VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigtinė tiesinio laiko logika

Finite Linear Temporal Logic

Magistro baigiamasis darbas

Atliko: 2 magistrantūros kurso, 8 grupės studentė Rima Želvytė
Darbo vadovas: Doc. dr. Stanislovas Norgėla
Recenzentė: A. Šalaviejienė

Vilnius - 2008

Turinys

1. Darbo temos aktualumas

Logika - mokslas, tiriantis mąstymo, samprotavimo dėsnius ir jo formas. Logikos pagalba mes galime nagrinėti teiginius ir įrodymus bei išsiaiškinti, teisingi jie ar ne. Logika yra naudojama daugelyje gyvenimo sričių: filosofijoje, matematikoje, kompiuterinėse programose, ir t.t.

Plačiai yra paplitusi klasikinė logika. Šios logikos formulėmis galima nagrinėti uždavinius, kurių objektų būsena yra statinė, tačiau taip pat yra uždavinių, kurių būsena yra dinaminė, t.y. reikšmės ne visada vienodos, jos kinta. Todėl jų negalima išspręsti klasikinėmis logikos formulėmis. Reikalinga kita logika, kuri atsižvelgtų, kaip keičiasi objektų reikšmės kintant laikui, kadangi uždavinio vertė gali visiškai skirtis jei jį nagrinėsime kitu laiko momentu.

Tiesinės baigtinės laiko logikos taisyklės gali nagrinėti ir tokius uždavinius, kuriuose reikia atsižvelgti į reikšmių kaitą atitinkamais laiko momentais. Pagrindinis šios logikos skirtumas: tikslesnis uždavinio detalizavimas, papildomos uždavinio nagrinėjimo (sprendimo) priemonės, t.y. naudosime naujus operatorius - laiko.

Svarbu atsižvelgti ir į tai, kad uždaviniuose informacija gali būti nepilna. Sprendžiant reikia apibrėžti visus galimus variantus, t.y. esant tam tikrom sąlygom būtina atlikti atitinkamus veiksmus.

Tiesinė baigtinė logika neapsiriboja vien tik praeitimi, taip pat yra atsižvelgiama ir į ateitį: nuo to laiko kai žingsnis po žingsnio einame nuo pradinės formos iki tikslo. Šios logikos struktūra yra baigtinė į abi puses, t.y. ir į praeitį ir į ateitį. Sprendžiant uždavinį, yra ieškoma būsenos, kurioje formulė būtų modelis. Kartais vienos būsenos nepakanka, todėl yra įtraukiami būsenų intervalai, kuriuose nurodoma būsenų ribos, laiko momentai, kur yra pasiekiamas tikslas, t.y. įrodoma, kad formulė turi modelį. Ir taip vykdant pažingsniui, atskiromis formulės dalimis einama prie tikslo ir konstruojama apribojimų aibė. Šioje aibėje turi būti tiksliai apibrėžti būsenų prioritetai: kokia būsena eina po kokios, kuri yra einamoji, ankstesnė, vėlesnė, atitinkamu atveju - sekanti.

Buvo išbandyta daug modelių ir logikų planavimo uždaviniams spręsti. Vienas jų yra laiko logika.

Ši logika yra naudinga šiuo atveju todėl, kad yra išraiškinga ir galime apibrėžti ne tik veiksmus, tačiau ir juos priskirti atitinkamiems laiko momentams. Suformulavus uždavinio

pradines sąlygas tiesinio laiko logikos išraiškomis ir pritaikius planavimo sprendimo taisykles galima pasiekti tikslą ir nustatyti kelią iki jo.

Sprendimo eigoje galima susidurti, kad vienu metu bus galimi keli veiksmai toje pačioje būsenoje. Šią problemą išsprendžiama tiesos laipsnių metodu. Remiantis šiuo modeliu, planavimo problema yra apibrėžiama naudojant ir tiesos laipsnius ir klasikines teisingumo reikšmes. Be to gali būti apibrėžtos klasės turinčios skirtingą taikomumo laipsnį.

Klasikinis modelis, kai veiksmai ir sekos išreiškiamos teisingumo reikšmėmis, yra ribotas, nes dažnai reikšmė nėra visiškai teisinga, arba visiškai klaidinga. Yra tarpinės teisingumo reikšmės arba tiesos laipsniai. Šiame modelyje tiesos laipsniai apibrėžia mūsų žinias, būsenos neapibrėžtumą arba įvykio sėkmės tikimybę, t.y. sritis yra visi realieji skaičiai iš [0;1], nurodomas sekos,veiksmo tinkamumo laipsnis

Kai kuri naudinga informacija gali atsirasti tik uždavinio sprendimo metu, galimos įvairios sprendimo modifikacios, kurios atsižvelgtų ir į tokius faktorius: mokymosi, numatomumo modeliai.

2. Darbo tikslas

Šio darbo tikslas: pratęsti baigiamojo bakalauro darbą, toliau gilintis į laiko logiką, jos panaudojimą sprendžiant uždavinius.

Baigiamajame bakalauro darbe:

- ✓ remiantis straipsniais (žr. [CM97], [MSC97]) suformuluotos tiesinės baigtinės logikos(BTL) skaičiavimo taisyklės
- ✓ remiantis straipsniais (žr. [CM97], [MSC97]) predikatų logikos virš baigtinio laiko logikos(PLBLL) skaičiavimo taisyklės

Magistro baigiamajame darbe:

- ✓ analizuotas Turingo mašinų(žr. [Nor04]) darbas virš baigtinės laiko struktūros. Suformuluoti reikalingi predikatai, kurių pagalba aprašomos perėjimų formulės.
- ✓ remiantis straipsniu (žr. [ML02]) aprašyti baigtinės tiesinio laiko logikos sąlyginio skaičiavimo taisykles.
- ✓ analizuoti planavimo uždavinius ir jų sprendimui taikyti baigtinės tiesinio laiko logikos sąlyginio skaičiavimo taisykles.

3. Baigtinė tiesinio laiko logika

3.1. Tradicinės loginės operacijos

Pradėsime nagrinėti kitą, baigtinę tiesinio laiko logiką. Joje gali būti tradicinės loginės operacijos, tokios kaip ir klasikinėje (žr.[Nor04]):

¬ : neiginys, kuris yra tik prieš loginius kintamuosius,

& : konjunkcija,

∨ : disjunkcija.

