{ext}(s,e,x)$ funkciją, kuri priklauso nuo įėjimo *x*, dabartinės būsenos *s,* ir laiko *e.* Pasikeitus būsenai iš *s* į *s'*, sistema veikia taip pat kaip ir prieš tai. Taip pat būsenai *s'* laikas nustatomas *ta(s')*, kuris nusako kada įvyks vidinis perėjimas su ta sąlyga jog įėjimo signalas neateis anksčiau.

Kad supaprastint DEVS formalizavimo kalbą reikia pakeisti standartinius įėjimus/išėjimus specifiniais įėjimais/išėjimais.

$$DEVS = < X, Y, S, \delta_{\text{int}}, \delta_{ext}, \lambda, ta >$$

čia:

$X = \{(p,v) \mid p \in IP, v \in X_p\}$ - įėjimo portų aibė ir įėjimai(IP yra įėjimų portų aibė, X_p – įėjimų aibė).

$Y = \{(p,v) \mid p \in OP, v \in Y_p\}$ - išėjimo portų aibė ir išėjimai(OP yra išėjimų portų aibė, Y_p – išėjimų aibė).

S - būsenų aibė,

$\delta_{\text{int}} : S \to S$ - vidinė perėjimo funkcija,

$\delta_{ext} : Q \ x \ X \to S$ - išorinė perėjimo funkcija, kaip $Q = \{(s,e) \mid s \in S \ ir \ 0 \le e \le ta(s)\}$, kur e – praėjęs laikas.

$\lambda : S \to Y$ - išėjimo funkcija,

$ta : S \to R^+_{0,\infty}$ - laiko eigos funkcija.

Šio DEVS modelio veikimo principas visai nesiskiria nuo anksčiau aprašytojo, tiesiog formalizmas yra labiau sukonkretintas. Šio modelio įėjimų ir įėjimų portų aibės atitinka vieną įėjimą su išoriniais įvykiais klasikiniame DEVS modelyje.

2.3.2 Jungtinis DEVS modelis

Jungtinis DEVS modelis – tai modelis susidedantis iš skirtingų komponentų, apibrėžtų modelio jungtimis.

Jungtinis DEVS modelis [4] aprašomas tokia formule:

$DEVS = < X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, Select >$

čia:

$X = \{(p,v) \mid p \in IP, v \in X_p\}$ - įėjimo portų aibė ir įėjimai(IP yra įėjimų portų aibė, X_p – įėjimų aibė).

$Y = \{(p,v) \mid p \in OP, v \in Y_p\}$ - išėjimo portų aibė ir išėjimai(OP yra išėjimų portų aibė, Y_p – išėjimų aibė).

D - komponentų vardų aibė,

M_d - komponentų aibė, kurių kiekvienas DEVS funkcija,

EIC - išorinių įėjimų apjungimų aibė,

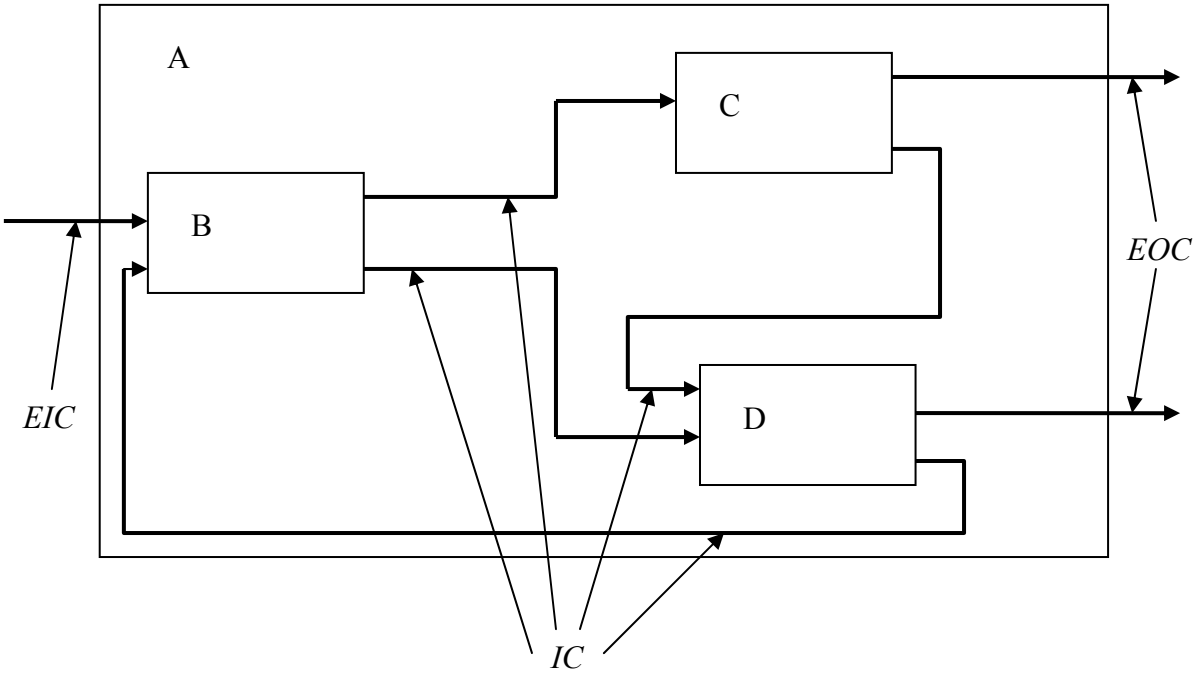
EOC - išorinių išėjimų apjungimų aibė,

IC - vidinių apjungimų aibė,

$Select : 2^D \to D$ - ryšių nutraukimo funkcija.

Jungtis DEVS modelis susideda iš keleto DEVS modelių, kurie jungtiniame DEVS modelyje tampa komponentais. Aibė M_d apibrėžiama kaip $M_d = < X_d, Y_d, S, \delta_{\text{int}}, \delta_{ext}, \lambda, ta >$, kur $d \in D$.

Kiekvienas iš jungtinių DEVS modelių gali būti pakeistas atominiu. Ši aplinkybė, dar vadinama apjungimų vieningumu, leidžia tobulinti modelius bei jų aprašus skirtingais abstrakcijos lygiais. Naujiems ryšiams tarp individualių komponentų, taip pat tarp komponentų bei išorinių įėjimų ar išėjimų, naudojami nauji pažymėjimai (EIC, EOC ir IC). EIC – yra sujungimai tarp išorinių sistemos įėjimų ir išorinių komponentų įėjimų. EOC – tai sujungimai tarp išorinių komponentų išėjimų bei sistemos. Tuo tarpu IC – tai vidiniai sujungimai tarp komponentų, nustatant sąryšius tarp jų išėjimų bei įėjimų. Aptartieji sujungimai pavaizduoti *7* paveiksle.



7 pav. *Sujungimų tipai.*

Kadangi tuo pačiu metu įdiegti keleto išorinių įvykių yra neįmanoma, tai naudojama ryšių nutraukimo funkcija. Ji nustato individualių komponentų prioritetus, jei keli iš jų nori vykdyt išorinį įvykį.

2.3.3 Dinaminis DEVS modelis

Dinaminio DEVS formalizmas buvo sukurtas sistemoms, kurios sugeba pritaikyti savo elgesį, struktūrą ir tarpusavio sąveikavimus naujai iškilusiems, lankstesniems reikalavimams.

Dinaminis DEVS, tai klasikinio DEVS modelio plėtinys, tačiau priešingai nei jo pirmtakas yra pritaikytas aprašyti struktūrom, kurių būsenos turi polinkį kisti.

Standartiniame DEVS modelyje perėjimai seka būsenos pasikeitimus tik modelyje, o dinaminiame DEVS perėjimai priverčia pradėti naujus modelius – modelis pakeičia savo struktūrą.

Automatinis dynDEVS modelis aprašomas taip:

$dynDEVS =< X, Y, m_{init}, M(m_{init}) >$

čia:

X - įėjimų aibė,

Y - išėjimų aibė,

m_{init} - pradinis modelis, kur $m_{init} \in M(m_{init})$,

$M(m_{init})$ - pati mažiausia modelių turinčių formą $\{< S, s_{init}, \delta_{ext}, \delta_{\text{int}}, \rho_\alpha, \lambda, ta >\}$ aibė.

$M(m_{init}) = \{< S, s_{init}, \delta_{ext}, \delta_{\text{int}}, \rho_\alpha, \lambda, ta >\}$,

kur:

S - būsenų aibė,

s_{init} - pradinė būsena, kur $s_{init} \in S$,

$\delta_{ext} : Q \ x \ X \to S$ - išorinė perėjimo funkcija, kur $Q = \{(s,e) | s \in S \ ir \ 0 \le e \le ta(s)\}$, e – praėjęs laikas.

$\delta_{\text{int}} : S \to S$ - vidinė perėjimo funkcija,

$\rho_\alpha : S \to M(m_{init})$ - modelių perėjimo funkcija,

$\lambda : S \to Y$ - išėjimo funkcija,

$ta : S \to R^+_{0,\infty}$ - laiko eigos funkcija.

Taip pat turi būti išpildyta sekanti sąlyga:

$\forall n \in M(m_{init}).(\exists m \in M(m_{init}).n = \rho_\alpha(s^m), \quad kur \quad s^m \in S^m) \vee n = m_{init}$

Taip pat dynDEVS turi išpildyti šį apribojimą: modelio perėjimas ρ_α netrukdo kitiems perėjimams, jis išsaugo kintamųjų reikšmes, kurie yra bendri dviems einantiems vienas po kito modeliams ir priskiria pradines reikšmes naujiems kintamiesiems.

$$i(S^m) = s^m \wedge \rho_\alpha(s^m) = n \wedge i(S^n) = s^n \to \forall d \in D^m \cap D^n . s^m_d = s^m_d \wedge \forall d \in D^n \setminus D^m . s^n_d = s_{init_d}$$

S^m ir S^n yra sekų aibė, kur $m \in M(m_{init})$ ir $n \in M(m_{init})$. Jei $m \ne n$ tada $s^n = s^n_{init}$, priešingu atveju $s^n = s^m$. Kaip ir DEVS, dinaminis DEVS apibrėžia būsenas kaip

struktūrizuotas sekas, kur D^m ir D^n yra kintamųjų vardų aibė ir *i* apibrėžiančioji funkcija, kuri apibrėžia reikšmes kintamiesiems.

Modelio būsenų aibė, vidinio ir išorinio perėjimų, išėjimo, laiko eigos ir modelio perėjimo funkcijos gali kisti. Dinaminis DEVS gali būti interpretuojamas kaip DEVS modelių grupė, kur yra sąsajos ir perėjimo funkcija, kuri sąlygoja kuris DEVS modelis pakeičia prieš jį veikusį modelį, priklausomai nuo esamos padėties [5]. Funkcijos kurios egzistavo ir standartiniame DEVS modelyje išliko tokios pačios. Kadangi kiekvienas veiksmas priklauso nuo modelio esamos būsenos, grupė modelių pradedančių veikti nauju pavidalu sudaro pogrupį iš modelių, gautų per pirminį modelio pavidalo pakeitimą m_{init} .

Keičiant modelio pavidalą yra iššaukiamas dinaminis DEVS modelis $< X, Y, m, M(m) >$, kur $M(m) \subseteq M(m_{init})$, t.y., dynDEVS formalizavimo kalba aprašomi modeliai, tokie, kurie bus įvykus tam tikram perėjimui.

Kaip aprašyta [6] dinaminis DEVS modelis suprantamas kaip paprastų DEVS modelių, turinčių tą pačią sąsają, aibė su papildoma perėjimo funkcija. Ši perėjimo funkcija apsprendžia į kurią būseną pereis modelis. Įėjimas, išėjimas ir būsenų aibė, o taip pat vidinė bei išorinė perėjimo funkcija bei laiko eigos funkcija dinaminiam DEVS modelyje aprašoma taip pat kaip ir klasikiniame.

2.4 dynDEVS ir dynPLA palyginimas

Kiekvienas dinaminis DEVS modelis evoliucionuoja kaip DEVS modelių seka. Dinaminio DEVS modelio dinamiškumas pasiekiamas naudojant atskirus modelius, skirtus kiekvienai sistemos struktūrai, tam tikru laiko momentu, aprašyti. Tuo tarpu dynPLA sistemos struktūra saugoma aibėje *A*. Šis metodas yra optimalesnis, nes nereikalauja skirtingoms sistemos struktūroms kurti atskirų specifikacijų, o tiesiog leidžia nuosekliai aprašyti sistemos agregatuose vykstančius veiksmus, nurodant tų perėjimų, įėjimų/išėjimų sąlygojamus rezultatus.

Dar vienas skirtumas egzistuojantis tarp šių formalizavimo kalbų yra tas, jog išėjimas dynPLA kalboje gali būti sugeneruojamas ir vidinio įvykio ir įėjimo signalo, tuo tarpu dynDEVS kalboje išėjimas gali būti generuojamas tik vidinio įvykio metu. Tokiais atvejais, kai įėjimo signalas turi generuoti išėjimą, dynDEVS formalizmo kalboje reikia kurti papildomus įvykius.

Nors šios formalizavimo kalbos yra gana panašios, tačiau peržvelgus abiejų kalbų principus bei ypatumus, galima pastebėti, jog dynPLA naudojimas yra patogesnis.

3. Mobilaus ryšio sistema

dynPLA tai labai jauna formalizavimo kalba, kurios nagrinėjimas ir galimybės pateiktos [2] straipsnyje. Tačiau ten yra pateikiamas tik nedidelis dinaminės sistemos pavyzdys, nagrinėjantis nesudėtingą sistemą. Išnagrinėjus tokį pavyzdį (nors dynPLA ir susidoroja su jai keliamais reikalavimais) negalime teigti, jog formalizavimo kalba sugebėtų aprašyti pilną dinaminę sistemą.

Kad įsitikintume, jog dynPLA formalizavimo kalba sugeba aprašyti laike kintančias sistemas buvo pasirinkta mobilaus ryšio sistema, turinti laike kintančių struktūrų. Tai sistema, kurioje duomenų perdavimas, objektų judėjimas, atsiradimas ar išnykimas nėra valdomi kažkokio konkretaus įrenginio, tačiau yra veikiami daugybės skirtingų aplinkybių. Tokia sistema turi visą gaubiantį agregatą, kuris nors nėra valdantysis, tačiau reikalui esant gali atskiriems sistemos agregatams suteikti reikiamą informaciją.

Jei yra aprašoma sistema, kurios vienas agregatas yra valdantysis, t. y. jis žino visus reikiamus įėjimus bei išėjimus, norint sudaryti sujungimų matricą, tokiu atveju, sistemos specifikavimas yra paprastas. Tačiau jei agregatai neturi pakankamai informacijos apie vienas kitą, atsiranda būtinybė kažkaip sužinoti reikiamą informaciją.

Nagrinėjama mobilaus ryšio sistema susidedanti iš trijų pagrindinių objektų turi tokią savybę, jog veiksmai vykstantys tarp tų objektų nėra pilnai valdomi vieno iš jų. Šie veiksmai gali būti inicijuojami vieno agregato, ar į tą agregatą atėjusio išorinio įvykio, o vykdomi kito agregato. Lygiai taip pat vykdymą gali atlikt keli agregatai pasikeisdami.

Bandant aprašyti visą sistemą neišskiriant nei vieno esminio objekto, labai greitai susipainiojama tarp begalės įvykių ir signalų. Tokias sistemas aprašinėjant būtina išskirti ir atskirai aprašyti pagrindinius sistemą sudarančius agregatus.

Atskirai aprašant sistemą sudarančius agregatus, yra išnagrinėjami tik tuose agregatuose vykstantys įvykiai, ar tų agregatų reakcija į atėjusius signalus, tačiau pats bendravimas tarp agregatų, informacijos siuntimas ar atsirandantys kanalai yra visiškai neaprašomi. Yra žinoma, jog signalas iš vieno agregato ateina į kitą, tačiau nei vienas iš jų negali aprašyt to signalo judėjimo, ar kanalo kuris jam sukuriamas tarp tų agregatų. Tai yra neįmanoma, nes agregatai neturi pakankamai informacijos apie tai iš kur tie signalai ateina, t. y. nežino tikslių kito agregato įėjimų/išėjimų.

Tačiau aprašyti veiksmus vykstančius tarp sistemos agregatų yra būtina, todėl šiam tikslui pasiekti teko papildyti sistemos modelį, sukuriant visą apimantį agregatą. Toks agregatas reikalui esant gali suteikti papildomą informaciją kitiems agregatams. Taip pat, kadangi jis yra

sistemą apimantis agregatas būtent jame reikia aprašyti visus tarp kitų sistemos agregatų vykstančius veiksmus.

Šiuo atveju gaubiantysis agregatas – mobiliojo ryšio tinklas.

Toliau aprašoma nagrinėtoji sistema.

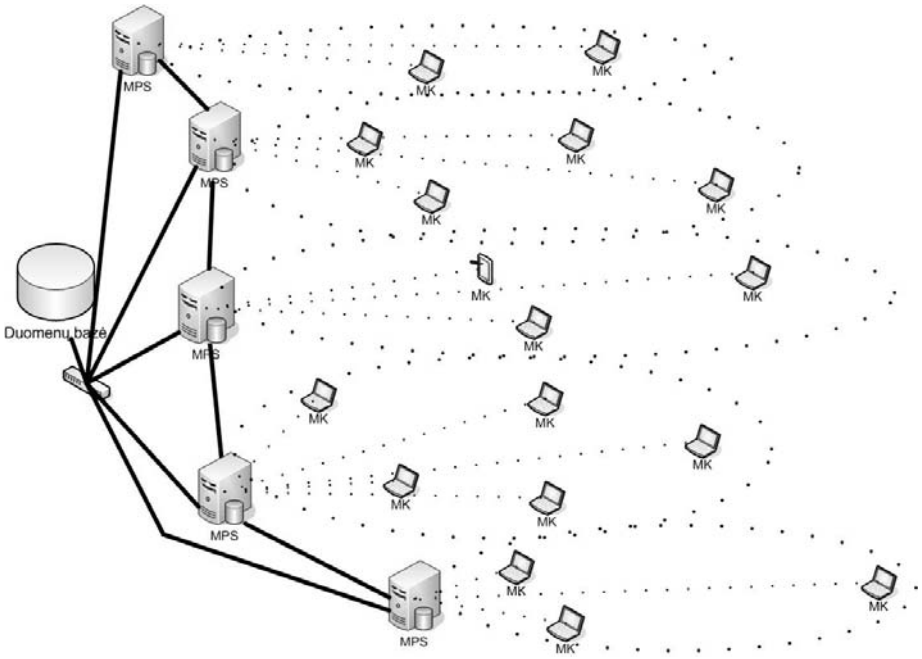
3.1 Mobilaus ryšio sistemos struktūra

Aprašoma mobilaus ryšio sistema, susidedanti iš trijų pagrindinių komponentų: mobilaus kompiuterio, palaikymo stoties bei duomenų bazės. Tokia sistema yra dinaminio tipo, kur ryšiai tarp mobilių kompiuterių ir tarp palaikymo stočių, kurios susietos beviele sąsaja su kompiuteriais, gali dinamiškai kisti. Šioje mobilaus ryšio sistemoje, vartotojai turintys mobilius(nešiojamus) kompiuterius gali prieiti prie duomenų bazės iš bet kurios vietos, nereikalaujant fiksuoto ryšio.

Sistemoje egzistuoja du pagrindiniai veiksmai: mobilaus kompiuterio judėjimas ir persijungimas prie kitos mobilios palaikymo stoties (į šį veiksmą taip pat įtraukiamas mobiliojo kompiuterio įsijungimas į sistemą bei atsijungimas nuo jos) bei transakcijų vykdymas, kai sistemoje yra vykdomas mobiliojo kompiuterio užklausimas duomenų bazėje esantiems duomenims.

Dėl mobilių tinklų sistemų resursų apribojimų yra labai sudėtinga patenkinti realaus laiko sistemai keliamus reikalavimus [7]. Mažas pralaidumas, nepatikimas bevielis internetas, nuolatiniai atsijungimai nuo sistemos reikalauja daugiau informacijos, kad palaikyt ryšį tarp mobilaus kompiuterio ir fiksuotos sistemos dalies. Vartotojo kompiuterio mobilumas padaro kompiuterio būvimo koordinates dinamine informacija, priversdamas nuolatos atlikinėti naujas koordinačių paieškas.

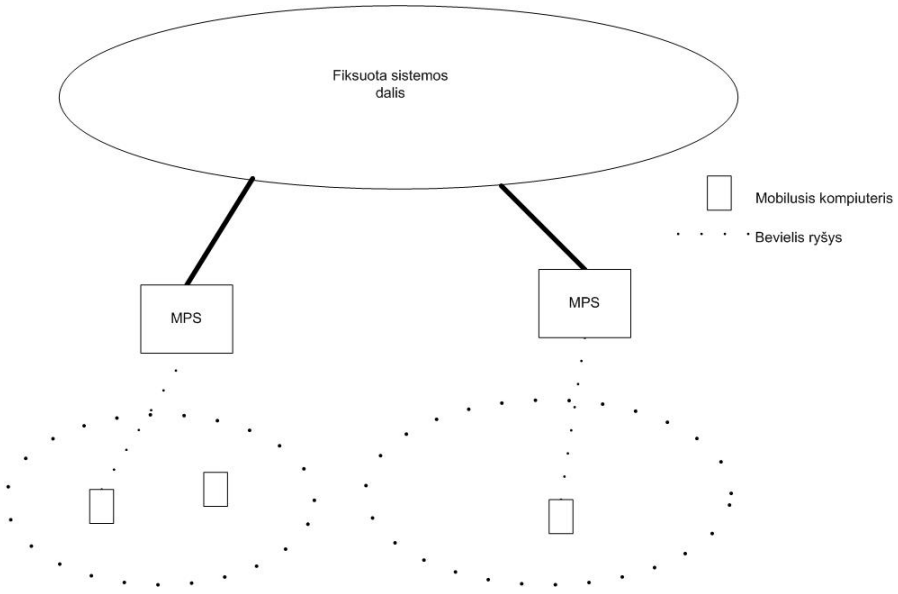
Atsižvelgti į visus apribojimus yra beveik neįmanoma, todėl tariame jog prasto pralaidumo, nepatikimo bevielio interneto ar netyčinio atsijungimo galimybės yra neįmanomos. Sistemoje visi veiksmai atliekami be pertraukimų ar klaidų, o būtent taip kaip numatyta.



8 pav. *Dinaminė kompiuterinių tinklų sistema.*

Tipiška dinaminė mobilaus ryšio sistema susideda iš daugybės mobilių ir fiksuotų kompiuterių (paveikslėlis *8*). Skirtumas tik tas, jog mobilieji kompiuteriai (MK) gali judėti sistemoje, išnykti bei vėl atsirasti (kompiuterio išsijungimas/ įsijungimas) kai tuo tarpu fiksuotą sistemos dalį sudarantys kompiuteriai (serveriai) neturi judėjimo galimybės. Taip pat yra tariama, jog fiksuotą sistemos dalį sudarantys aparatai negali išsijungti. Jie yra nuolatos egzistuojantys aparatai (fiksuotos sistemos dalies kompiuterio atsijungimas dėl gedimų tokioje sistemoje traktuojamas neįmanomu).

Mobilaus ryšio sistemos supaprastintas modelis atrodytų taip:



9 pav. *Mobilios ryšio sistemos supaprastintas modelis.*

Dinaminė tinklų sistema [8], tokia kaip parodyta paveikslėlyje *8* ir paveikslėlyje *9* susideda iš dviejų dalių: fiksuotos ir mobilios sistemos dalių.

Kompiuteriai, kurie jungiasi tarpusavyje greitu laidiniu ryšiu ir sudaro fiksuotą sistemos dalį, vadinami fiksuotais kompiuteriais (FK). MK – mobilus kompiuteris sugebantis jungtis su fiksuota sistemos dalimi bevieliu internetu. Dauguma fiksuotų kompiuterių (FK) yra mobilios palaikymo stotys – MPS

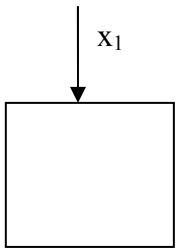
Mobiliosios palaikymo stotys, tai fiksuotos sistemos dalyje esantys kompiuteriai, kurie turi bevielę sąsają tam kad galėtų bendrauti su mobiliais kompiuteriais. Ryšiai tarp MPS ir MK gali dinamiškai kisti. Geografinis taškas, kuriame būdamas MK gali bendrauti su MPS vadinamas to MPS cele. Mobilus kompiuteris priskiriamas atitinkamai tam MPS, kurio celėje jis yra.

Esant daug palaikymo stočių ir jų celių, kompiuteris (MK) priskiriamas tai palaikymo stočiai (yra to MPS celėje), kurios skleidžiamas signalas yra stipresnis. T.y., mobilusis kompiuteris (MK) gauna signalus iš visų aplinkui esančių palaikymo stočių (MPS) ir jungiasi prie tos, kurios signalas yra stipriausias.

3.2 Atskirų sistemos objektų aprašas

3.2.1 Mobilusis kompiuteris

Tai paprastas nešiojamas kompiuteris arba delninukas, kurio kordinatės gali kisti. Vienas iš pagrindinių reikalavimų, keliamų nagrinėjamos sistemos mobiliajam kompiuteriui yra tas, jog jis turi būti prijungtas prie sistemos bevieliu internetiniu ryšiu.



10 pav. *Elementarus mobiliojo kompiuterio modelis.*

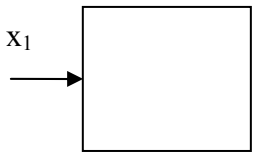
Nagrinėjant paprasčiausią mobiliojo kompiuterio (MK) konceptualųjį modelį, tokį koks pavaizduotas *10* paveikslėlyje, galima pastebėti, jog jis turi vienintelį įėjimą (x_1). Tai yra išorinis signalas reiškiantis kompiuterio įjungimo įvykį.

Pradiniu laiko momentu kiti įėjimai ar išėjimai neegzistuoja. Pirmiausia reikia, kad kompiuteris (MK) būtų įjungiamas, tada šio pradinio įvykio inicijuojami atsiranda papildomi įėjimai/išėjimai. Šie įėjimai/išėjimai naudojami tokiems veiksmams kaip prisijungimas prie sistemos, perėjimas į kitą celę ar transakcijų vykdymui aprašyti.

Todėl išsamiau bus aptarti prie minėtų veiksmų.

3.2.2 Mobilioji palaikymo stotis

Mobiliosios palaikymo stotys (MPS) – tai serveriai esantys fiksuotoje sistemos dalyje.



11 pav. *Elementarus mobiliosios palaikymo stoties modelis pradiniu laiko momentu.*

Kaip pavaizduota paveikslėlyje *11* pradiniu laiko momentu palaikymo stotis turi vienintelį įėjimą (x_1). Tai yra įėjimas, į kurį ateina mobiliojo kompiuterio (MK) prašymas įsijungti į sistemą. Įsijungus vienam MK, sukuriamas naujas įėjimas (x_2, x_3, ..., x_n) sekantiems kompiuteriams (MK). Papildomi įėjimai/išėjimai sukuriami reikalui esant ir išsamiau bus aptarti nagrinėjant sistemoje vykstančius įvykius, kuriuose mobili palaikymo stotis (MPS) atlieka vieną iš pagrindinių vaidmenų.

Mobilios palaikymo stotys (MPS) viena su kitom sujungtos laidiniu ryšiu, taip pat turi laidinį ryšį su sistemos duomenų baze, iš kurios, prie MPS prisijungęs mobilusis kompiuteris (MK) gali prašyti tam tikrų duomenų.

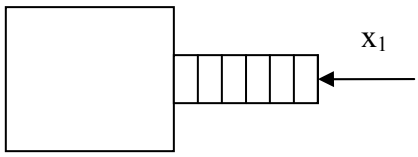
MPS palaiko ryšį tarp jo celėje esančio mobiliojo kompiuterio ir fiksuotos sistemos dalies. O taip pat yra atsakinga už duomenų ir žinučių persiuntimą tarp MK ir fiksuotos sistemos dalies. Mobilaus kompiuterio (MK) adresas traktuojamas kaip duomenys, kurie gali keistis, kai vartotojas peržengia gretimas celes skiriančią liniją.

Šiame modelyje yra tariama, jog visi fiksuotos dalies kompiuteriai yra mobilios palaikymo stotys (MPS).

3.2.3 Duomenų bazė

Trečias mobilaus ryšio sistemoje esantis objektas yra duomenų bazė. Nors realiai gali egzistuoti tūkstančiai skirtingų duomenų bazių, veikiančių tame pačiame tinkle [9], šiame modelyje tariame jog duomenų bazė yra vienintelė.

Palyginus su tradicinėmis duomenų bazėmis, ši duomenų bazė, veikianti realiu laiku turi vieną išskirtinę savybę: saugo laikinus duomenis, kol yra atliekama laiko apribojimais suvaržyta transakcija. Duomenų bazė turi išsiųsti atsakymą į gautą užklausą tenkindama transakcijai keliamus laiko apribojimus [10], vadinamus laiko terminais (*deadline*).



12 pav. Supaprastintas duomenų bazės modelis.

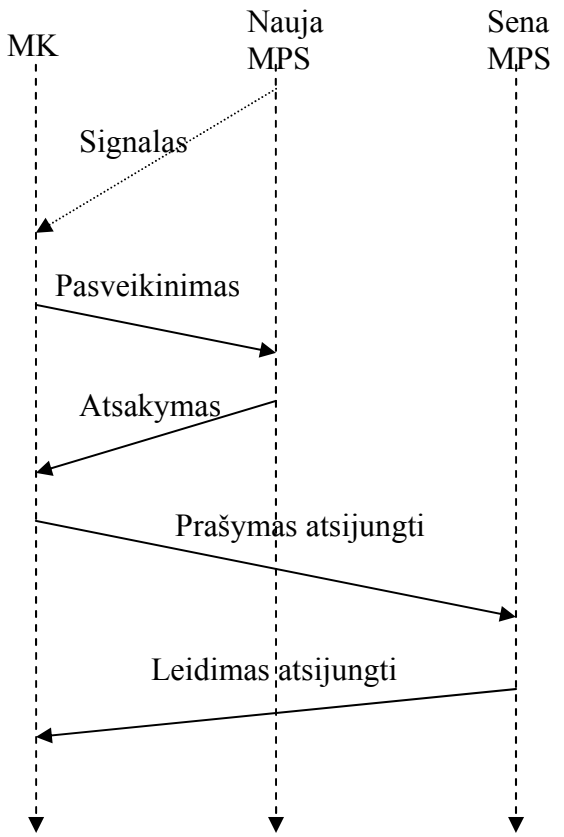
Kaip pavaizduota paveikslėlyje 12, supaprastintai duomenų bazę galima žymėti kaip paprasčiausią agregatą turintį vieną įėjimą. Per minėtąjį įėjimą ateina visos mobiliųjų kompiuterių (MK) sugeneruotos ir mobiliųjų palaikymo stočių (MPS) valdomos užklausos.