3.2. Laiko operatoriai

Aprašytųjų loginių operacijų nepakanka, norint peržvelgti uždavinio reikšmes laikui kintant. Todėl yra reikalingi laiko operatoriai. Įsivesime šešis : tris, kad apibrėžtume būsenas ateityje, ir taip pat tris, kad išnagrinėtume kaip kinta reikšmės praeityje. Juos skaitome taip:

□ : visada ateityje,

⬰ : visada praeityje,

◇ : kai kada ateityje,

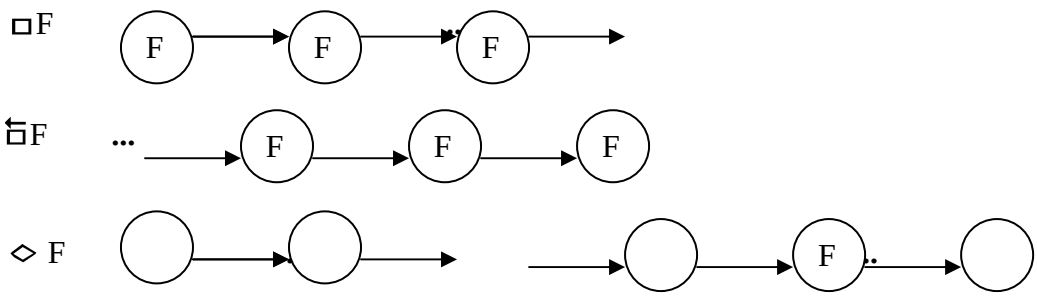
⬱ : kai kada praeityje,

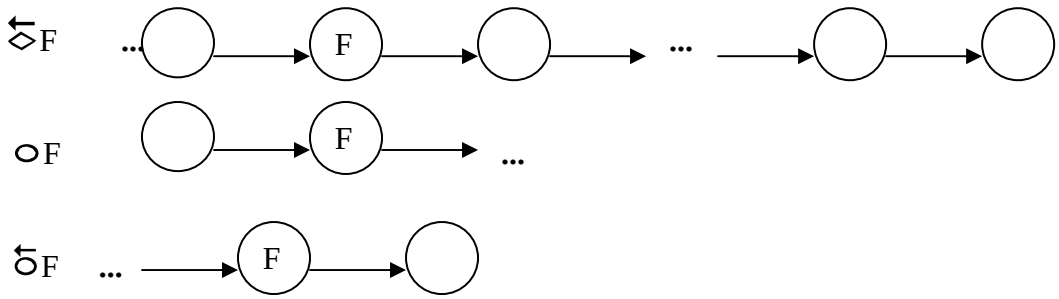
○ : sekantis (kitu laiko momentu),

⟲ : ankstesnis (kitu laiko momentu),

3.2.1. Laiko operatorių vaizdavimas schemomis

Laiko operatorius taip pat galime paaiškinti schemomis:





3.3. Pagalbinės formulės

Laikysime, kad:

$\neg\Box F = \Diamond \neg F$,

$\neg\overleftarrow{\Box}F = \overleftarrow{\Diamond}\neg F$,

$\neg\Diamond F = \Box\neg F$,

$\neg\overleftarrow{\Diamond}F = \overleftarrow{\Box}\neg F$,

PVZ.: jei $\neg\Box F = \Diamond \neg F$, tai reiškinį tariame : “ formulė: netiesa, kad visada ateityje F ekvivalenti tvirtinimui kartais ateityje nėra F”.

3.4. Laiko struktūra. Formulės struktūra

1 apibrėžimas: Laiko struktūra vadinsime būsenų seką $T = <q_0, q_1, ..., q_i>$, čia $q_0 < q_1 <...< q_i$.

Formulės F struktūra vadinsime porą <T,V>, čia:

T yra kuri nors laiko struktūra,

V- formulės loginių kintamųjų interpretacijų būsenose aibė, t.y. aibė funkcijų (priklausančių nuo būsenų), apibrėžtų formulės F loginių kintamųjų aibėje su reikšmėmis iš {t, k}.

3.5. Formulės teisingumas

2 apibrėžimas: Formulės F teisingumas struktūros < T,V > būsenoje q_i nusakomas indukcija pagal formulės pavidalą:

1) jei F yra loginis kintamasis, tai F yra teisinga tada ir tik tada, kai jis (loginis kintamasis) teisingas būsenoje q_i,
2) jei F = ¬ G, tai F teisinga tada ir tiktai tada, kai G klaidinga būsenoje q_i,
3) jei F = G ∨ H, tai F teisinga tada ir tiktai tada, kai bent viena iš G, H teisinga būsenoje q_i,
4) jei F = G & H, tai F teisinga tada ir tiktai tada, kai abi teisingos būsenoje q_i,
5) jei F = G → H, tai F teisinga tada ir tiktai tada, kai G klaidinga arba H teisinga būsenoje q_i,
6) jei F = □ G, tai F teisinga tada ir tiktai tada, kai G teisinga visose būsenose $q_m >= q_i$,
7) jei F = ⬚ G, tai F teisinga tada ir tiktai tada, kai visose būsenose $q_n <= q_i$ teisinga G,
8) jei F = ◇ G, tai F teisinga tada ir tiktai tada, kai atsiras tokia q_m T, kad $q_m >= q_i$ ir G yra teisinga būsenoje q_m,
9) jei F = ⬚ G, tai F teisinga tada ir tiktai tada, kai atsiras tokia q_n T, kad $q_n <= q_i$ ir G yra teisinga būsenoje q_n,
10) jei F = ○G, tai F teisinga tada ir tiktai tada, kai q+1 T, arba G yra teisinga būsenoje q+1,
11) jei F = ⬚G, tai F teisinga tada ir tiktai tada, kai $q \neq q_0$, arba G yra teisinga būsenoje q-1,

Sakoma, kad formulė F yra teisinga struktūroje < T, V >, jei ji teisinga tos struktūros pradinėje būsenoje, t.y. būsenoje q_i. Formulė F įvykdoma, jei atsiras struktūra, kurioje ji teisinga.

3.6. Modelis

3 apibrėžimas: Tiesinė baigtinė struktūra vadinama modeliu (žr.[CM97], [MSC97]), jei formulė F yra joje teisinga.

Baigtinės logikos skaičiavimo taisyklių pagalba galime nustatyti ar neigiamos normaliosios formos (neiginys gali būti tik prieš loginius kintamuosius) formulei egzistuoja modelis.

3.7. BTLL skaičiavimo taisyklių žymėjimai

Taisyklėse raidėmis F ir G žymėsime predikatų logikos formules. Būsenoms nurodomi tam tikri apribojimai:

q_i – būsena,

q’ ir q” – naujos būsenos, nesutinkamos šakoje,

$[q_n,q_i]$ ir $[q_i,q_m]$ – struktūros intervalai,

q_r – einamoji būsena,

q_m – didžiausia struktūros būsena,

q_n – mažiausia struktūros būsena,

$s(q_i)$ – sekantis, elementas po q_i,

$a(q_i)$ – ankstesnis, elementas prieš q_i,

Gali būti nurodyta, kad apribojimai išvedami iš aukščiau šakoje sutinkamų apribojimų. Juos žymėsime Δ. Pradinės formulės atveju apribojimų aibė tuščia. Sakysime, kad apribojimų aibė prieštaringa, jei iš Δ išvedama nelygybė $q_i < q_i$, kur q_i - kuri nors būsena. Sakysime, kad aibėje Δ visiška tvarka, jei kokios bebūtų būsenos q’ ir q”, iš Δ išvedama arba q’ = q”, arba q’ < q” arba q”<q’.