Kadangi sistemoje gali būti neviena mobilioji palaikymo stotis (MPS), tai visos jų siunčiamos užklausos stoja į eilę. Eilė yra aptarnaujama FIFO (*first in first out*) metodu. Papildomi išėjimai, tokie kaip sugeneruoto atsakymo siuntimui sukuriami reikalui esant ir bus aptariami vėliau, aprašant sistemoje vykstančius įvykius.

3.3 Perėjimo procesas (*handoff*)

Mobilieji vartotojai gali bet kada peržengti gretimas celes skiriančią ribą. Tam, kad nenutrūktų ryšys tarp mobiliojo vartotojo ir fiksuoto įrenginio yra naudojamas perėjimo (*handoff*) procesas [11]. Šio proceso metu MPS, kuriam priklauso naujoji celė prisiima visą atsakomybę už bevielės sąsajos tiekimą mobiliajam vartotojui. Mobilusis kompiuteris saugo MPS, prie kurios jungęsi adresą, tam jog jei perėjimo metu yra vykdoma transakcija, galėtų pranešti senajai savo palaikymo stočiai, kur siųsti atėjusį transakcijos rezultatą.

Judėjimo proceso metu mobilusis kompiuteris (MK) gavęs stipresnį signalą iš naujos MPS, nei iš dabartinės (tos, prie kurios prisijungęs) inicijuoja perėjimą tarp palaikymo stočių.



13 pav. *Perėjimo proceso schema.*

Mobiliajam kompiuteriui judant tarp gretimų celių vyksta procesai pavaizduoti paveikslėlyje *13*

Kiekviena palaikymo stotis (MPS) transliuoja signalą (*beacon*) [12]. Kiekvienas palaikymo stoties signalas turi jį siuntusios MPS adresą. Kai MK atsiduria celių persidengimo vietoje, MK gauna abiejų MPS signalus.

Remiantis signalo stiprumu, MK gali inicijuoti perėjimo (*handoff*) procesą nuo dabartinės palaikymo stoties prie naujos. Perėjimo procesas vyksta taip:

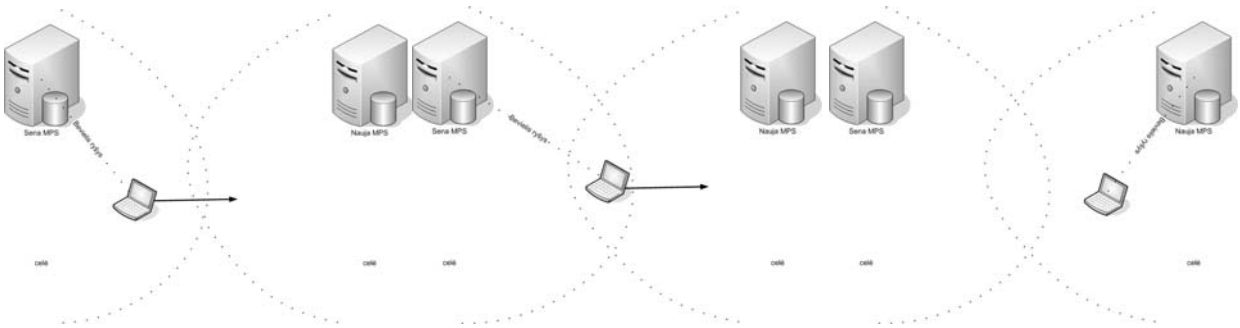
1. MK gauna naujos MPS signalą (*beacon*), kuriame yra mobiliosios palaikymo stoties adresas (šiuo atveju adresas traktuojamas kaip MPS numeris, kuris gali būti reikalingas senąjai stočiai, jei mobiliojo kompiuterio persijungimo metu buvo vykdoma transakcija).
2. Jei gautas signalas pasirodo esąs stipresnis nei dabartinės mobilios palaikymo stoties, tai MK siunčia pasveikinimo žinutę naujajai MPS, prašydamas leidimo prisijungti.
3. Naujoji MPS atsako į pasveikinimo žinutę leisdama prisijungti (šioje sistemoje tariama, jog neigiamas atsakymas į prašymą prisijungti yra neįmanomas).

4. Prisijungimo prie naujos mobiliosios palaikymo stoties metu, MK nusiunčia pranešimą senajai palaikymo stočiai prašydama atsijungimą
5. Atsakydama į gautą prašymą atsijungti, stotis nusiunčia atsijungimo patvirtinimą ir likviduoja visus su atsijungusiu kompiuteriu susijusius įėjimus/išėjimus.

Perdavimo procesas inicijuojamas mobiliojo kompiuterio. Tai natūralu, nes jis geriausiai gali spręsti apie prisijungimo prie palaikymo stočių kokybę. Mobilusis kompiuteris (MK) atlieka daugumą veiksmų, tačiau jis nėra valdantysis aparatas. Signalus, kuriuos siunčia MPS, MK gauna iš tinklo, o ne tiesiogiai iš MPS, nors mobilusis vartotojas to nepastebi. Taip pat visi MK bendravimai su MPS vyksta per tinklą, todėl būtent ten ir aprašomas perėjimo („handoff“) procesas.

Magistriniame darbe, perėjimo proceso formalusis aprašymas, tai tinklo aprašymo dalis, kuri naudoja prieš tai aprašytus sistemos sudedamuosius komponentus, t.y. mobiliąją palaikymo stotį, bei mobilųjį kompiuterį.

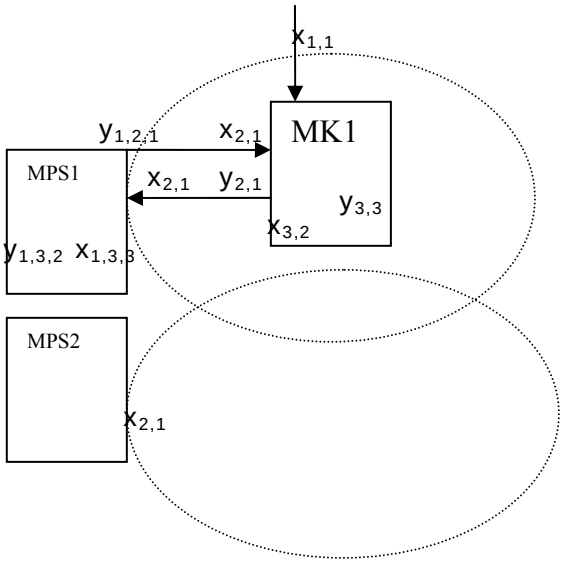
14 paveikslėlyje pavaizduotas persijungimas prie naujos palaikymo stoties, nekreipiant dėmesio į kitus įvykius.



14 pav. *Perėjimo proceso pagrindinės dalys.*

Visas perėjimo procesas, įskaitant kompiuterio įsijungimą į sistemą ir išsijungimą iš jos, skaidomas į tris pagrindines dalis.

1. Tariama, jog pradiniu laiko momentu mobilusis kompiuteris gauna įsijungimo signalą (vartotojas paspaudžia kompiuterio įjungimo mygtuką).



15 pav. *Pradinė perėjimo proceso dalis.*

Po to kai kompiuteris yra įjungiamas, tikrinami MPS siunčiamų signalų stiprumai bei mobiliajame kompiuteryje sukuriami reikiami įėjimai, išėjimai:

$Y(t_{m+1}) = Y(t_m) \cup \{y_{2,1}\}$ – įtraukiamas išėjimas, kuris bus naudojamas prisijungimo prie MPS prašymo siuntimui, į agregatą.

$MK_1(st_k(t_{m+1})) = \{\eta_k\}, kai \quad k = \overline{1, mps(t_m)}$ – apskaičiuojami visi MPS signalų stiprumai nagrinėjamam mobiliajam kompiuteriui.

$MK_{1i}(mps\_naujas(t_{m+1})) = k, kai \quad \max(MK_1(st_k(t_{m+1}))), k = \overline{1, mps(t_m)}$ – randamas MPS turintis stipriausią signalą.

$X(t_{m+1}) = X(t_m) \cup \{x_{2,1}\}$ – į agregatą įtraukiamas įėjimas (prisijungimo prie MPS patvirtinimo signalui gauti).

$M(t_{m+1}) = M(t_m) \cup \{MK1, \quad y_{2,1}, \quad MPS1, \quad x_{2,1}\}$ – sukuriamas kanalas MK1 prisijungimo prašymo žinutei siųsti.

$Y(t_{m+1}) = Y(t_{\mathrm{m}}) \cup \{y_{1,2,1}\}$ – MPS gavęs prašymą prisijungti sukuria išėjimą prašymo atsakymui nusiųsti.

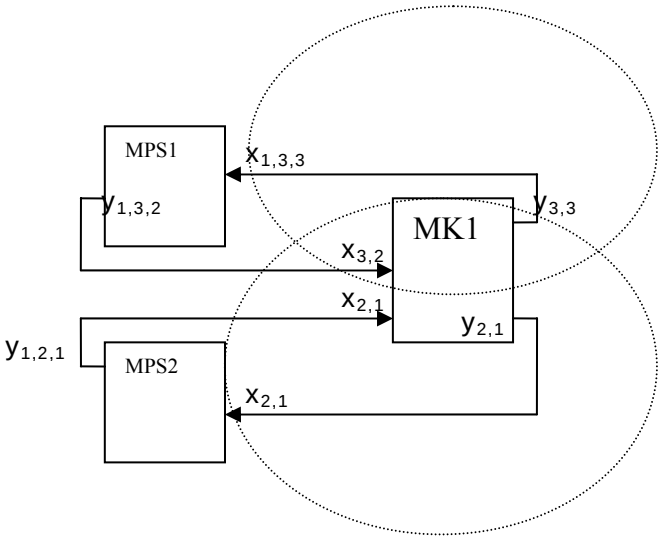
$X(t_{m+1}) = X(t_m) \cup \{x_{1,3,3}\}$ – taip pat sukuria įėjimą bei išėjimą $Y(t_{m+1}) = Y(t_{\mathrm{m}}) \cup \{y_{1,3,2}\}$, kurie bus naudojami tolimesniems bendravimams tarp MPS ir MK

Analogiški įėjimai bei išėjimai taip pat sukuriami ir mobiliajame kompiuteryje.

$M(t_{m+1}) = M(t_m) \cup \{MPS1, \quad y_{1,2,1}, \quad MK1, \quad x_{2,1}\}$ – sukuriamas kanalas tam, kad MPS1 nusiųstų prisijungimo patvirtinimo žinutę.

Kai mobilusis kompiuteris subendrauja su mobiliąja palaikymo stotimi, laikoma, jog jis yra pilnai įsijungęs į sistemą.

2. Mobiliajam kompiuteriui atsidūrus kelių celių persidengimo zonoje gali būti, jog kitos MPS skleidžiamas signalas yra stipresnis nei dabartinės. Tokiu atveju MK nusprendžia inicijuoti perėjimo procesą. Nepaisant to, jog didžiausia tikimybė stipresnį signalą gaut iš naujos MPS tada kai MK yra celių persidengimo zonoje, mobilių palaikymo stočių skleidžiamas signalas yra tikrinamas nuolatos, kas tam tikrą atsitiktinį laiko periodą.



16 pav. *Antroji perėjimo proceso dalis.*

Iš tikrųjų perėjimo nuo vienos palaikymo stoties prie kitos procesas yra labai panašus į patį prisijungimą.

Gavęs stipresnį signalą iš naujosios MPS, mobilusis kompiuteris siunčia prašymą prisijungti prie naujosios palaikymo stoties bei laukia iš jos atsakymo (šiam veiksmui naudojami tie patys įėjimai bei išėjimai kaip ir bandant prisijungti prie pirmosios stoties).

Gavęs patvirtinimą, jog prisijungimas leidžiamas, MK siunčia senajai mobiliajai palaikymo stočiai prašymą atsijungti.

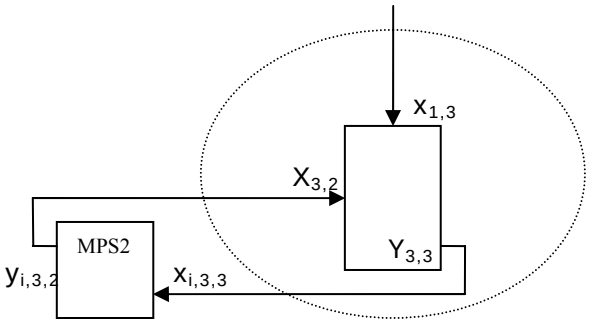
2 lentelė. *Ryšiai tarp MK ir senosios palaikymo stoties.*

Iš agregato	Išėjimas	Į agregatą	Įėjimas
MK1	$y_{3,3}$	MPS1	$x_{1,3,3}$
MPS1	$y_{1,3,2}$	MK1	$x_{3,2}$

Lentelėje *2* pavaizduoti sujungimai, kurie buvo sukurti po to kai MK1 prisijungė prie MPS1, tolimesniam bendravimui. Būtent per šiuos sujungimus ir yra siunčiamas prašymas atsijungti bei leidimas.

Kai MK atsijungia nuo senosios MPS, būna sukuriami būtent tokie patys kanalai su naująja palaikymo stotimi, tolimesniems bendravimams.

3. Paskutinis etapas yra kompiuterio atsijungimas iš sistemos. Tai vyksta tada kaip MK vartotojas nusprendžia išjungti kompiuterį. Šioje sistemoje tariama, jog kompiuterio atsijungimas iš sistemos yra tvarkingas (kompiuteris išjungiamas naudojantis *start* meniu), o ne dėl internetinių tinklų sutrikimo [13].



17 pav. *Paskutinė perėjimo proceso dalis.*

Kompiuterio atsijungimas iš sistemos palaikymo stoties atžvilgiu yra toks pat veiksmas kaip ir atsijungimas nuo MPS jungiantis prie naujosios palaikymo stoties. Prašymas atsijungti siunčiamas tuo pačiu kanalu kai parodyta lentelėje *2*.

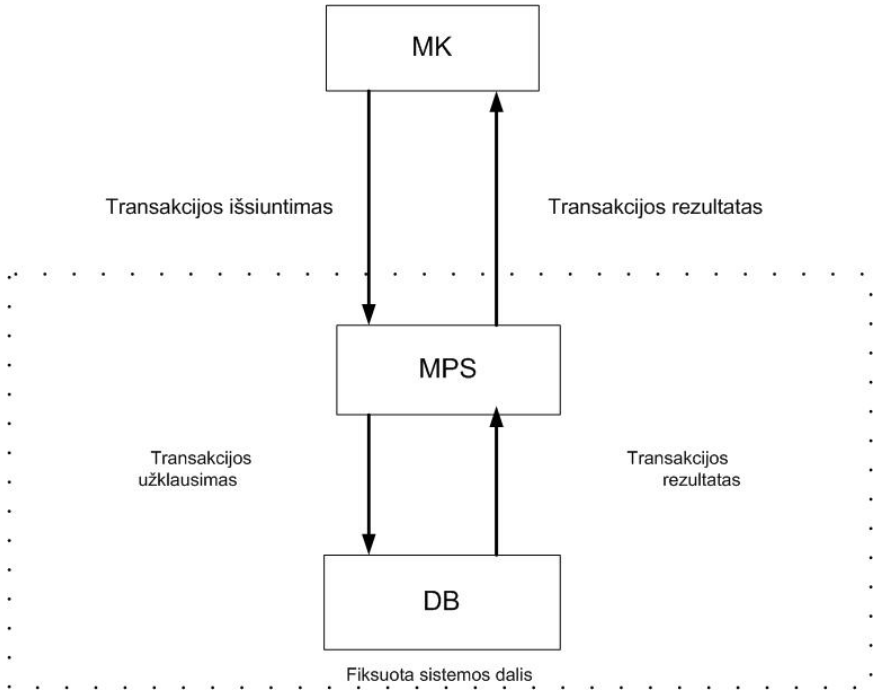
Po to kai pasibaigia MK ir MPS bendravimas, kompiuteris atsijungia iš sistemos, prieš tai sunaikindamas visus likusius įėjimus ir išėjimus. Po atsijungimo MPS taip pat sunaikina įėjimus ir išėjimus, kurie buvo susiję su minimu mobiliuoju kompiuteriu.

3.4 Transakcijų vykdymas

Bevieliu internetiniu ryšiu prie sistemos prisijungę mobilieji vartotojai gali prieiti prie duomenų bazėje esančių duomenų [14]. Duomenų bazė yra fiksuotoje sistemos dalyje, kurią galima pasiekti iš bet kurio mobiliojo ryšio tinklo taško, net ir mobiliajam kompiuteriui judant. Duomenų užklausimas (gali būti skaitymas iš duomenų bazės ar rašymas į ją, arba net abu vienu metu) vadinamas transakcijomis.

Transakcijos – tai procesas, kuris skaito ir analizuoja duomenų bazėje esančią informaciją tam, kad galėtų atsakyti į vartotojo užduotas užklausas.

Transakcijos dažniausiai turi laiko apribojimus [15]. Nagrinėjamoje mobilaus ryšio sistemoje taip pat įvedame laiko apribojimą, t.y. jei per nustatytą laiką duomenų bazė nespėja apdoroti transakcijos ir išsiųsti rezultato, tokios transakcijos vykdymas yra nutraukiamas.



18 pav. *Transakcijos vykdymas.*

Sistemoje nagrinėjama mobilioji transakcija [16]. Tai tokia transakcija, kuri yra sukuriama mobilaus įrenginio ir yra valdoma to paties mobilaus įrenginio. Šiuo atveju palaikymo stotis yra reikalinga tik transakcijos užklausimui, bei rezultatui persiųsti. Ji sistemoje dalyvauja tik kaip tarpininkas.

Kad neiškiltų duomenų konfliktų, t.y., kai duomenų bazėje yra bandoma rašyti ir skaityti tuos pačius duomenis, transakcijos yra apdorojamos paeiliui (kiekviena iš MPS atėjusi užklausa stoja į eilę).

18 paveikslėlyje pavaizduotas transakcijos vykdymas.

Tarp MPS ir DB sujungimai egzistuoja visą laiką (vielinis internetinis ryšys), o tarp MK ir MPS sujungimas sukuriamas tada kai mobilusis kompiuteris prisijungia prie savo palaikymo stoties.

3 lentelė. *Ryšiai tarp MK ir MPS naudojami transakcijų vykdymui.*

Iš agregato	Išėjimas	Į agregatą	Įėjimas
MK1	$y_{3,1}$	MPS1	$x_{1,3,1}$
MPS1	$y_{1,3,1}$	MK1	$x_{3,1}$

3 lentelėje pavaizduoti sujungimai naudojami transakcijos užklausai ir rezultato gavimui.

Transakcijos vykdymas yra labai paprastas, vienintelis atvejis kai reikia papildomų veiksmų yra tada kai transakcija vykdoma mobiliojo kompiuterio perėjimo prie naujos MPS metu. Jei MK pradeda perėjimo procesą dar nesulaukęs išsiųstos transakcijos rezultato, tada persijungdamas kartu su siunčiamu prašymu atsijungti (senajai MPS), MK taip pat nusiunčia savo naujosios palaikymo stoties adresą (numerį). Todėl gavus transakcijos rezultatą senoji MPS persiunčia jį reikiamai mobiliajai palaikymo stočiai (ryšiai tarp palaikymo stočių yra vieliniai), o šioji, savo ruožtu grąžina rezultatą mobiliajam kompiuteriui.

3.5 Siūlomi dynPLA formalizavimo kalbos pakeitimai

Aprašant dinamines sistemas dynPLA formalizavimo kalba pastebėta, jog reikalingi kalbos pataisymai. Aibė A su indeksu t įtraukta į pagrindinį dynPLA specifikacijos aprašą, taip pat išskiriamas punktas matricos M su indeksu t aprašymui. Prieš tai buvo aprašomi veiksmai su šią matricą, tačiau nenurodoma jos pradinė būsena. Šie pakeitimai reikalingi tam, kad sistemos modelį sudarantys agregatai, reikalui esant, galėtų būti skirstomi į smulkesnius agregatus tarp kurių irgi egzistuoja ryšiai.

Kitas siūlomas pakeitimas būtų kanalų įvedimas. Kanalas apima grupę panašias funkcijas atliekančių sujungimų, taip supaprastindamas aprašą.

Lentelėje *2* galima pastebėti, jog mobiliosios palaikymo stoties ir mobiliojo kompiuterio įėjimai bei išėjimai turi skirtingus indeksus. Taip yra todėl, jog prie vienos mobiliosios palaikymo stoties vienu metu gali būti prisijungę daugybė mobiliųjų kompiuterių, tačiau mobilusis kompiuteris gali priklausyti tik vienai MPS.

MPS įėjimų/išėjimų pirmasis indekso skaitmuo reiškia mobiliojo kompiuterio numerį, su kuriuo stotis bendrauja. Jis reikalingas tam, kad galima būtų išskirti reikiamą kompiuterį (jei šio skaitmens nebūtų, visi kompiuteriai bandytų bendrauti su MPS per tą patį įėjimą, kas neleistų palaikymo stočiai normaliai atlikti sekančias funkcijas). Vienintelė išimtis yra $x_{2,1}$ įėjimas. Taip yra todėl, jog šis įėjimas naudojamas tik pradinei žinutei (prašymas prisijungti) gauti iš bet kurio MK, kuris bando įsijungti į sistemą. Vėliau jiems būna sukuriami asmeniniai kanalai bendravimui su sava palaikymo stotimi.

Antrasis mobiliosios palaikymo stoties ir pirmasis mobilaus kompiuterio įėjimų/išėjimų indeksas nurodo koks kanalas yra sukuriamas. Šiuo atveju kanalai tiksliai nurodo kokio tipo bendravimai jais vyksta. Vienetukas reiškia išorinį signalą, dvejetukas – naudojamas MK prisijungimui prie MPS, trejetukas – naudojamas tolimesniems bendravimams. MPS dar

turi ketvertuką, kuris naudojamas bendravimui su duomenų baze ir penketukas – reikalingas transakcijos rezultatui siųsti į kitą MPS (taip atsitinka, jei MK atlieka perėjimą prie kitos MPS kai vyksta transakcijos vykdymas).

Paskutinis skaitmuo reiškia kuris signalas bus siunčiamas nurodytu kanalu, pvz.: MPS įėjimai $x_{i,3,1}$ - nurodo, jog atėjo užklausa transakcijai iš i-tojo MK, $x_{i,3,2}$ - iš i-tojo MK ateina naujos MPS (prie kurios MK ketina jungtis) adresas, $x_{i,3,3}$ - iš i-tojo MK atėjo prašymas atsijungti nuo MPS. Taigi antrasis skaitmuo nurodo jog vyksta bendravimas, tuo tarpu trečiasis – nusako koks tiksliai signalas siunčiamas.

4. Mobilaus ryšio sistemos modelio formalios specifikacijos vykdymo algoritmai

Remiantis dynPLA formalizavimo kalba, aprašyta mobilaus ryšio sistemos modelio specifikacija, buvo sukurti šios sistemos algoritmai (juos aprašant pseudo kalba). Tokie algoritmai palengvina sistemos modelio realizaciją pasirinktoje programavimo kalboje. Tačiau patys algoritmai (aprašyti pseudo kalba) nėra priklausomi nuo konkrečios programavimo kalbos.

4.1 Mobilaus kompiuterio specifikacijos vykdymo algoritmas

when an initialize request (t_0) is received

set allowed incoming signals to $x_{1,1}$

remove all internal events

remove all outgoing signals

$mps(t_0) = 0$

$mps\_naujas(t_0) = 0;$

$vyksta\_trans(t_0) = 0;$

$st_k(t_0) = 0$ for every k

end

when an input ($x_{1,1}$, t_m) is received

add allowed incoming signal $x_{1,3}$

remove signal $x_{1,1}$ from allowed signals

add outgoing signals $y_{1,1}$ and $y_{2,1}$

add internal event e_1'' and set its execution to t_m

end

when an event (e_1'' , t_m) is received

add allowed incoming signal $x_{2,1}$

$w(e_1'', t_{m+1}) = \{t_m + \eta\}$ - set next time event is triggered

if MPS has changed then

send ($y_{2,1}$) to new MPS

end

end

when an input ($x_{2,1}$, t_m) is received

add allowed incoming signals $x_{1,2}$ and $x_{3,2}$

add outgoing signals $y_{3,1}$, $y_{3,2}$ and $y_{3,3}$

if MK wasn't connected to MPS

$mps(t_{m+1}) = mps\_naujas(t_m)$ - start using new MPS

end

if MK was connected to MPS

if transaction is started

send ($y_{3,2}$, $y_{3,3}$) to new MPS
else
send ($y_{3,3}$) to new MPS
end
end
end

when an input ($x_{3,2}$, t_m) is received
$mps(t_{m+1}) = mps\_naujas(t_m)$ - start using new MPS
if MK is shutting down
set allowed incoming signals to $x_{1,1}$
remove all internal events
remove all outgoing signals
$mps(t_{m+1}) = 0$
$mps\_naujas(t_{m+1}) = 0;$
$vyksta\_trans(t_{m+1}) = 0;$
$st_k(t_{m+1}) = 0$ for every k
send ($y_{1,1}$) to the user
end
end

when an input ($x_{1,2}$, t_m) is received
$vyksta\_trans(t_{m+1}) = 1;$
add allowed incoming signal $x_{3,1}$
remove signal $x_{1,2}$ from allowed signals
send ($y_{3,1}$) to the MPS
end

when an input ($x_{3,1}$, t_m) is received
$vyksta\_trans(t_{m+1}) = 0;$
add allowed incoming signal $x_{2,1}$
remove signal $x_{3,1}$ from allowed signals
end

when an input ($x_{1,3}$, t_m) is received
remove all internal events
if MK is not connected
set allowed incoming signals to $x_{1,1}$
remove all outgoing signals
$mps(t_{m+1}) = 0$
$mps\_naujas(t_{m+1}) = 0;$
$vyksta\_trans(t_{m+1}) = 0;$
$st_k(t_{m+1}) = 0$ for every k
send ($y_{1,1}$) to the user
else
set allowed incoming signals to $x_{3,2}$
$mps\_naujas(t_{m+1}) = 0;$

send ($y_{3,1}$) to the MPS
end
end

4.2 Mobilios palaikymo stoties specifikacijos vykdymo algoritmas

when an initialize request (t_0) is received
set allowed incoming signals to $x_{2,1}$
remove all internal events
remove all outgoing signals
remove all discreet state information except $n(t_m)$
$n(t_0) = 0;$
end

when an input ($x_{2,1}$, t_m) is received
$n(t_{m+1}) = n(t_m) + 1;$
add discreet state information for new MK
$mps\_nauja_{n(t_{m+1})}(t_{m+1}) = 0$
$baigti\_trans_{n(t_{m+1})}(t_{m+1}) = 0$
add allowed incoming signal $x_{i,3,1}, x_{i,3,2}, x_{i,3,3}, x_{i,5,1}$
add outgoing signals $y_{i,2,1}, y_{i,3,1}, y_{i,3,2}, y_{i,4,1}$
send ($y_{i,2,1}$) to MK
end

when an event ($e'_{i,3,1}$, t_m) is received
add allowed incoming signal $x_{2,1}$
add allowed incoming signal $x_{i,4,1}$
remove allowed incoming signal $x_{i,5,1}$
send ($y_{i,4,1}$) to DB
end

when an input ($x_{i,3,2}$, t_m) is received
transaction should be marked for destruction if MK is shutting down
$baigti\_trans_i(t_m) = 1$
remove allowed incoming signals $x_{i,3,1}, x_{i,3,2}, x_{i,3,3}$
remove outgoing signals $y_{i,3,1}, y_{i,3,2}$
if MK has sent new MPS address
add outgoing signal $y_{i,5,1}$
end
send ($y_{i,3,2}$) to MK
end

when an input ($x_{i,3,3}$, t_m) is received
remove discreet state information for this MK
remove all allowed incoming signals for this MK
remove all outgoing signals for this MK
send ($y_{i,3,2}$) to MK
end

when an input ($x_{i,4,1}$, t_m) is received
if MK is disconnected
remove discreet state information for this MK

```
                        remove all allowed incoming signals for this MK
                        remove all outgoing signals for this MK
                        if new MPS is known
                                    send ( y_{i,5,1} ) to new MPS
                        end
            else
                        remove allowed incoming signal  x_{i,4,1}
                        send ( y_{i,3,1} ) to new MK
            end
end

when an input ( x_{i,5,1} , t_m ) is received
            if MK is disconnected
                        remove discreet state information for this MK
                        remove all allowed incoming signals for this MK
                        remove all outgoing signals for this MK
                        if new MPS is known
                                    send ( y_{i,5,1} ) to new MPS
                        end
            else
                        remove allowed incoming signal  x_{i,4,1}
                        send ( y_{i,3,1} ) to new MK
            end
end
```

4.3 Duomenų bazės specifikacijos vykdymo algoritmas

```
when an initialize request ( t_0 ) is received
            set allowed incoming signals to  x_{4,1}
            set outgoing signals  y_{4,1}
            internal event never starts
            n(t_0) = 0
end

when an input ( x_{4,1} , t_m ) is received
            n(t_{m+1}) = n(t_m) + 1;
            if first transaction in row
                        calculate when internal event will occur
            end
end

when an event ( e_1'' , t_m ) is received
            n(t_{m+1}) = n(t_m) - 1;
            if last transaction in row
                        internal event never starts
            end
end
```

4.4 Tinklo specifikacijos vykdymo algoritmas

```
when an initialize request ( t_0 ) is received
            mk(t_0) = 0
            mps(t_0) = 2
            two MPS and one DB are added to the system
            non-dynamic connections are created between:
                        MPS and DB
                        MPS and MPS
```

```
                        External and MK
            end
end

when an input ( x_{1,1} , t_m ) is received by MK
            add new MK to the system
            calculate wireless signal strength between all MPS and this MK
end

when an event ( e''_1 , t_m ) is received by MK
            recalculate wireless signal strength between all MPS and this MK
            find MPS with strongest signal
            add communication channel for connection request between MK and MPS with
             strongest signal
end

when an input ( x_{2,1} , t_m ) is received by MK
            add communication channel between MK and new MPS
            remove communication channel for connection request between MK and new
             MPS
            remove communication channel between MK and old MPS
end

when an input ( x_{3,2} , t_m ) is received by MK
            remove communication channel between MK and MPS
end

when an input ( x_{i,3,2} , t_m ) is received at MPS
            set new MPS address to send transaction result
end
```

5. Išvados

Išanalizavus mobiliojo ryšio sistemą ir sukūrus jos formalųjį modelį galima teigti, jog naujai sukurtas PLA formalizavimo kalbos išplėtimas (dynPLA) yra tinkamas aprašinėti dinaminėms sistemoms. Dar daugiau, dynPLA yra ne vien tik tinkama, bet ir reikalinga kalba tokioms užduotims atlikti. Dinaminių sistemų formalizavimas, bandant naudotis tik standartine PLA kalba, būtų labai sudėtingas, jei neįmanomas. Naudojantis dynPLA papildomomis galimybėmis dinaminių sistemų formalizavimas tampa daug paprastesnis ir patogesnis (be tokių savybių, kaip sujungimų matrica *M* su indeksu *t,* ar galimybės keisti įėjimų/išėjimų ar vidinių/išorinių įvykių aibes, sistemos aprašymas taptų labai sudėtingas).