Nagrinėsime reiškinius pavidalo F : q_i, čia F yra formulė, kuriai nustatinėsime ar egzistuoja jai modelis. Pradiniais duomenimis gali būti ir ne viena formulė, o kuri nors baigtinė formulių aibė $\{F_1,..,F_s\}$.

3.8. BTLL skaičiavimas

3.8.1. BTLL skaičiavimo taisyklės

(∨) $\dfrac{F \vee G :q_i}{F :q_i \quad G :q_i}$ $\dfrac{F \vee G :[q_i, q_m]}{F :[q_i, q_m] \quad G :[q_i, q_m]}$

(&) $\dfrac{F\&G :q_i}{F :q_i \quad G :q_i}$ $\dfrac{F\&G :[q_i, q_m]}{F :[q_i,q_m] \quad G :[q_i,q_m]}$

(□) $\dfrac{\Box F : q_i}{F :[q_i,q_m]}$ $\dfrac{\Box F :[q_i, q_m]}{F : \Delta [q_r, q_m]}$

(⬱) $\dfrac{\overleftarrow{\Box} F : q_i}{F :[q_n,q_i]}$ $\dfrac{\overleftarrow{\Box} F : [q_i, q_m]}{F : \Delta [q_n,q_r]}$

(◇) $\dfrac{\Diamond F :q_i}{F :q', \; q_i <= q'}$ $\dfrac{\Diamond F : [q_i,q_m]}{F : \Delta [q',q_m], \; q_i <= q'}$

(⬰) $\dfrac{\overleftarrow{\Diamond} F :q_i}{F : q', \; q' <= q_i}$ $\dfrac{\overleftarrow{\Diamond} F :[q_i,q_m]}{F : \Delta [q_n, q'], \; q' <= q_i}$

(○) $\dfrac{\bigcirc F :q_i}{F : s(q_i) : \; q_i \neq q_m}$ $\dfrac{\bigcirc F :[q_i, q_m]}{\bot}$

(⬲) $\dfrac{\overleftarrow{\bigcirc} F :q_i}{F : a(q_i) \; q_i \neq q_n}$ $\dfrac{\overleftarrow{\bigcirc} F :[q_i, q_m]}{\bot}$

Būsenų kolizijos taisyklė:

(Bkt) $\dfrac{q_i <= q', \quad q_i <= q''}{q_i <= q', \; q_i <= q'', \; q' <= q'' \quad q'' <= q'}$

3.8.2. Uždara šaka

Sakoma, kad išvedimo medyje šaka uždara, jei tenkinama bent viena sąlyga:

- apribojimų aibė Δ yra prieštaringa,
- joje sutinkamos dvi formulės pavidalo F : q_i, ¬F :q_i ,
- šakoje sutinkama formulė F :q_{i+1} ir iš Δ išvedama, kad q_i => q_{i+1} (q_i žymi pradinę būseną),
- jei joje sutinkamas simbolis ⊥.

Jei šaka nėra uždara, tai ji vadinama atvira.

3.8.3. Išplėstoji būsena

4 apibrėžimas: Sakykime I yra išvedimo paieškos medžio šaka. Būsena q_i šakoje I vadinama išplėstąją, jei tenkina sąlygas:

1) koks bebūtų aptinkamas šakoje I reiškinys pavidalo □ F :q_i ir kokia bebūtų būsena q_m, jei Δ├ q_i<= q_m, tai F: q_m irgi aptinkamas šakoje I,
2) koks bebūtų aptinkamas šakoje I reiškinys pavidalo ⬚ F :q_i ir kokia bebūtų būsena q_n, jei Δ ├ q_n<= q_i, tai F: q_i irgi aptinkamas šakoje I,
3) jei šakoje I aptinkamas reiškinys F&G :q_i ir Δ├ q_i= q_m, tai joje yra ir F :q_m bei G :q_m,
4) jei šakoje I aptinkamas reiškinys F ∨ G :q_i ir Δ├ q_i= q_m, tai joje yra ir F: q_m arba G: q_m,
5) jei šakoje I aptinkamas reiškinys ◇ F :q_i, tai atsiras toks q_m, kad Δ├ q_i<= q_m ir šakoje yra F :q_m,
6) jei šakoje I aptinkamas reiškinys ⬚ F :q_i, tai atsiras toks q_n, kad Δ├ q_n<= q_i ir šakoje yra F :q_n,
7) jei šakoje I aptinkamas reiškinys ○F :q_i ir Δ├ q_i<= q’, tai atsiras toks q’’, kad šakoje yra F :q’’ ir Δ├ q’’= s(q_i),
8) jei šakoje I aptinkamas reiškinys ⬚F :q_i ir Δ├ q’<= q_i, tai atsiras toks q’’, kad šakoje yra F :q’’ ir Δ├ q_i= a(q_i).

Šaka vadinama pilna, jei visos šakoje aptinkamos būsenos yra išplėstosios ir šakos būsenų apribojimų aibėje Δ įvesta visiška tvarka.

3.9.TBL skaičiavimo pavyzdžiai

3.9.1. Pavyzdys, kai formulė turi modelį

Užduotis: Nustatyti ar formulė F = ◇ p& ◇ (q & □¬p) turi modelį.

Sprendimas:

◇ p& ◇ (q & □¬p) : q_1

◇ p : q_1

p : q', q_1 <= q'

◇ (q & □¬p) : q_1

(q & □¬p) : q", q_1 <= q"

q : q", q_1 <= q"

□¬p : q"

¬p : [q",q_m]

p_1 <= q"

Išvedime susiduriama su keliais tais pačiais ◇ laiko operatoriais, todėl reikia taikyti būsenų koalizijos taisyklę, tam, kad būtų nustatytas būsenų eiliškumas.

(Bkt) q_1 <= q'

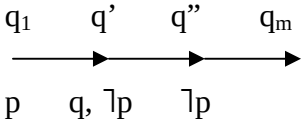
q_1 <= q"

q"<= q'"

q_1 <= q' q_1<= q"

q' <= q" q" <= q'

Rezultatas: Nagrinėjama formulė F **turi modelį,** kai:



3.9.2. Pavyzdys, kai formulė neturi modelio

Užduotis: Nustatyti ar formulė F = ◇ p& ◇ (q & □¬p&¬q) turi modelį.

Sprendimas:

◇ p& ◇ (q & □¬p&¬q) : q_1

◇ p : q_1

p : q', q_1<= q'

◇ (q & □¬p&¬q) : q_1

(q & □¬p&¬q) : q", q_1<= q"

q : q", q_1<= q"

□¬p : q"

¬p : [q",q_m]

p_1 <= q"

&¬q :q"

p_1 <= q"

Išvedime susiduriama su keliais tais pačiais ◇ laiko operatoriais, todėl reikia taikyti būsenų koalizijos taisyklę, tam, kad būtų nustatytas būsenų eiliškumas.

(Bkt) q_1 <= q'

q_1 <= q"

q"<= q'"

q_1 <= q' q_1<= q"

q' <= q" q" <= q'

Rezultatas: Nagrinėjamai formulei F neegzistuoja modelis, nes apribojimų aibė Δ yra prieštaringa(pirmu atveju : būsenoje q", kitu - q" ir q'):

q_1 q' q" q_m

p **q**, ¬p, ¬**q** ¬p

arba

q_1 q" q' q_m

q, ¬p, ¬**q** **p,** ¬**p** ¬p

3.10. Modelio radimas

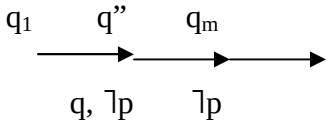
5 apibrėžimas: jei šaka pilna (4.10. Išplėstoji būsena) ir atvira (žr. 4.9. Uždara šaka), tai formulė įvykdoma.

1 Pavyzdys:

Užduotis: Taikant BTLL skaičiavimą nustatyti ar formulė F = ◇ (q & □¬p) turi modelį.
Sprendimas:

◇ (q & □¬p) : q_1
(q & □¬p) : q", q_1<= q"
q : q", q_1<= q"
□¬p : q"
¬p : [q",q_m]
p_1 <= q"

Rezultatas:



Formulė įvykdoma, nes šakos yra atviros ir pilnos.

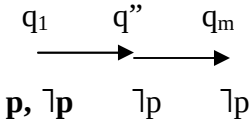
2 Pavyzdys:

Užduotis: Taikant BTLL skaičiavimą nustatyti ar formulė F = (p & □¬p) turi modelį.
Sprendimas:

(p & □¬p) : q_1
p : q_1

□¬p : q_1
¬p : [q_1,q_m]

Rezultatas:



Šakoje sutinkama šaka, nėra atvira: joje sutinkamos dvi formulės pavidalo F : q_i, ¬F :q_i ,

Teiginys: F turi modelį, t.y. išvedimo paieškos medyje atsiras pilna ir atvira šaka.

Įrodymas: Galimos operacijos(žr. **4.8. BTLL skaičiavimo taisyklės**):

$\vee$, &, $\Box$, $\overline{\Box}$, $\Diamond$, $\overleftarrow{\Diamond}$, $\circ$, $\overleftarrow{\circ}$.

Imame disjunkcijos taisyklę:

$$(\vee)\quad \frac{F \vee G \quad :q_i}{F :q_i \quad G :q_i}$$

Kai yra disjunkcija tarp F ir G būsenoje q_i, tai bent viena šaka turi būti atvira ir pilna, tada formulė turės modelį, t.y. arba F turi būti teisinga būsenoje q_i, arba G turi būti teisinga būsenoje q_i. Žinoma ir abi šakos gali būti teisingojs šioje būsenoje.

$$(\vee)\quad \frac{F \vee G \quad :[\, q_i\, ,\, q_m]}{F :[\, q_i\, ,\, q_m] \quad G :[\, q_i,\, q_m]}$$

Kai yra disjunkcija tarp F ir G būsenų intervale, tai bent viena šaka turi būti atvira ir pilna, tada formulė turės modelį, t.y. arba F turi būti teisinga būsenų intervale :[q_i, q_m] , arba G turi būti teisinga būsenų intervale :[q_i, q_m]. Žinoma ir abi šakos gali būti pilnos ir atviros būsenų intervale :[q_i, q_m].

Imame konjunkcijos taisyklę:

$$(\&)\quad \frac{F\&G :q_i}{\begin{array}{l} F \quad :q_i \\ G \quad :q_i \end{array}}$$

Kai yra konjungcija tarp F ir G būsenoje q_i, tai abi šakos turi būti atviros ir pilnos, tada formulė turės modelį, t.y. formulė F turi būti teisinga būsenoje q_i, ir būtinai formulė G turi būti teisinga būsenoje q_i.

$$(\&)\quad \frac{F\&G \quad :[\, q_i, q_m]}{\begin{array}{l} F \quad :[q_i, q_m] \\ G \quad :[q_i, q_m] \end{array}}$$

Kai yra konjunkcija tarp F ir G būsenų intervale, tai abi šakos turi būti atviros ir pilnos, tada formulė turės modelį, t.y. F turi būti teisinga būsenų intervale :[q_i, q_m] , ir būtinai G turi būti teisinga būsenų intervale :[q_i, q_m].

Imame laiko operatoriaus „visada ateityje“ taisyklę:

$$(\Box) \qquad \frac{\Box F \ : q_i}{F \ :[q_i, q_m]}$$

Kad F turėtų modelį, reikia, kad ji būtų pilna ir atvira visuose būsenų intervalo $[q_i,q_m]$ taškuose.

$$(\Box) \qquad \frac{\Box F \ :[q_i, q_m]}{F \ : \Delta [q_r, q_m]}$$

Kad F turėtų modelį, reikia, kad ji būtų pilna ir atvira visuose būsenų intervalo $[q_r,q_m]$ taškuose. Taip pat tenkinti visus prieš tai apibrėžtus būsenų ribojimus.

Imame laiko operatoriaus „visada praeityje“ taisyklę:

$$(\overleftarrow{\Box}) \qquad \frac{\overleftarrow{\Box} F \ : q_i}{F \ :[q_n, q_i]}$$

Kad F turėtų modelį, reikia, kad ji būtų pilna ir atvira visuose būsenų intervalo $[q_n,q_i]$ taškuose.

$$(\overleftarrow{\Box}) \qquad \frac{\overleftarrow{\Box} F \ : [q_i, q_m]}{F \ : \Delta [q_n, q_r]}$$

Kad F turėtų modelį, reikia, kad ji būtų pilna ir atvira visuose būsenų intervalo $[q_n,q_r]$ taškuose. Taip pat tenkinti visus prieš tai apibrėžtus būsenų ribojimus.

Imame laiko operatoriaus „kai kada ateityje“ taisyklę:

$$(\Diamond) \qquad \frac{\Diamond F \ :q_i}{\begin{array}{l} F \ :q', \\ q_i <= q' \end{array}}$$

Kad F turėtų modelį, reikia, kad rasti bent vieną būseną $q_i <= q'$, kurioje F būtų pilna ir atvira.

$$(\Diamond) \qquad \frac{\Diamond F \ : [q_i, q_m]}{\begin{array}{l} F \ : \Delta [q', q_m], \\ q_i <= q' \end{array}}$$

Kad F turėtų modelį, reikia, kad rasti bent vieną būseną $q_i <= q'$ iš intervalo $[q',q_m]$, kurioje F būtų pilna ir atvira.