Taip pat aprašius mobiliojo ryšio sistemą dynPLA formalizavimo kalba, dėl patogesnio kalbos naudojimo yra siūlomi keli pataisymai. Tai būtų aibės *A* su indeksu *t* bei matricos *M* su indeksu *t* įtraukimas tarp dynPLA formalizavimo kalbos punktų, leidžiant nustatyt šios aibės bei matricos pradines būsenas. Toks papildymas suteikia galimybę, to prireikus, aprašinėti agregatus net jei jie yra skaidomi į dar smulkesnius agregatus, tarp kurių egzistuoja ryšiai. Taip pat siūloma grupuoti panašias funkcijas atliekančius sujungimus, įvedant papildomus įėjimų/išėjimų indeksus, kurie nurodytu ir kanalą ir tikslų siunčiamą signalą.

Kadangi dynPLA yra labai jauna formalizavimo kalba, tai jai dar nėra sukurtų verifikavimo įrankių. Tokią užduotį siūlyčiau kitiems su šią formalizavimo kalba dirbsiantiems studentams.

6. Literatūra

[1] Pranevičius, H. Kompiuterinių tinklų formalusis specifikavimas ir analizė: agregatinis metodas, Kaunas, 2003.

[2] Packevičius, Š.; Kazla, A.; Pranevičius, H. PLA specifikacijų išplėtimas dinaminių sistemų formalizmui. *Informacinės technologijos 2006*: konferencijos pranešimų medžiaga. Kaunas, 2006, p. 1-5.

[3] Uhrmacher, A. Weltsichten in der ereignisorientierten Simulation: traditionell und systemorientiert: DEVS, *Modellierung und Simulation*: skript der vorlesung, an der Universitat Rostock, WS, 2002.

[4] Zeigler, B.; Praehofer, H.; Tag Gon Kim, Theory of Modeling and Simulation. Academic Press, Auflage, 2000.

[5] Barros, F.J.; Zeigler, B.P. Adapting Queueing: A Case Study Using Dynamic Structure DEVS, *International Trans. In Oper. Res.,* 1997, Nr. 2, p. 87-98.

[6] Uhrmacher, A. Dynamic Structures in Modeling and Simulation: A Reflective Approach, *ACM Transaction on Modeling and Computer Simulation*, 2001, Nr. 2, p. 203-232.

[7] Ulusoy, O. Real-Time Data Management fot Mobile Computing. *Proceedings of the International Workshop on Issues and Applications of Database Technology,* 1998, p 233-240.

[8] Kayan, E.; Ulusoy, O. An Evaluation of Real-Time Transaction Management Issues in Mobile Database Systems, *Departmenr of Computer Engineering and Information Science Bilkent University,* 2000.

[9] Sanh, H. Real-Time Database Systems: Present and Future, *Department of Computer Science University of Virginia,* 1995.

[10] Ramamritham, K. Real-Time Databases, *Journal of Distributed and Parallel Databases,* 1993, Nr. 2, p. 199-226.

[11] Caceres, R.; Padmanabhan, V. N. Fast and Scalable Handoffs for Wireless Internetworks, *Proceedings of ACM MobiCom'96,* 1996.

[12] Seshan, S. Low-Latency Handoffs for Celluler Data Networks, *Technical Report UCB/ CSD, University of California at Berkeley,* 1996, p. 96-899.

[13] Pitoura, E. Data Management for Mobile Computing, *Computer Science Department, University of Ioannina, Summer School,* 1998, p. 1-9.

[14] Kayan, E.; Ulosoy, O. Real-Time Transaction Management in Mobile Computing Systems, *Proceedings of the IEEE Workshop on mobile Computing Systems and Applications,* 1999, p. 127-134.

[15] Abbot, R. K.; Garcia-Molina, H. Sheduling Real-Time Transactions: A Performance Evaluation, *ACM Transactions on Database Systems,* 1992, Nr. 3, p. 513-560.

[16] Chrysanthis, P. K. Transaction Procesising in Mobile Computing Environment, *Proceedings of the IEEE Workshop on Andances in Parallel and Distributed Systems,* 1993, p. 77-82.

7. Priedai

7.1 Mobilaus kompiuterio specifikacija

1. $X(t_m) \subseteq \{x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{3,1}, x_{3,2}\}$

$x_{1,1}$ - atėjo MK įjungimo signalas

$x_{1,2}$ - atėjo signalas inicijuojantis transakciją

$x_{1,3}$ - atėjo MK išjungimo signalas

$x_{2,1}$ - atėjo prisijungimo prie MPS patvirtinimo signalas

$x_{3,1}$ - atėjo transakcijos rezultatas

$x_{3,2}$ - atėjo atsijungimo nuo MPS patvirtinimo signalas

2. $Y(t_m) \subseteq \{y_{1,1}, y_{2,1}, y_{3,1}, y_{3,2}, y_{3,3}\}$

$y_{1,1}$ - siunčiamas MK išjungimo signalas

$y_{2,1}$ - siunčiamas prašymas prisijungti prie MPS

$y_{3,1}$ - siunčiama transakcija į MPS

$y_{3,2}$ - siunčiamas prašymas atsijungti ir naujos MPS adresas, kad žinotu kur persiųsti transakcijos rezultatą, į senąją MPS

$y_{3,3}$ - siunčiamas prašymas atsijungti nuo MPS

3. $E'(t_m) \subseteq \{e'_{1,1}, e'_{1,2}, e'_{1,3}, e'_{2,1}, e'_{3,1}, e'_{3,2}\}$

$e'_{1,1}$ - atėjo MK įjungimo signalas

$e'_{1,2}$ - atėjo signalas inicijuojantis transakciją

$e'_{1,3}$ - atėjo MK išjungimo signalas

$e'_{2,1}$ - atėjo prisijungimo prie MPS patvirtinimo signalas

$e'_{3,1}$ - atėjo transakcijos rezultatas

$e'_{3,2}$ - atėjo atsijungimo nuo MPS patvirtinimo signalas

4. $E''(t_m) \subseteq \{e''_1\}$

e''_1 - tikrinami MPS signalu stiprumai

5. $\{e''_1\} \rightarrow \{\eta\}$

6. $z_\nu(t_m) \subseteq \{w(e''_1, t_m)\}$

$w(e''_1, t_m)$ - kito tikrinimo ar nepasikeitė MPS momentas

7. $\nu(t_m) = \{mps(t_m), mps\_nauja(t_m), vyksta\_trans(t_m), st_k(t_m)\}$

$mps(t_m)$ - MPS, prie kurios dabar prisijungęs MK, numeris (0 – neprisijungęs prie jokios MPS)

$mps\_naujas(t_m)$ - MPS, prie kurios planuoja jungsis MK, numeris

$vyksta\_trans(t_m)$ - saugo informacija apie tai ar yra nebaigtų vykdyti transakcijų (0 – nėra, 1 – yra)

$st_k(t_m)$ - signalo stiprumas tarp MK ir k-tojo MPS

8. $A(t_m) = \varnothing$

9. $M(t_m) = \varnothing$

10. $t_0 = 0;$

$X(t_0) = \{x_{1,1}\};$

$Y(t_0) = \varnothing;$

$E'(t_0) = \{e'_{1,1}\};$

$E''(t_0) = \varnothing;$

$z_\nu(t_0) = \varnothing;$

$\nu(t_0) = \{mps(t_0), mps\_nauja(t_0), vyksta\_trans(t_0), st_k(t_0)\}$

$mps(t_0) = 0$

$mps\_naujas(t_0) = 0;$

$vyksta\_trans(t_0) = 0;$

$st_k(t_0) = 0, \quad k = \overline{1..\infty};$ - pradžioje signalu nėra nes kompiuteriukas išjungtas

11. $\hat{H}(e'_{1,1})$: – **atėjo MK įjungimo signalas**

$X(t_{m+1}) = X(t_m) \cup \{x_{1,3}\}, \quad jei \quad \{x_{1,3}\} \notin X(t_m);$ - galima MK išjungti

$X(t_{m+1}) = X(t_m) - \{x_{1,1}\}, \quad jei \quad \{x_{1,1}\} \in X(t_m);$ - negalima įjungti MK

$Y(t_{m+1}) = Y(t_m) \cup \{y_{2,1}\}, \quad jei \quad \{y_{2,1}\} \notin Y(t_m);$ - galima siųsti prašymą prisijungti prie MPS

$Y(t_{m+1}) = Y(t_m) \cup \{y_{1,1}\}, \quad jei \quad \{y_{1,1}\} \notin Y(t_m);$ - galima siųsti MK išjungimo signalą

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{1,3}\}, \quad jei \quad \{e'_{1,3}\} \notin E'(t_m);$ - galima MK išjungti

$E'(t_{m+1}) = E'(t_m) - \{e'_{1,1}\}, \quad jei \quad \{e'_{1,1}\} \in E'(t_m);$ - negalima įjungti MK

$E''(t_{m+1}) = E''(t_m) \cup \{e''_1\}, \quad jei \quad \{e''_1\} \notin E''(t_m)$

$z_\nu(t_{m+1}) = z_\nu(t_m) \cup \{w(e''_1, t_m)\}, \quad jei \quad \{w(e''_1, t_m)\} \notin z_\nu(t_m)$

$w(e''_1, t_{m+1}) = \{t_m\}$

$\hat{H}(e''_1)$: **- tikrinami MPS signalu stiprumai**

$X(t_{m+1}) = X(t_m) \cup \{x_{2,1}\}, \quad jei \quad ps(t_{m+1}) \neq mps\_naujas(t_{m+1}) \quad ir \quad \{x_{2,1}\} \notin X(t_m);$ - gali ateiti prisijungimo prie MPS patvirtinimo signalas

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{2,1}\}, \quad jei \quad ps(t_{m+1}) \neq mps\_naujas(t_{m+1}) \quad ir \quad \{e'_{2,1}\} \notin E'(t_m);$ - gali ateiti prisijungimo prie MPS patvirtinimo signalas

$w(e''_1, t_{m+1}) = \{t_m + \eta\}$

$G(e''_1)$:

$Y = \{y_{2,1}\}, \quad jei \quad ps(t_{m+1}) \neq mps\_naujas(t_{m+1});$

$\hat{H}(e'_{2,1})$: **- atėjo prisijungimo prie MPS patvirtinimo signalas**

$X(t_{m+1}) = X(t_m) \cup \{x_{1,2}\}, \quad jei \quad \{x_{1,2}\} \notin X(t_m);$ - gali ateiti signalas inicijuojantis transakciją

$X(t_{m+1}) = X(t_m) \cup \{x_{3,2}\}, \quad jei \quad \{x_{3,2}\} \notin X(t_m);$ - gali ateiti atsijungimo nuo MPS patvirtinimas

$Y(t_{m+1}) = Y(t_m) \cup \{y_{3,1}\}, \quad jei \quad \{y_{3,1}\} \notin Y(t_m);$ - galima siųsti transakciją į MPS

$Y(t_{m+1}) = Y(t_m) \cup \{y_{3,2}\}, \quad jei \quad \{y_{3,2}\} \notin Y(t_m);$ - galima siųsti senojo MPS adresą

$Y(t_{m+1}) = Y(t_m) \cup \{y_{3,3}\}, \quad jei \quad \{y_{3,3}\} \notin Y(t_m);$ - galima siųsti prašymą atsijungti nuo MPS

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{1,2}\}, \quad jei \quad \{e'_{1,2}\} \notin E'(t_m);$ - gali ateiti signalas inicijuojantis transakciją

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{3,2}\}, \quad jei \quad \{e'_{3,2}\} \notin E'(t_m);$ - gali ateiti atsijungimo nuo MPS patvirtinimas

$mps(t_{m+1}) = mps\_naujas(t_m), \quad jei \quad mps(t_m) = 0$ - jei prieš tai buvo neprisijungęs, tai toliau bendrauja su naująją MPS

$G(e'_{2,1})$:

$$Y = \begin{cases} \{y_{3,2}\}, & jei \quad vyksta\_trans(t_m) = 1 \quad ir \quad mps(t_m) \neq 0; \\ \{y_{3,3}\}, & jei \quad vyksta\_trans(t_m) = 0 \quad ir \quad mps(t_m) \neq 0; \end{cases}$$ - jei vyko transakcija, tai reikia nusiusti naujos MPS adresą ir prašymą atsijungti, arba jei transakcijos nebuvo, tik prašymą atsijungti

$\hat{H}(e'_{3,2})$: **- atėjo atsijungimo nuo MPS patvirtinimo signalas (persijungimo metu)**

$mps(t_{m+1}) = mps\_naujas(t_m);$ - persijungiam prie naujo MPS

$X(t_{m+1}) = \{x_{1,1}\}, jei \quad mps(t_{m+1}) = 0$ - jei MK išjunginėjamas

$Y(t_{m+1}) = \{\varnothing\}, jei \quad mps(t_{m+1}) = 0$

$E'(t_{m+1}) = \{e'_{1,1}\}, jei \quad mps(t_{m+1}) = 0$

$E''(t_{m+1}) = \{\varnothing\}, jei \quad mps(t_{m+1}) = 0;$

$z_v(t_{m+1}) = \{\varnothing\}, \; jei \quad mps(t_{m+1}) = 0;$

$vyksta\_trans(t_{m+1}) = 0, jei \quad mps(t_{m+1}) = 0;$

$G(e'_{3,2})$:

$Y = \{y_{1,1}\}, jei \quad mps(t_{m+1}) = 0$ - kompiuteris išjungiamas, jei atsijungiama nuo visu MPS

$H(e'_{1,2})$: **- atėjo signalas inicijuojantis transakciją**

$vyksta\_trans(t_{m+1}) = 1;$

$X(t_{m+1}) = X(t_m) \cup \{x_{3,1}\}, \quad jei \quad \{x_{3,1}\} \notin X(t_m);$ - gali ateiti transakcijos rezultatas

$X(t_{m+1}) = X(t_m) - \{x_{1,2}\}, \quad jei \quad \{x_{1,2}\} \in X(t_m);$ - negalima inicijuoti naujos transakcijos, kol neapdorota esama

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{3,1}\}, \quad jei \quad \{e'_{3,1}\} \notin E'(t_m);$ - gali ateiti transakcijos rezultatas

$E'(t_{m+1}) = E'(t_m) - \{e'_{1,2}\}, \quad jei \quad \{e'_{1,2}\} \in E'(t_m);$ - negalima inicijuoti naujos transakcijos, kol neapdorota esama

$E''(t_{m+1}) = E''(t_m);$

$G(e'_{1,2})$:

$Y = \{y_{3,1}\}$ - siunčiama transakcija į MPS

$H(e'_{3,1})$: **- atėjo transakcijos rezultatas**

$vyksta\_trans(t_{m+1}) = 0;$

$X(t_{m+1}) = X(t_m) \cup \{x_{1,2}\}, \quad jei \quad \{x_{1,2}\} \notin X(t_m);$ - gali ateiti signalas inicijuojantis transakciją

$X(t_{m+1}) = X(t_m) - \{x_{3,1}\}, \quad jei \quad \{x_{3,1}\} \in X(t_m);$ - negali ateiti transakcijos rezultatas