Imame laiko operatoriaus „kai kada praeityje“ taisyklę:

$$(\overleftarrow{\Diamond}) \qquad \frac{\overleftarrow{\Diamond} F \quad : q_i}{\begin{array}{l} F \quad : q', \\ q' <= q_i \end{array}}$$

Kad F turėtų modelį, reikia, kad rasti bent vieną būseną q’ <= q_i , kurioje F būtų pilna ir atvira.

$$(\overleftarrow{\Diamond}) \qquad \frac{\overleftarrow{\Diamond} F \quad : [q_i, q_m\ ]}{\begin{array}{l} F \quad : \Delta\,[q_n, q'], \\ q' <= \ q_i \end{array}}$$

Kad F turėtų modelį, reikia, kad rasti bent vieną būseną q’ <= q_i iš intervalo $[q_{\cdot}, q_m\ ]$, kurioje F būtų pilna ir atvira.

Imame laiko operatorių „sekantis (kitu laiko momentu)“:

$$(\bigcirc) \qquad \frac{\bigcirc F \quad : q_i}{\begin{array}{l} F \ : s(q_i) \ : \\ q_i \neq q_m \end{array}}$$

Kad F turėtų modelį, reikia, kad rasti būseną q’, kurioje F būtų atvira ir pilna ir tenkintų šiuos apribojimus: q‘ > q_i, ir q_i negali būtų lygus q_m , t.y. negali būti paskutinė būsena intervale.

$$(\bigcirc) \qquad \frac{\bigcirc F \quad : [\, q_i\, , q_m]}{\perp}$$

Šiuo atveju F niekada neturės modelio, kadangi yra prieštaravimas: negalima rasti sekančios būsenos po paskutinės būsenos intervale, t.y. po q_m.

Imame laiko operatorių „ankstesnis (kitu laiko momentu)“:

$$(\overleftarrow{\bigcirc}) \qquad \frac{\overleftarrow{\bigcirc} F \quad : q_i}{\begin{array}{l} F \ : a(q_i) \\ q_i \neq q_n \end{array}}$$

Kad F turėtų modelį, reikia, kad rasti būseną q’, kurioje F būtų atvira ir pilna ir tenkintų šiuos apribojimus: q_i > q‘, ir q_i negali būtų lygus q_n , t.y. negali būti pradinė būsena intervale.

$$(\overleftarrow{\bigcirc}) \qquad \frac{\overleftarrow{\bigcirc} F \quad : [\, q_i\, , q_m]}{\perp}$$

Šiuo atveju F niekada neturės modelio, kadangi yra prieštaravimas: negalima rasti ankstesnės būsenos iki pirmos būsenos intervale, t.y. iki q_n.

Pastaba: Jei išvedime susiduriama su keliais tais pačiais $\Diamond$, $\overleftarrow{\Diamond}$ laiko operatoriais reikia taikyti būsenų koalizijos taisyklę, tam, kad būtų nustatytas būsenų eiliškumas.

4. Predikatų logika virš baigtinės laiko struktūros

4.1. Predikatų logika

4.1.1. Kvantoriai

Teiginių logikos abėcėlę praplečiame dviem kvantoriais(žr.[Nor04]):

$\forall$: bendrumo kvantorius,

$\exists$: egzistavimo kvantorius.

Naudojant šiuos kvantorius galima kiekybiškai apibūdinti objektų sritį. Kvantorius naudojame tik kartu su individiniais kintamaisiais (žr. Priedas, 3.2.Kintamieji) ir vadiname **kvantoriniais kompleksais**.

PVZ.: jei $\forall x$, tai tariame : “kiekvienam x”, “kiekvienam x teisinga”,

jei $\exists x$, tai tariame : “egzistuoja x”,”yra toks x”.

4.1.2. Laisvoji ir suvaržytoji įeitys

5 apibrėžimas : Individinio kintamojo x įeitis formulėje F vadinama suvaržytąja (žr.[Nor04]), jei ji patenka į kvantorinio komplekso $\forall x$ arba $\exists x$ veikimo sritį. Priešingu atveju nagrinėjamoji individinio kintamojo įeitis vadinama laisvąja**.**

4.2. Predikatų logika virš baigtinės laiko struktūros

4.2.1. Viršutinė būsenų skaičiaus riba

Apibrėšime baigtinį skaičiavimą(žr. [CMP99]), kuriam yra būtina turėti konstantą k-viršutinę kintamųjų skaičiaus ribą. Naujų kintamųjų skaičius negali viršyti k, jei taip įvyksta, skaičiavimas yra nutraukiamas.

Aprašysime predikatų logikos virš baigtinės laiko struktūros skaičiavimo taisykles (PLBLS), kurių pagalba galime nustatyti ar neigiamos normaliosios formos (neiginys gali būti tik prieš loginius kintamuosius) formulei F egzistuoja modelis.

4.2.2. PLBLS skaičiavimo taisyklių žymėjimai

Taisyklėse raidėmis F ir G žymėsime predikatų logikos formules, taip pat nurodomi tam tikri apribojimai būsenoms:

q_i – būsena,

$[q_n, q_i]$ ir $[q_i, q_m]$ – struktūros intervalai,

q’ ir q’’ – naujos būsenos, nesutinkamos šakoje,

q_r – einamoji būsena,

q_m – didžiausia struktūros būsena,

q_n – mažiausia struktūros būsena.

Gali būti nurodyta, kad apribojimai išvedami iš aukščiau šakoje sutinkamų apribojimų . Juos žymėsime Δ. Pradinės formulės atveju apribojimų aibė tuščia. Sakysime, kad apribojimų aibė prieštaringa, jei iš Δ išvedama nelygybė $q_i < q_i$, kur q_i kuri nors būsena. Sakysime, kad aibėje Δ visiška tvarka, jei kokios bebūtų būsenos q’ ir q”, iš Δ išvedama arba q’ = q”, arba q’ < q” arba q”<q’.

Nagrinėsime reiškinius pavidalo F : q_i, čia F yra formulė, kuriai nustatinėsime ar egzistuoja modelis(žr. 4.6 Modelis). Pradiniais duomenimis gali būti ir ne viena formulė, o kuri nors baigtinė aibė $\{F_1,..,F_s\}$.

Kintamuosius žymėsime :

suvaržytieji : x, y, z,..,

laisvieji : a, b, c,..,

metakintamieji : α (i), β(i), γ(i),.., kur i=0, 1, ...,m

5.1. PLBLS skaičiavimas

5.1.1. PLBLS skaičiavimo taisyklės(žr.[CMP99]):

(∨) $\dfrac{F \vee G : q_i}{F : q_i \quad G : q_i}$ $\qquad$ $\dfrac{F \vee G : [q_i, q_m]}{F : q_i, \quad G : q_i, \quad F \vee G : [q_{i+1}, q_m]}$

(&) $\dfrac{F\&G : q_i}{F : q_i \quad G : q_i}$ $\qquad$ $\dfrac{F\&G : [q_i, q_m]}{F : [q_i, q_m] \quad G : [q_i, q_m]}$

(□) $\dfrac{\Box F : q_i}{F : [q_i, q_m]}$ $\qquad$ $\dfrac{\Box F : [q_i, q_m]}{F : \Delta [q_r, q_m]}$

(◇) $\dfrac{\Diamond F : q_i}{F : q', \; q_i <= q'}$ $\qquad$ $\dfrac{\Diamond F : [q_i, q_m]}{F : \Delta [q', q_m], \; q_i <= q'}$

(○) $\dfrac{\bigcirc F : q_i}{F : s(q_i) : \; q_i \neq q_m}$ $\qquad$ $\dfrac{\bigcirc F : [q_i, q_m]}{\perp}$

(∀) $\dfrac{\forall x F(x) : q_i}{F(\alpha(j)) : q_i, \quad \forall x F(x) : [q_{i+1}, q_m],}$ $\qquad$ $\dfrac{\forall x F(x) : [q_i, q_m]}{F(\alpha(j)) : [q_i, q_j], \quad \forall x F(x) : [q_{j+1}, q_m],}$

čia:

α (j) metakintamasis, t.y., bet kuris laisvasis kintamasis sutinkamas šakoje aukščiau, α (j) C, C – laisvųjų kintamųjų aibė; j{i, i+1, .., m} : atitinkama būsena.

(∃) $\dfrac{\exists x F(x) : q_i}{F(a) : q_i,}$ $\qquad$ $\dfrac{\exists x F(x) : [q_i, q_m]}{F(a) : [q_i, q_m], \;\big|\; F(a) \; [q_i, q'], \; q_i <= q' <= q_m, \; \exists x F(x) : [q'+1, q_m], \; q' <= q'+1 <= q_m,}$

čia:

a: naujas kintamasis,

q': naujas kintamasis, nesutinkamas šakoje,

Būsenų kolizijos taisyklė:

(Bkt)
$$\frac{q_i <= q',\quad q_i <= q''}{q_i <= q',\ q' <= q'' \qquad q_i <= q'',\ q'' <= q'}$$

5.1.2. Laisvojo kintamojo taikymas

Teiginys: Kvantoriniame baigtinės teiginių logikos skaičiavimo variante taisyklės ($\forall$) taikymui galimi apribojimai: α(j) bet kuris laisvasis kintamasis sutinkamas šakoje aukščiau.

Įrodymas:

1 Tarkime, turime išvedimo šaką A, kurioje yra laisvasis kintamasis ß.

...

A (ß)

...

Išvedimo šakoje sutinkamas ($\forall$):
$$\frac{\forall x F(x) \quad :q_i}{F(\alpha(j)) \quad :q_i,}$$
$$F(ß) \quad :q_i,$$

α (j) nėra šakoje aukščiau, todėl yra pakeičiamas į ß, t.y. į bet kurį laisvą kintamąjį sutinkamą šakoje aukščiau.

Po pakeitimų gauname išvedimą, atvirą ir pilną šaką, vadinasi sprendimo šakoje sutikus bendrumo kvantorių, vietoj α (j), galima naudoti bet kurį laisvą kintamąjį, sutinkamą šakoje aukščiau.

2. Tarkime turime išvedimą A:

...

F(α (j))

A ...

¬F(α (j))

Imkime naują išvedimą B:

...

F(α (j))

F(ß)

A ...

¬F(ß)

Naujame išvedime vietoj α (j)), naudojame laisvąjį kintamąjį ß sutinkamą šakoje aukščiau. Matome, kad aksiomos išvedime A ir naujame išvedime B lieka aksiomomis.

5.2. PLBLS skaičiavimo pavyzdžiai

5.2.1. Pavyzdys, kai formulė turi modelį

Užduotis: Nagrinėjama formulė $\Box(\Diamond\neg F(y, a, a, b) \& \exists y \forall x F(y, x, a, b))$, $k = 3$.

Sprendimas:

$\Box(\Diamond\neg F(y, a, a, b) \& \exists y \forall x F(y, x, a, b)) : q_0$

$(\Diamond\neg F(y, a, a, b) \& \exists y \forall x F(y, x, a, b)) : [q_0, q_m]$

$(\Diamond\neg F(y, a, a, b) : [q_0, q_m]$

$\neg F(y, a, a, b) : [q', q_m]$

$q_0 <= q';$

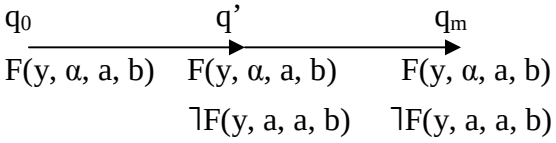
$\exists y \forall x F(y, x, a, b)) : [q_0, q_m]$

$\forall x F(z, x, a, b)) : [q_0, q_m]$

$F(z, \alpha, a, b) : [q_0, q_m]$

α : metakintamasis, t.y., bet kuris laisvasis kintamasis sutinkamas šakoje aukščiau.

Rezultatas: Nagrinėjama formulė F turi modelį(k neviršytas), kai :



5.2.2. Pavyzdys, kai formulė neturi modelio

Užduotis: Nagrinėjama formulė : (◇ □F(x, y)& □ ∀ y F(x, y) & ◇F(x, y)), k =3.

Sprendimas:

(◇ ¬F(x, y)& □ ∀ y F(x, y) & ◇F(x, y)) :q_0

◇ ¬F(x, y) :q_0

¬F(x, y) :q’

q_0<=q’

□ ∀ y F(x, y) :q_0

∀ y F(x, y) :[q_0, q_m]

F(z, y) :[q_0, q_m]

◇F(x, y)) :q_0

F(x, y)) :q’’

q_0<=q’’

Išvedime susiduriama su keliais tais pačiais ◇ laiko operatoriais, todėl reikia taikyti būsenų koalizijos taisyklę, tam, kad būtų nustatytas būsenų eiliškumas.

(Bkt)

q_0<=q’

q_0<=q’’

q_0<=q’ q_0<=q’’

q’<=q’’ q’’<=q’

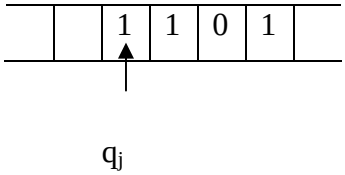
Rezultatas: Nagrinėjamai formulei F neegzistuoja modelis, nes k yra viršytas:

k=3, o gautos 4 būsenos {q_0, q’, q’’, q_m}.

6. Turingo mašinų darbas virš baigtinės laiko struktūros

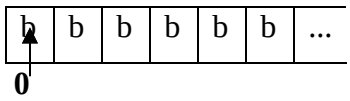
6.1. Turingo mašina

Turingo mašiną galime pavaizduoti schematiškai:



Nagrinėsime Turingo mašinas(žr.[Nor04]) su vienpuse begaline juosta į dešinę. Juostos ląsteles numeruojame: pirmajai priskiriame 0, o toliau iš eilės, t.y. 1, 2, 3, 4,...