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{1,2}\}, \quad jei \quad \{e'_{1,2}\} \notin E'(t_m);$ - gali ateiti signalas inicijuojantis transakciją

$E'(t_{m+1}) = E'(t_m) - \{e'_{3,1}\}, \quad jei \quad \{e'_{3,1}\} \in E'(t_m);$ - negali ateiti transakcijos rezultatas

$G(e'_{3,1})$:

$Y = \{\varnothing\}$

$H(e'_{1,3})$: **- atėjo MK išjungimo signalas**

$$X(t_{m+1}) = \begin{cases} \{x_{1,1}\}, jei \quad mps(t_m) = 0 \\ \{x_{3,2}\}, jei \quad mps(t_m) \neq 0 \end{cases}$$

$$Y(t_{m+1}) = \begin{cases} \{\varnothing\}, jei \quad mps(t_m) = 0 \\ Y(t_m), jei \quad mps(t_m) \neq 0 \end{cases}$$

$$E'(t_{m+1}) = \begin{cases} \{e'_{1,1}\}, jei \quad mps(t_m) = 0 \\ \{e'_{3,2}\}, jei \quad mps(t_m) \neq 0 \end{cases}$$

$E''(t_{m+1}) = \{\varnothing\};$

$z_v(t_{m+1}) = \{\varnothing\};$

$mps\_naujas(t_{m+1}) = 0;$

$G(e'_{1,3})$:

$$Y = \begin{cases} \{y_{1,1}\}, jei \quad mps(t_m) = 0 \\ \{y_{3,2}\}, jei \quad mps(t_m) \neq 0 \quad ir \quad vyksta\_trans(t_m) = 1 \\ \{y_{3,3}\}, jei \quad mps(t_m) \neq 0 \quad ir \quad vyksta\_trans(t_m) = 0 \end{cases}$$

7.2 Mobilios palaikymo stoties specifikacija

1. $X(t_m) \subseteq \{x_{2,1}, x_{i,3,1}, x_{i,3,2}, x_{i,3,3}, x_{i,4,1}, x_{i,5,1}\}$

$x_{2,1}$ - atėjo prašymas prisijungti prie MPS

$x_{i,3,1}$ - iš i-tojo MK atėjo transakcija ($i = \overline{1..\infty}$)

$x_{i,3,2}$ - iš i-tojo MK atėjo naujos MPS adresas ($i = \overline{1..\infty}$)

$x_{i,3,3}$ - iš i-tojo MK atėjo prašymas atsijungti nuo MPS ($i = \overline{1..\infty}$)

$x_{i,4,1}$ - iš DB atėjo transakcijos rezultatas i-tajam MK ($i = \overline{1..\infty}$)

$x_{i,5,1}$ - iš MPS atėjo transakcijos rezultatas i-tajam MK ($i = \overline{1..\infty}$)

2. $Y(t_m) \subseteq \{y_{i,3,1}, y_{i,3,2}, y_{i,4,1}, y_{i,5,1}\}$

$y_{i,2,1}$ - i-tajam MK siunčiamas prisijungimo patvirtinimas ($i = \overline{1..\infty}$)

$y_{i,3,1}$ - i-tajam MK siunčiamas transakcijos rezultatas ($i = \overline{1..\infty}$)

$y_{i,3,2}$ - i-tajam MK siunčiamas atsijungimo patvirtinimas ($i = \overline{1..\infty}$)

$y_{i,4,1}$ - į DB siunčiama i-tojo MK transakcija ($i = \overline{1..\infty}$)

$y_{i,5,1}$ - į nauja MPS siunčiamas i-tojo MK transakcijos rezultatas ($i = \overline{1..\infty}$)

3. $E'(t_m) \subseteq \{e'_{2,1}, e'_{i,3,1}, e'_{i,3,2}, e'_{i,3,3}, e'_{i,4,1}, e'_{i,5,1}\}$

$e'_{2,1}$ - atėjo prašymas prisijungti prie MPS

$e'_{i,3,1}$ - iš i-tojo MK atėjo transakcija ($i = \overline{1..\infty}$)

$e'_{i,3,2}$ - iš i-tojo MK atėjo naujos MPS adresas ir prašymas atsijungti($i = \overline{1..\infty}$)

$e'_{i,3,3}$ - iš i-tojo MK atėjo prašymas atsijungti nuo MPS ($i = \overline{1..\infty}$)

$e'_{i,4,1}$ - iš DB atėjo transakcijos rezultatas i-tajam MK ($i = \overline{1..\infty}$)

$e'_{i,5,1}$ - iš MPS atėjo transakcijos rezultatas i-tajam MK ($i = \overline{1..\infty}$)

4. $E''(t_m) = \varnothing$

5. $\varnothing$

6. $z_v(t_m) = \varnothing$

7. $v(t_m) \subseteq \{n(t_m), mps\_nauja_i(t_m), baigti\_trans_i(t_m)\}$

$n(t_m)$ - MK indeksas

$mps\_nauja_i(t_m)$ - saugoma MPS prie kurio persijungė i-tasis MK (transakcijos metu) adresas (0 – MK nepersijungė)

$baigti\_trans_i(t_m)$ - ar persiųsti, ar naikinti transakcijos rezutatą (0 – persiųsti, 1 – naikinti)

8. $A(t_m) = \varnothing$

9. $M(t_m) = \varnothing$

10. $t_0 = 0;$

$X(t_0) = \{x_{2,1}\};$

$Y(t_0) = \varnothing;$

$E'(t_0) = \{e'_{2,1}\};$

$E''(t_0) = \varnothing;$

$z_\nu(t_0) = \varnothing;$

$\nu(t_0) \subseteq \{n(t_0)\}$

$n(t_0) = 0;$

11. $H(e'_{2,1})$: – **atėjo prašymas prisijungti prie MPS**

$n(t_{m+1}) = n(t_m) + 1;$ - padidinamas indeksas

$mps\_nauja_{n(t_{m+1})}(t_{m+1}) = 0$ - transakcijos rezultatą siųsti MK

$\nu(t_{m+1}) = \nu(t_m) \cup mps\_nauja_{n(t_{n+1})}, \quad jei \quad mps\_nauja_{n(t_{n+1})} \notin \nu(t_m);$

$baigti\_trans_{n(t_{m+1})}(t_{m+1}) = 0$ - transakcijos rezultatą persiųsti

$\nu(t_{m+1}) = \nu(t_m) \cup baigti\_trans_{n(t_{n+1})}, \quad jei \quad baigti\_trans_{n(t_{n+1})} \notin \nu(t_m);$

$X(t_{m+1}) = X(t_m) \cup \{x_{n(t_{m+1}),3,1}\}, \quad jei \quad \{x_{n(t_{m+1}),3,1}\} \notin X(t_m);$ - iš MK gali ateiti transakcija

$X(t_{m+1}) = X(t_m) \cup \{x_{n(t_{m+1}),3,2}\}, \quad jei \quad \{x_{n(t_{m+1}),3,2}\} \notin X(t_m);$ - iš MK gali ateiti naujo MPS adresas

$X(t_{m+1}) = X(t_m) \cup \{x_{n(t_{m+1}),3,3}\}, \quad jei \quad \{x_{n(t_{m+1}),3,3}\} \notin X(t_m);$ - iš MK gali ateiti prašymas atsijungti nuo MPS

$X(t_{m+1}) = X(t_m) \cup \{x_{n(t_{m+1}),5,1}\}, \quad jei \quad \{x_{n(t_{m+1}),5,1}\} \notin X(t_m);$ - iš MPS gali ateiti transakcijos rezultatas MK

$Y(t_{m+1}) = Y(t_{\mathrm{m}}) \cup \{y_{n(t_{m+1}),2,1}\}, \quad jei \quad \{y_{n(t_{m+1}),2,1}\} \notin Y(t_m);$ - galima siųsti prisijungimo patvirtinimą į MK

$Y(t_{m+1}) = Y(t_{\mathrm{m}}) \cup \{y_{n(t_{m+1}),3,1}\}, \quad jei \quad \{y_{n(t_{m+1}),3,1}\} \notin Y(t_m);$ - galima siųsti transakcijos rezultatą į MK

$Y(t_{m+1}) = Y(t_{\mathrm{m}}) \cup \{y_{n(t_{m+1}),3,2}\}, \quad jei \quad \{y_{n(t_{m+1}),3,2}\} \notin Y(t_m);$ - galima siųsti atsijungimo patvirtinimą į MK

$Y(t_{m+1}) = Y(t_m) \cup \{y_{n(t_{m+1}),4,1}\}, \quad jei \quad \{y_{n(t_{m+1}),4,1}\} \notin Y(t_m);$ - galima siųsti transakciją į DB

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{n(t_{m+1}),3,1}\}, \quad jei \quad \{e'_{n(t_{m+1}),3,1}\} \notin E'(t_m);$ - iš MK gali ateiti transakcija

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{n(t_{m+1}),3,2}, e'_{n(t_{m+1}),3,3}\}, \quad jei \quad \{e'_{n(t_{m+1}),3,2}, e'_{n(t_{m+1}),3,3}\} \notin E'(t_m);$ - iš MK gali ateiti prašymas atsijungti nuo MPS su naujo MPS adresu

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{n(t_{m+1}),3,3}\}, \quad jei \quad \{e'_{n(t_{m+1}),3,3}\} \notin E'(t_m);$ - iš MK gali ateiti prašymas atsijungti nuo MPS

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{n(t_{m+1}),5,1}\}, \quad jei \quad \{e'_{n(t_{m+1}),5,1}\} \notin E'(t_m);$ - iš MPS gali ateiti transakcijos rezultatas MK

$G(e'_{2,1})$:

$Y = \{y_{n(t_{m+1}),2,1}\}$ - MK siunčiamas prisijungimo patvirtinimas

$H(e'_{i,3,1})$: **- iš i-tojo MK atėjo transakcija (** $i = \overline{1..\infty}$ **)**

$X(t_{m+1}) = X(t_m) \cup \{x_{i,4,1}\}, \quad jei \quad \{x_{i,4,1}\} \notin X(t_m);$ - gali ateiti transakcijos rezultatas iš DB

$X(t_{m+1}) = X(t_m) - \{x_{i,5,1}\}, \quad jei \quad \{x_{i,5,1}\} \in X(t_m);$ - jei atėjo nauja transakcijos užklausa iš MK, vadinas nėra ankstesnės transakcijos, kurios rezultatas galėtų ateit iš kito MPS.

$E'(t_{m+1}) = E'(t_m) \cup \{e'_{i,4,1}\}, \quad jei \quad \{e'_{i,4,1}\} \notin E'(t_m);$ - gali ateiti transakcijos rezultatas iš DB

$E'(t_{m+1}) = E'(t_m) - \{e'_{i,5,1}\}, \quad jei \quad \{e'_{i,5,1}\} \in E'(t_m);$

$G(e'_{i,3,1})$:

$Y = \{y_{i,4,1}\}$ - į DB siunčiama i-tojo MK transakcija

$\hat{H}(e'_{i,3,2})$: **- iš i-tojo MK atėjo naujos MPS adresas ir prašymas atsijungti (** $i = \overline{1..\infty}$ **)**

$baigti\_trans_i(t_m) = 1, \quad jei \quad mps\_nauja_i(t_{m+1}) = 0;$ - transakcijos rezultatą naikinti

$X(t_{m+1}) = X(t_m) - \{x_{i,3,1}\}, \quad jei \quad \{x_{i,3,1}\} \in X(t_m);$ - iš MK negali ateiti transakcija

$X(t_{m+1}) = X(t_m) - \{x_{i,3,2}\}, \quad jei \quad \{x_{i,3,2}\} \in X(t_m);$ - iš MK negali ateiti naujo MPS adresas

$X(t_{m+1}) = X(t_m) - \{x_{i,3,3}\}, \quad jei \quad \{x_{i,3,3}\} \in X(t_m);$ - iš MK negali ateiti prašymas atsijungti nuo MPS

$Y(t_{m+1}) = Y(t_m) - \{y_{i,3,1}\}, \quad jei \quad \{y_{i,3,1}\} \in Y(t_m);$ - negalima siųsti transakcijos rezultato į MK

$Y(t_{m+1}) = Y(t_m) - \{y_{i,3,2}\}, \quad jei \quad \{y_{i,3,2}\} \in Y(t_m);$ - negalima siųsti atsijungimo patvirtinimo į MK

$Y(t_{m+1}) = Y(t_m) \cup \{y_{i,5,1}\}, \quad jei \quad \{y_{i,5,1}\} \notin Y(t_m) \quad ir \quad mps\_nauja_i(t_{m+1}) \neq 0;$ - į nauja MPS galima siųsti transakcijos rezultatą, jei kompiuteriukas persijungia, o ne išsijungia

$E'(t_{m+1}) = E'(t_m) - \{x'_{i,3,1}\}, \quad jei \quad \{e'_{i,3,1}\} \in E'(t_m);$ - iš MK negali ateiti transakcija

$E'(t_{m+1}) = E'(t_m) - \{e'_{i,3,2}\}, \quad jei \quad \{e'_{i,3,2}\} \in E'(t_m);$ - iš MK negali ateiti naujo MPS adresas

$E'(t_{m+1}) = E'(t_m) - \{e'_{i,3,3}\}, \quad jei \quad \{e'_{i,3,3}\} \in E'(t_m);$ - iš MK negali ateiti prašymas atsijungti nuo MPS

$G(e'_{i,3,2})$:

$Y = \{y_{i,3,2}\}$ - siunčiamas atsijungimo patvirtinimas į MK

$H(e'_{i,3,3})$: **- iš i-tojo MK atėjo prašymas atsijungti (** $i = \overline{1..\infty}$ **)**

$\nu(t_{m+1}) = \nu(t_m) - mps\_nauja_i, \quad jei \quad mps\_nauja_i \in \nu(t_m);$

$\nu(t_{m+1}) = \nu(t_m) - baigti\_trans_i, \quad jei \quad baigti\_trans_i \in \nu(t_m);$

$X(t_{m+1}) = X(t_m) - \{x_{i,*,*}\}, \quad jei \quad \{x_{i,*,*}\} \in X(t_m);$ - negali ateiti joks pranešimas susijęs su šiuo MK

$Y(t_{m+1}) = Y(t_m) - \{y_{i,*,*}\}, \quad jei \quad \{y_{i,*,*}\} \in Y(t_m);$ - negalima siųsti jokio pranešimo susijusio su šiuo MK

$E'(t_{m+1}) = E'(t_m) - \{x'_{i,*,*}\}, \quad jei \quad \{e'_{i,*,*}\} \in E'(t_m);$ - negali ateiti joks pranešimas susijęs su šiuo MK

$G(e'_{i,3,3})$:

$Y = \{y_{i,3,2}\}$ - siunčiamas atsijungimo patvirtinimas į MK

$H(e'_{i,4,1})$: **- iš DB atėjo transakcijos rezultatas i-tajam MK (** $i = \overline{1..\infty}$ **)**