Jeigu mašina neturi pradinių duomenų, ji pradeda darbą nulinės būsenos, jos skaitymo galvutė yra ties nuline ląstele ir visose ląstelėse yra simboliai b. Tokios mašinos iš pradžių užrašo pradinius duomenis, o paskui atlieka skaičiavimus.



Turingo mašinos darbas yra diskretus. Jo žingsnius numeruojame skaičiais 0, 1, 2,...

6.2. Predikatai

Aprašome predikatus, kurių apibrėžimų aibė yra natūraliųjų skaičių aibė:

- S(s,k) :q_0, yra teisinga tada ir tik tada, kai momentu q_0 ląstelėje s yra simbolis k,
- B(q) :q_0, yra teisinga tada ir tik tada, kai momentu q_0 mašina yra būsenoje q,
- P(s) :q_0, yra teisinga tada ir tik tada, kai momentu q_0 skaitymo galvutė yra ties ląstele su numeriu s .

6.3. Turingo mašinos darbo vaizdavimas formulėmis

Pradinė situacija, kai mašina neturi pradinių duomenų, aprašoma formule:

$\forall uS(u, b)\& B(0)\& P(0)$:0

Egzistavimas ir vienatingumas nusakomi formule:

$\Box \exists x \exists y \exists z \exists u(B(x)\&P(y)\&S(z,u)\& \forall v(B(v)\rightarrow(x=v)))\& \forall v(P(v)\rightarrow(y=v))\&$
$\forall v(S(z,v)\rightarrow(u=v))))$

Šia formule tvirtinama, kad kiekvienoje būsenoje q_0, mašina yra kurios nors vienintelės būsenos x, skaitymo galvutė yra ties vienintele ląstele y ir ląstelėje įrašytas vienintelis abėcėlės elementas.

Modeliuojame aprašytosios mašinos perėjimus formulėmis. Tarkime yra *r* perėjimų ir j-tasis yra pavidalo $\delta(q, m) = (q', m', D)$ (D- dešinėn).

Perėjimų formulė:

PR(j): $\Box \forall s((B(q)\&P(s)\&S(s, m)) \rightarrow \bigcirc (B(1)\& P(s+1)\& S(s, m)\& \forall u \forall v((\neg (u=s)\& \Box(S(u, v)) \rightarrow \bigcirc S(u, v)))))$

Panašiai priskiriamos formulės ir likusiems (r-1) perėjimų.

ST: $\Box \forall s((B(1)\&P(s)\&S(s, m)) \rightarrow \bigcirc (B(1)\&P(s)\&S(s, m)))$

Tikslas: $\Diamond B(1)$

7. Planavimo uždaviniai

7.1. Planavimo uždavinio apibrėžimas

Planavimo uždavinys susideda iš(žr. [MCL02], [SGLL06]):

1. $F_1, F_2,\ldots,F_n$, - pradinės sekos formulių (**pradiniai duomenys**),
2. $\Diamond G$ – **tikslo formulės** – „kada nors G“,
3. **įvykių aibės**, kuriuos agentas gali įvykdyti: $\Box(V_1 \rightarrow \bigcirc D_0)$, $\Box(V_2 \rightarrow \bigcirc D_2)$,….
4. **rezultato** ($F_1, F_2,\ldots,F_n, V_1, V_{2,\ldots}$) - jei įvykdoma: įvykių seka, kurią įvykdžius, nuo pradinės būsenos pasiekiamas tikslas, jei neįvykdoma: nesėkmė.

Sprendžiant planavimo uždavinius yra svarbiausia, kad agentas:

1. turėtų baigtinius duomenis,
2. turėtų galimybę valdyti visą pasaulį.

Baigtiniai duomenys – sąlygų aibė, pagal kurią galima nustatyti formulės įvykdomumą. Turi būti apibrėžtos visos galimos įvykių baigtys, tačiau sudarinėjant tokį modelį, sąlygų gali gautis be galo daug. Nagrinėti formulę pagal tokį modelį gali būti labai sudėtinga, ar netgi neįmanoma.

Dauguma planavimo uždavinių sprendimo modelių remiasi tuo, kad agentas turi baigtinius duomenis(žr. [ML02]). Tačiau yra tokių uždavinių, kuriuose informacija nėra tiksli.

7.2. Sąlyginis ir kontingentinis modeliai

Sąlyginiame modelyje (žr. [MCL02]) yra svarbu uždavinio požymiai, t.y. kokiomis esant sąlygomis, kaip elgsis modelis. Sąlygos turi būti apibrėžtos teisingumo principu: jei sąlyga teisinga – atliekami vieni veiksmai, jei sąlyga klaidinga – atliekami kiti veiksmai.

Taip pat yra naudojamas ir **kontingentinis modelis**. Šis modelis yra pora <planas, sąlygos>, t.y. esant skirtingiems planams sutikus tas pačias sąlygas gali būti atliekami skirtingi veiksmai, ir analogiškai atvirkščiai.

7.3. Planavimo uždavinių sprendimas naudojant laiko logiką.

Buvo išbandyta daug modelių ir logikų planavimo uždaviniams spręsti. Vienas jų yra **laiko logika.**

Ši logika yra naudinga (žr. [MLOP01]) todėl, kad yra išraiškinga ir galime apibrėžti ne tik veiksmus, tačiau ir juos priskirti atitinkamiems laiko momentams. Suformulavus uždavinio pradines sąlygas tiesinio laiko logikos išraiškomis ir pritaikius planavimo sprendimo taisykles būtų galima pasiekti tikslą ir nustatyti kelią iki jo.

Sprendžiant uždavinį pirmiausia reikia išsiaiškinti kokius duomenis turime ir kaip jie gali būti valdomi; po to tai išreikšti laiko logika (žr. [MLOB02], [MLOP01]). Tada galime naudoti paieškos modelį **(**žr. 9. Paieškos modelis), t.y. galima sumažinti paieškos sritį, rasti kelią(nustatyti, kad jo nėra), arba rasti optimalesnį kelią nei turime.

Uždavinio aprašymui naudosime logines operacijas(žr. 4.1.Tradicinės loginės operacijos): ¬, &, V , taip pat laiko operatorius(žr. 4.2. Laiko operatoriai): □, ◇, ○.

7.3.1. Planavimo uždavinio pavyzdys

Pradinė būsena: stovime prie kambario durų,

Tikslas ◇G- įeitį į kambarį.

Sąlygų aibė ***{ kambario durys atidarytos,***
durys uždarytos, durys užrakintos,
durys atrakintos,
turime raktą,
neturime rakto,....}

Veiksmų aibė: ***{ (įeiti → durys atidarytos)***
(atidaryti duris → durys uždarytos & durys neužrakintos)
(atrakinti → durys užrakintos)
(gauti raktą → durys užrakintos & neturime rakto)
..... }

Atsakymas galėtų būti : **atrakinti, atidaryti duris, įeiti.**

8. Paieškos modelis(žr. [MLOP02], [CM98])

Planavimo uždavinys gali būti išreikštas tiesinio laiko logikos formulėmis: planavimas suvedamas į modelio paiešką.