$mps\_nauja_i(t_{m+1}) = 0, \quad jei \quad mps\_nauja_i(t_m) \neq 0$

$baigti\_trans_i(t_{m+1}) = 0, \quad jei \quad baigti\_trans_i(t_m) \neq 0$

$\nu(t_{m+1}) = \nu(t_m) - mps\_nauja_i, \quad jei \quad (mps\_nauja_i(t_m) \neq 0 \quad arba \quad baigti\_trans_i(t_m) = 1) \quad ir \quad mps\_nauja_i \in \nu(t_m);$

$$\nu(t_{m+1}) = \nu(t_m) - baigti\_trans_i, \quad jei \quad (mps\_nauja_i(t_m) \neq 0 \quad arba \quad baigti\_trans_i(t_m) = 1) \quad ir \quad baigti\_trans_i \in \nu(t_m);$$

$$X(t_{m+1}) = \begin{cases} X(t_m) - \{x_{i,*,*}\}, & jei \quad (mps\_nauja_i(t_m) \neq 0 \quad arba \quad baigti\_trans_i(t_m) = 1) \quad ir \\ & \{x_{i,*,*}\} \in X(t_m) \\ X(t_m) - \{x_{i,4,1}\}, & jei \quad mps\_nauja_i(t_m) = 0 \quad ir \quad \{x_{i,4,1}\} \in X(t_m) \end{cases}; \text{ - jei}$$

MK neatsijungęs, tai negali ateiti transakcijos rezultatas iš DB, jei atsijungęs negali ateiti joks pranešimas susijęs su šiuo MK

$$Y(t_{m+1}) = Y(t_m) - \{y_{i,*,*}\}, \quad jei \quad (mps\_nauja_i(t_m) \neq 0 \quad arba \quad baigti\_trans_i(t_m) = 1) \quad ir \quad \{y_{i,*,*}\} \in Y(t_m);$$ -

negalima siųsti jokio pranešimo susijusio su šiuo MK

$$E'(t_{m+1}) = \begin{cases} E'(t_m) - \{e'_{i,*,*}\}, & jei \quad (mps\_nauja_i(t_m) \neq 0 \quad arba \quad baigti\_trans_i(t_m) = 1) \quad ir \\ & \{e'_{i,*,*}\} \in E'(t_m) \\ E'(t_m) - \{e'_{i,4,1}\}, & jei \quad mps\_nauja_i(t_m) = 0 \quad ir \quad \{e'_{i,4,1}\} \in E'(t_m) \end{cases}; \text{ - jei}$$

MK neatsijungęs, tai negali ateiti transakcijos rezultatas iš DB, jei atsijungęs negali ateiti joks pranešimas susijęs su šiuo MK

$G(e'_{i,4,1})$:

$$Y = \begin{cases} \{y_{i,5,1}\}, & jei \quad mps\_nauja_i(t_m) \neq 0 \\ \{y_{i,3,1}\}, & jei \quad mps\_nauja_i(t_m) = 0 \quad ir \quad baigti\_trans_i(t_m) = 0 \\ \varnothing, & jei \quad mps\_nauja_i(t_m) = 0 \quad ir \quad baigti\_trans_i(t_m) = 1 \end{cases} \text{ - jei MK}$$

neatsijungęs, tai jam siunčiamas transakcijos rezultatas, jei persijungęs transakcijos rezultatas siunčiamas naujam MPS, jei išjungtas transakcijos rezultatas naikinamas

$H(e'_{i,5,1})$: **- iš MPS atėjo transakcijos rezultatas i-tajam MK (** $i = \overline{1..\infty}$ **)**

$$mps\_nauja_i(t_{m+1}) = 0, \quad jei \quad mps\_nauja_i(t_m) \neq 0$$

$$baigti\_trans_i(t_{m+1}) = 0, \quad jei \quad baigti\_trans_i(t_m) \neq 0$$

$$\nu(t_{m+1}) = \nu(t_m) - mps\_nauja_i, \quad jei \quad mps\_nauja_i(t_m) \neq 0 \quad ir \quad mps\_nauja_i \in \nu(t_m);$$

$$\nu(t_{m+1}) = \nu(t_m) - baigti\_trans_i, \quad jei \quad mps\_nauja_i(t_m) \neq 0 \quad ir \quad baigti\_trans_i \in \nu(t_m);$$

$$X(t_{m+1}) = \begin{cases} X(t_m) - \{x_{i,*,*}\}, & jei \quad mps\_nauja_i(t_m) \neq 0 \quad ir \quad \{x_{i,*,*}\} \in X(t_m) \\ X(t_m) - \{x_{i,5,1}\}, & jei \quad mps\_nauja_i(t_m) = 0 \quad ir \quad \{x_{i,5,1}\} \in X(t_m) \end{cases}; \text{ - jei MK}$$

neatsijungęs, tai negali ateiti transakcijos rezultatas iš MPS, jei atsijungęs negali ateiti joks pranešimas susijęs su šiuo MK

$$Y(t_{m+1}) = Y(t_m) - \{y_{i,*,*}\}, \quad jei \quad \{y_{i,*,*}\} \in Y(t_m) \quad ir \quad baigti\_trans_i(t_m) = 1; \text{ - negalima}$$

siųsti jokio pranešimo susijusio su šiuo MK

$$E'(t_{m+1}) = \begin{cases} E'(t_m) - \{e'_{i,*,*}\}, & jei \quad mps\_nauja_i(t_m) \neq 0 \quad ir \quad \{e'_{i,*,*}\} \in E'(t_m) \\ E'(t_m) - \{e'_{i,5,1}\}, & jei \quad mps\_nauja_i(t_m) = 0 \quad ir \quad \{e'_{i,5,1}\} \in E'(t_m) \end{cases}; \text{ - jei MK}$$

neatsijungęs, tai negali ateiti transakcijos rezultatas iš MPS, jei atsijungęs negali ateiti joks pranešimas susijęs su šiuo MK

$G(e'_{i,5,1})$:

$$Y = \begin{cases} \{y_{i,5,1}\}, & jei \quad mps\_nauja_i(t_m) \neq 0 \\ \{y_{i,3,1}\}, & jei \quad mps\_nauja_i(t_m) = 0 \quad ir \quad baigti\_trans_i(t_m) = 0 \\ \varnothing, & jei \quad mps\_nauja_i(t_m) = 0 \quad ir \quad baigti\_trans_i(t_m) = 1 \end{cases} \text{ - jei MK}$$

neatsijungęs, tai jam siunčiamas transakcijos rezultatas, jei persijungęs transakcijos rezultatas siunčiamas naujam MPS, jei išjungtas transakcijos rezultatas naikinamas

7.3 Duomenų bazės specifikacija

1. $X(t_m) \subseteq \{x_{4,1}\}$

   $x_{4,1}$ - atėjo nauja transakcija

2. $Y(t_m) \subseteq \{y_{4,1}\}$

   $y_{4,1}$ - siunčiamas transakcijos rezultatas

3. $E'(t_m) \subseteq \{e'_{4,1}\}$

   $e'_{4,1}$ - atėjo signalas $x_{4,1}$

4. $E''(t_m) \subseteq \{e''_1\}$

   e''_1 - baigėsi transakcijos apdorojimas

5. $\{e''_1\} \to \{\eta_j\}$

   $\{\eta_j\}$ - j-tosios transakcijos aptarnavimo trukmė

6. $z_\nu(t_m) \subseteq \{w(e''_1, t_m)\}$

   $w(e''_1, t_m)$ - transakcijos apdorojimo pabaigos momentas

7. $\nu(t_m) = \{n(t_m)\}$

   $n(t_m)$ - transakcijų kiekis eilėje

8. $A(t_m) = \varnothing$

9. $M(t_m) = \varnothing$

10. $t_0 = 0;$

    $X(t_0) = \{x_{1,4}\};$

    $Y(t_0) = \{y_{1,4}\};$

    $E'(t_0) = \{e'_{4,1}\};$

    $E''(t_0) = \{e''_1\};$

    $z_\nu(t_0) = \{w(e''_1, t_0)\};$

    $w(e''_1, t_0) = \infty$

    $\nu(t_0) = \{n(t_0)\}$

    $n(t_0) = 0$ - pradžioje eilė tuščia

11. $H(e'_{4,1})$: **– atėjo nauja transakcija**

    $$n(t_{m+1}) = n(t_m) + 1;$$

    $$w(e''_1, t_{m+1}) = \begin{cases} t_m + \eta_j, & jei \quad n(t_m) = 0 \\ w(e''_1, t_m), & jei \quad n(t_m) > 0 \end{cases}$$

    $H(e''_1)$: **- baigėsi transakcijos apdorojimas**

    $$n(t_{m+1}) = n(t_m) - 1;$$

    $$w(e''_1, t_{m+1}) = \begin{cases} t_m + \eta_j, & jei \quad n(t_{m+1}) > 0 \\ \infty, & jei \quad n(t_{m+1}) = 0 \end{cases}$$

    $G(e''_1)$:

    $$Y = \{y_{4,1}\}$$

7.4 Tinklo specifikacija

1. $X(t_m) = \varnothing$
2. $Y(t_m) = \varnothing$
3. $E'(t_m) = \varnothing$
4. $E''(t_m) = \varnothing$

5. $\varnothing$

6. $z_\nu(t_m) = \varnothing$

7. $\nu(t_m) \subseteq \{mk(t_m), mps(t_m)\}$

$mk(t_m)$ - mobiliųjų kompiuterių indeksas

$mps(t_m)$ - mobiliųjų palaikymo stočių indeksas

8. $A(t_m) = \{DB, MPS_j, MK_i\}$

DB - duomenų bazė

MPS_j - j-toji mobilioji palaikymo stotis

MK_i - i-tasis mobilusis kompiuteris

9. $M(t_m) = \{\{Is\ \ agregato, Isejimu\ \ grupe, I\ \ agregata, Iejimu\ \ grupe\}\}$

10. $t_0 = 0;$

$mk(t_0) = 0$

$mps(t_0) = 2$

$A(t_0) = \{DB, MPS_1, MPS_2\}$

$$M(t_0) = \begin{matrix} \{\{MPS_*, & y_{*,4,*}, & DB, & x_{4,*}\}, \\ \{DB, & x_{4,*}, & MPS_*, & x_{*,4,*}\}, \\ \{MPS_*, & y_{*,5,*}, & MPS_*, & x_{*,5,*}\}, \\ \{THIS, & y_{i',1,j'}, & MK_{i'}, & x_{1,j'}\}, \\ \{MK_{i'}, & y_{1,j'}, & THIS, & x_{i',1,j'}\}\} \end{matrix}$$

11. $MK_i(H(e'_{1,1}))$: – **į MK atėjo įsijungimo signalas**

$mk(t_{m+1}) = mk(t_m) + 1$

$A(t_{m+1}) = A(t_m) \cup \{MK_{mk(t_{m+1})}\}$; - MK įtraukiamas į sistemą

$MK_i(st_k(t_0)) = \{\eta_k\}, kai\ \ k = \overline{1, mps(t_m)}$ - nustatomi pradiniai MPS signalų stiprumai

$MK_i(H(e''_1))$: **- MK tikrina MPS signalų stiprumus**

$MK_i(st_k(t_{m+1})) = \{\eta_k\}, kai\ \ k = \overline{1, mps(t_m)}$ - apskaičiuojami visu MPS signalų stiprumai

$MK_i(mps\_naujas(t_{m+1})) = k, kai\ \ \max(MK_i(st_k(t_{m+1}))), k = \overline{1, mps(t_m)}$ - randama MPS turinti stipriausia signalą

$M(t_{m+1}) = M(t_m) \cup \{MK_i, \quad y_{2,j'}, \quad MPS_{MK_i(mps\_naujas(t_{m+1}))}, \quad x_{2,j'}\}, jei$

$mps\_naujas(t_{m+1}) \neq mps(t_m)\ \ ir$ - jei atsirado

$\{MK_i, \quad y_{2,j'}, \quad MPS_{MK_i(mps\_naujas(t_{m+1}))}, \quad x_{2,j'}\} \notin M(t_m)$

MPS su stipresniu signalu, sukuriamas prisijungimui skirtas kanalas iš MK į MPS

$M(t_{m+1}) = M(t_m) \cup \{MPS_j, \quad y_{MPS_j(n(t_{m+1})),2,j'}, \quad MK_{MPS_j(n(t_{m+1}))}, \quad x_{2,j'}\}, jei$

$mps\_naujas(t_{m+1}) \neq mps(t_m)\ \ ir$ - jei

$\{MPS_j, \quad y_{MPS_j(n(t_{m+1})),2,j'}, \quad MK_{MPS_j(n(t_{m+1}))}, \quad x_{2,j'}\} \notin M(t_m)$

atsirado MPS su stipresniu signalu, sukuriamas prisijungimui skirtas kanalas iš MPS į MK

$MK_i(H(e'_{2,1}))$: **- į MK atėjo prisijungimo patvirtinimo signalas**

$$M(t_{m+1}) = M(t_m) \cup \{MK_i, \quad y_{3,j'}, \quad MPS_{MK_i(mps\_naujas(t_m))}, \quad x_{3,j'}\}, jei \quad \{MK_i, \quad y_{3,j'}, \quad MPS_{MK_i(mps\_naujas(t_m))}, \quad x_{3,j'}\} \notin M(t_m)$$ - sukuriamas

komunikacijai skirtas kanalas iš MK į naująją MPS

$$M(t_{m+1}) = M(t_m) \cup \{MPS_{MK_i(mps\_naujas(t_m))}, \quad y_{3,j'}, \quad MK_i, \quad x_{3,j'}\}, jei \quad \{MPS_{MK_i(mps\_naujas(t_m))}, \quad y_{3,j'}, \quad MK_i, \quad x_{3,j'}\} \in M(t_m)$$ - sukuriamas

komunikacijai skirtas kanalas iš naujos MPS į MK

$$M(t_{m+1}) = M(t_m) - \{MK_i, \quad y_{3,j'}, \quad MPS_{MK_i(mps\_naujas(t_m))}, \quad x_{3,j'}\}, jei \quad \{MK_i, \quad y_{3,j'}, \quad MPS_{MK_i(mps\_naujas(t_m))}, \quad x_{3,j'}\} \in M(t_m)$$ - naikinamas

prisijungimui skirtas kanalas iš MK į MPS

$$M(t_{m+1}) = M(t_m) - \{MPS_{MK_i(mps\_naujas(t_m))}, \quad y_{3,j'}, \quad MK_i, \quad x_{3,j'}\}, jei \quad \{MPS_{MK_i(mps\_naujas(t_m))}, \quad y_{3,j'}, \quad MK_i, \quad x_{3,j'}\} \in M(t_m)$$ - naikinamas

prisijungimui skirtas kanalas iš MPS į MK

$$M(t_{m+1}) = M(t_m) - \{MK_i, \quad y_{2,j'}, \quad MPS_{MK_i(mps(t_m))}, \quad x_{2,j'}\}, jei \quad \{MK_i, \quad y_{2,j'}, \quad MPS_{MK_i(mps(t_m))}, \quad x_{2,j'}\} \in M(t_m)$$ - naikinamas

bendravimui skirtas kanalas iš MK į senąją MPS

$$M(t_{m+1}) = M(t_m) - \{MPS_{MK_i(mps(t_m))}, \quad y_{2,j'}, \quad MK_i, \quad x_{2,j'}\}, jei \quad \{MPS_{MK_i(mps(t_m))}, \quad y_{2,j'}, \quad MK_i, \quad x_{2,j'}\} \in M(t_m)$$ - naikinamas

bendravimui skirtas kanalas iš senosios MPS į MK

$MK_i(H(e'_{3,2}))$: **- į MK atėjo atsijungimo nuo MPS patvirtinimo signalas**

$$M(t_{m+1}) = M(t_m) - \{MK_i, \quad y_{2,j'}, \quad MPS_{MK_i(mps(t_m))}, \quad x_{2,j'}\}, jei \quad \{MK_i, \quad y_{2,j'}, \quad MPS_{MK_i(mps(t_m))}, \quad x_{2,j'}\} \in M(t_m)$$ - naikinamas

bendravimui skirtas kanalas iš MK į MPS

$$M(t_{m+1}) = M(t_m) - \{MPS_{MK_i(mps(t_m))}, \quad y_{2,j'}, \quad MK_i, \quad x_{2,j'}\}, jei \quad \{MPS_{MK_i(mps(t_m))}, \quad y_{2,j'}, \quad MK_i, \quad x_{2,j'}\} \in M(t_m)$$ - naikinamas

bendravimui skirtas kanalas iš MPS į MK

$MPS_j(H(e'_{i,3,2}, e'_{i,3,3}))$: **- - iš i-tojo MK atėjo naujos MPS adresas ir prašymas atsijungti (**$i = \overline{1..\infty}$**)**

$MPS_j(mps\_nauja_i(t_{m+1})) = MK_i(mps\_naujas(t_m))$ - transakcijos rezultatą siųsti naujam MPS

Patvirtinu, kad straipsnis „The Usage of Dynamic PLA Formalism in Mobile Computing System“ buvo priimtas į KTU vasaros mokyklos „Formalūs sistemų analizės metodai informatikoje“ leidinį.

Mokyklos vadovas prof. habil. dr. H. Pranevičius

THE USAGE OF DYNAMIC PLA FORMALISM IN MOBILE COMPUTING SYSTEM

Lina Cibulskienė, supervisor Prof. habil. dr. Henrikas Pranevičius

Kaunas University of Technology, Faculty of Informatics Studentų str. 50 LT – 51368 Kaunas

Currently majority of systems are becoming more and more complex, exhibiting structural changes. The first step of implementing these systems in programming language, is analyzing and formalizing. One of the formalization languages – PLA, was recently extended for the sole purpose of formalization of dynamic systems.

This paper covers the usage of new extended formalization language (dynPLA) for mobile computing system, giving an example of said system specification.

1 Introduction

Currently many devices are becoming smaller and more powerful. They consume less power, and become highly mobile. Everyone have mobile phones, PDA's, and mobile computers. Almost all of them connect to the Internet at some point. So there is a need to support these mobile devices, so they can connect and reconnect to the different networks seamlessly, without disturbing the end user experience.

The mobile computing system will be analyzed in this paper. It is assumed that the network consists of two types of devices: mobile and fixed. Fixed devices are always on and always connected to the network. These devices include, but are not limited to database servers and mobile support stations (MSS). Mobile devices access data from database. They connect and reconnect to different MSS depending on the signal strength. All of this is done seamlessly to the end user and the data transfer is not disturbed.

DynPLA specification language will be used to formalize this system model, as standard PLA is not able to formalize such dynamic systems. Some guidelines will be given how to use dynPLA for complex dynamic system formalization.

The organization of the paper is as follows. In section2, dynPLA extensions are described. In section 3, the mobile computing system model is described. In section 4, developed guidelines are given for system formalization. Finally, the conclusions are provided in section 5.

2 Dynamic PLA model (dynPLA)

Dynamic Peace Linear Aggregates (dynPLA) formalization language is the extension of standard PLA [6]. This extension allows to define system model, which can change in time and is described as the introduction of new sets and operations. These sets are used to store information in certain moments. Hence in the course of time and system said sets change as well.

The set A with index t is the set of system aggregates at time t. That is, the set A represents the structure of system model in some precise time.

$A_{t+1} = A_t \cup \{A_{new}\}$ - where the aggregate A_{new} is added to the model.

$A_{t+1} = A_t - \{A_{old}\}$ - where the aggregate A_{old} is removed from the model.

When the new aggregate is introduced to the system, it is required to define its inputs and outputs. Furthermore, the sets of H, G, E' and E" with index t is also needed to define.

The following operations extend the specification of aggregates:

9. Addition of new input to the aggregate: $X_{t+1} = X_t \cup \{X_{new}\}$.
10. Removal of input from the aggregate: $X_{t+1} = X_t - \{X_{old}\}$.
11. Addition of new output to the aggregate: $Y_{t+1} = Y_t \cup \{Y_{new}\}$.
12. Removal of output from the aggregate: $Y_{t+1} = Y_t - \{Y_{old}\}$.
13. Creation of inner events in aggregate: $E''_{t+1} = E''_t \cup \{e''_{new}\}$, where e''_{new} is an inner event added to the aggregate.
14. Removal of inner events from the aggregate: $E''_{t+1} = E''_t - \{e''_{old}\}$.
15. Creation of outer events in aggregate: $E'_{t+1} = E'_t \cup \{e'_{naujas}\}$.
16. Removal of inner events from the aggregate: $E'_{t+1} = E'_t - \{e'_{senas}\}$.

Set H, G with index t contains all H and G operators at some specific time. H and G operators depend on inner and outer events. That's why the ability to modify inner and outer event sets requires modifying H and G sets, respectively.

$H_{t+1} = H_t \cup \{H(e'_{new})\}$ - the operator H is added, which processes the event e'_{new}.

$G_{t+1} = G_t \cup \{G(e'_{new})\}$ - the operator G is added, which processes the event e'_{new}.

The same rule applies to the removal of inner and outer events. That is, by removing inner or outer events, the operators H and G, which process these events must be removed too.

Dynamic PLA formalization language also uses matrix M with index t. This matrix defines connections, between system aggregates at certain time.

3 A Mobile Computing System model

A mobile computing system is a dynamic system [2], which consists of three main components: mobile host (MH), mobile support station (MSS) and database. Links between mobile computers and mobile support stations that provide wireless interface to mobile computers can change dynamically. In a mobile computing system, users carrying mobile portable computers can access database services from any location without requiring to maintain a fixed position in network.

Two main events exist in said system: the movement of mobile host through network (handoff operation) and transaction execution, when a mobile host query is executed in database.

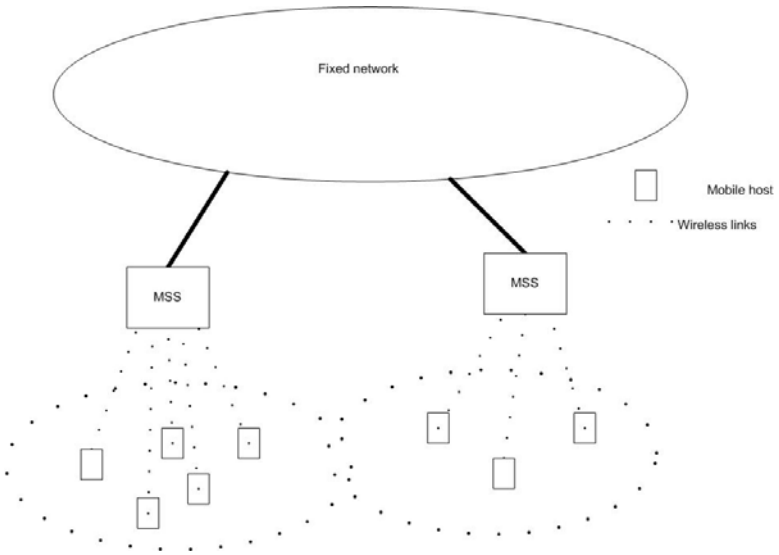


Figure 1. **A mobile Computing System.**

A typical mobile computing system consists of a number of mobile and fixed hosts (Figure 1). Fixed hosts are connected with each other via a fixed high-speed wired network and constitute the fixed part of the system. A mobile host (MH) connects to the fixed network via a wireless link. The fixed part of the network consists of a database and mobile support stations (MSS). Mobile support stations communicate with mobile hosts and the links between them can change dynamically. The geographical region in which mobile hosts can communicate with an MSS is called the cell of that MSS.

MSS acts as an interface between mobile hosts and the fixed part of the network; it forwards messages and data between the local mobile computer and fixed network.

4 Mobile computing system's specification using dynPLA

First of all, to specify mobile computing system it is needed to divide it into smaller elements. Such elements are above-mentioned mobile hosts (MH), mobile support stations (MSS) and database. Furthermore in order to correctly specify system it is needed to define the network as the main, controlling aggregate, because other aggregates do not coordinate events.

Events, which occur in the system, are not fully controlled by MH, or MSS or database. Usually these aggregates share the responsibility, but they never take full control. It follows thence that it is impossible to specify all connections and data transfer specifying some particular aggregate, because it doesn't have all the necessary information about other aggregates, which partake in certain events.

MH_i:

1. $X(t_m) \subseteq \{x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{3,1}, x_{3,2}\}$
2. $Y(t_m) \subseteq \{y_{1,1}, y_{2,1}, y_{3,1}, y_{3,2}, y_{3,3}\}$
3. $E'(t_m) \subseteq \{e'_{1,1}, e'_{1,2}, e'_{1,3}, e'_{2,1}, e'_{3,1}, e'_{3,2}\}$
4. $E''(t_m) \subseteq \{e''_1\}$

   e''_1 - the check of signal strength („beacon")
5. $\{e''_1\} \rightarrow \{\eta\}$
6. $z_v(t_m) \subseteq \{w(e''_1, t_m)\}$
7. $v(t_m) = \{mss(t_m), mss\_new(t_m), trans(t_m), st_k(t_m)\}$

   $mss(t_m)$ - MSS number, which is connected to MH(0 – if MH it is not connected)

   $mss\_new(t_m)$ - the next MSS

   $trans(t_m)$ - information about transaction execution (0 – no execution is in process)

   $st_k(t_m)$ - the strength of signal between MH and MSS_k
8. $t_0 = 0;$

   $X(t_0) = \{x_{1,1}\};$

   $Y(t_0) = \varnothing;$

   $E'(t_0) = \{e'_{1,1}\};$

   $E''(t_0) = \varnothing;$

   $z_v(t_0) = \varnothing;$

   $v(t_0) = \{mss(t_0), mss\_new(t_0), trans(t_0), st_k(t_0)\}$

   $mss(t_0) = 0$

   $mss\_new(t_0) = 0;$

   $trans(t_0) = 0;$

   $st_k(t_0) = 0, \quad k = \overline{1..\infty};$
9. $\hat{H}(e'_{1,1})$: – MH connections to the system

   */*proccesing */*

Figure 2. Specification of mobile host in dynPLA.

The specification of mobile host is shown in Figure 2.

In 1-4 lines all possible inputs, outputs, inner and outer events are described. Later, point 8, they are described in initial state t_0. All possible subsets of input, output, inner and outer event sets are described for the convenience and further references.

The best way to show dynPLA characteristics is to examine a particular dynamic event of the mobile computing system.

4.1 Handoff

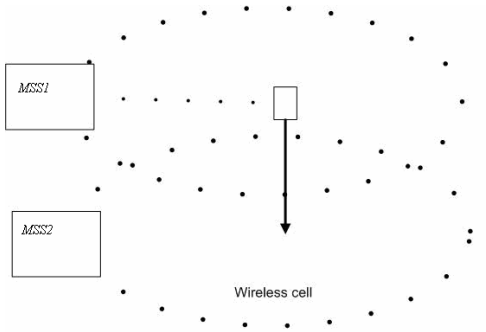
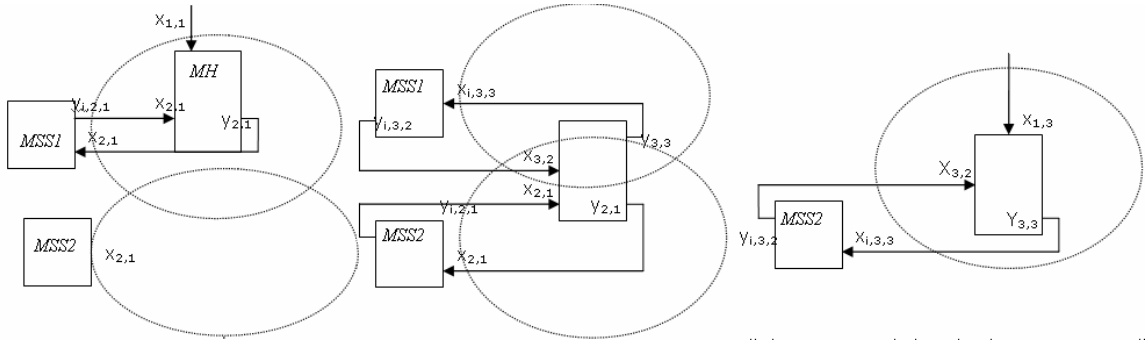


Figure 3. Handoff Operation.

Mobile hosts can cross the boundary between any two cells (Figure 3). In order to keep the connectivity of the mobile host to the fixed network a handoff process is implemented. During handoff the new cell's MSS takes the responsibility of providing a wireless interface to the mobile host [3].

Handoff process, including the connection of mobile host to the system and disconnection from it, is divided into three main parts. (All input and output indexes are marked exactly as in whole system specification.)



a) **b)** **c)**

Figure 4. Three main parts of handoff operation.

The connection of MH to its nearest MSS is show in figure 4a.

Each MSS broadcasts beacons over its wireless link. A mobile host monitors the wireless signal strength. The MSS, which broadcasts the strongest signal is considered the nearest and gets the greet message from the MH.

$MH(st_k(t_{m+1})) = \{\eta_k\}, when \quad k = \overline{1, mss(t_m)}$ - MH stores all signal strengths, received from different MSS.

$MH(mss\_new(t_{m+1})) = k, when \quad \max(MH(st_k(t_{m+1}))), k = \overline{1, mss(t_m)}$ - MH determines MSS with the strongest signal.

$M(t_{m+1}) = M(t_m) \cup \{MH, \quad y_{2,1}, \quad MSS1, \quad x_{2,1}\}$ - new channel is added to the *M* matrix. This connection is used for MH to send a greet message.

$M(t_{m+1}) = M(t_m) \cup \{MSS1, \quad y_{i,2,1}, \quad MH, \quad x_{2,1}\}$ - the greet acknowledgement message is sent through this channel

After exchange of these messages, MH is considered connected to the network. If at any point of time MH crosses the boundaries of cells and the signal of the neighboring MSS is stronger, mobile host initiates a handoff process. Firstly, MH sends greet message to the new MSS and waits for its acknowledgement using these channels.

$M(t_{m+1}) = M(t_m) \cup \{MH, \quad y_{2,1}, \quad MSS2, \quad x_{2,1}\}$

$M(t_{m+1}) = M(t_m) \cup \{MSS2, \quad y_{i,2,1}, \quad MH, \quad x_{2,1}\}$

After establishing new connection, MH sends notifying message to the old MSS, which deletes inputs and outputs related to the disconnected MH.

$M(t_{m+1}) = M(t_m) \cup \{MH, \quad y_{3,3}, \quad MSS1, \quad x_{i,3,3}\}$.

Figure 4c shows mobile hosts disconnection from network. Disconnection from network is similar to disconnection from old MSS. All channels are the same (disconnection notify message and acknowledgement), the only difference is that this disconnection is initiated by an outer event.

$M(t_{m+1}) = M(t_m) - \{MH, \quad y_{3,3}, \quad MSS2, \quad x_{i,3,3}\}$

5 Conclusions

As it is shown above, dynPLA is essential for describing dynamic system models. Without features, such as connection matrix *M* with index *t,* or the ability to modify the sets of input/output, or inners/outer events, it would be impossible to define a system model with ability to change its structure and/or behavior.

References

[17] **Š.Packevičius, A.Kazla, H.Pranevičius**. Extensions of PLA specification for dynamic system formalization. 2006, p. 1-2.

[18] **O.Ulusoy.** Real-Time Data Management for Mobile Computing. Proceedings of the International Workshop on Issues and Applications of Database Technology. 1998, p. 233-240.

[3] **R.Caceres, V.N.Padmanabhan.** Fast and Scalable Handoffs for Wireless Internetrworks. Proceedings of ACM MobiCom'96. 1996.