Laiko logika planavime gali būti taikoma keliais būdais: planavimas **išvadomis** remiantis laiko intervalais, planavimas taikant **patikrinimo modelį**, planavimas remiantis **tiesinio laiko logika**. Laiko logika naudota užkoduoti kontrolinę informaciją ir pagerinti planavimo algoritmą.

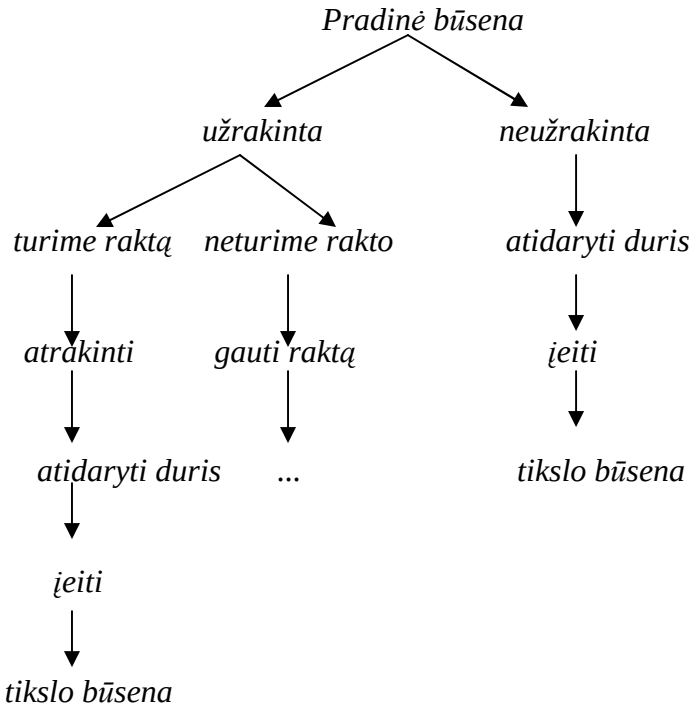
Loginis planavimo kodavimas numato formalią planavimo kalbos semantiką ir realizuoja naudingą įrankį siekiant išvengti dubliavimosi ir prieštaravimų.

Nagrinėjamos sistemos pagrindas yra paieškos modelio mašina paremta BTLL formulėmis. Formulių pagalba mašina gali būti efektyviai įgyvendinta.

Nepaisant TLL sudėtingumo, šis planavimo modelis teikia didelę naudą skaičiavimo galimybėms.

Lygiagretus ir padalijimo modeliai plačiai išnagrinėti. Analizuojant BTLL formules nustatyta skirtingos galimos **lygiagretumo** formos, kurios gali būti kontroliuojamos skirtingų agentų. Taikant **dalijimo** modelį: naudojant TLL formules užduotis gali būti suskaidoma į kelis homogeniniu procesus. Kiekvienoje laiko būsenoje atliekamas paralelizmas: kad išskaidyti formulių sekas reikia suskirstyti į k atskirus poaibius (kur k yra procesų skaičius). Formulių skaidymas yra lygiagretus, o k rezultatai yra atitinkamai suderinti.

8.1. Paieškos medžio fragmentas(8.3.1.pavyzdžiui)



8.2. Sąlyginio skaičiavimo taisyklių žymėjimai

Planavimo uždavinius taip pat galime spręsti naudojantis sąlyginio skaičiavimo taisyklėmis.

Taisyklėse raidėmis F ir G žymėsime teiginių logikos formules, taip pat nurodomi tam tikri apribojimai būsenoms:

q_i – būsena,

q’ ir q” – naujos būsenos, nesutinkamos šakoje,

$[q_i, q_m]$ – struktūros intervalai,

q_r – einamoji būsena,

q_m – didžiausia struktūros būsena,

$s(q_i)$ – sekantis, elementas po q_i,

C – sąlyga,

K – atitinkama šaka.

Gali būti nurodyta, kad apribojimai išvedami iš aukščiau šakoje sutinkamų apribojimų . Juos žymėsime Δ. Pradinės formulės atveju apribojimų aibė tuščia. Sakysime, kad

apribojimų aibė prieštaringa, jei iš Δ išvedama nelygybė $q_i < q_i$, kur q_i kuri nors būsena. Sakysime, kad aibėje Δ visiška tvarka, jei kokios bebūtų būsenos q’ ir q”, iš Δ išvedama arba q’ = q”, arba q’ < q” arba q”<q’.

Nagrinėsime reiškinius pavidalo F : q_i, čia F yra formulė, kuriai nustatinėsime ar egzistuoja jai modelis. Pradiniais duomenimis gali būti ir ne viena formulė, o kuri nors baigtinė aibė $\{F_1,..,F_s\}$.

6 apibrėžimas: Ciklinė(periodinė) šaka:

Šaka(K) = {k+p+i: l | k+i: l; l priklauso šaka(K), i priklauso N}, k- būsenos indeksas, nuo kurios prasideda ciklas, p – periodas,

8.3. Sąlyginio skaičiavimo taisyklės(žr[ML02]):

Pradinė formulė: **Jei C, tai H: q_0,** H: neiginys įkeltas; C= tuščia aibė.

(&) $\dfrac{\text{Jei C, tai F\&G :}q_i}{\begin{array}{l}\text{Jei C tai, F :}q_i\\ \text{Jei C tai, G :}q_i\end{array}}$ $\qquad$ $\dfrac{\text{F\&G :}[q_i, q_m]}{\begin{array}{l}\text{Jei C tai, F :}[q_i,q_m]\\ \text{Jei C tai, G :}[q_i,q_m]\end{array}}$

(∨) $\dfrac{\text{Jei C, tai F}\vee\text{ G :}q_i}{\text{Jei C, tai F :}q_i \qquad \text{Jei C, tai G :}q_i}$

$$\frac{F \vee G :[q_i, q_m]}{\begin{array}{ll}\text{Jei C, tai F :}q_i & \text{Jei C, tai G :}q_i\\ \text{Jei C, tai }\circ\text{F: }[q_i,q_m] & \text{Jei C, tai }\circ\text{G: }[q_i,q_m]\end{array}}$$

(□) $\dfrac{\text{Jei C, tai }\Box\text{F : }q_i}{\text{Jei C, tai F :}[q_i,q_m]}$ $\qquad$ $\dfrac{\text{Jei C, tai }\Box\text{F :}[q_i, q_m]}{\begin{array}{l}\text{Jei C, tai F : }q_i\\ \text{Jei C, tai }\circ\Box\text{ F :}[q_i,q_m]\end{array}}$

(◇) $\dfrac{\text{Jei C, tai }\Diamond\text{F :q}}{\text{Jei C, tai F :}q_i, \qquad \text{Jei C U }\{\neg\text{F :}q_i\}\text{, tai }\circ\Diamond\text{F :}q_i,}$

$$\frac{\text{Jei C, tai }\Diamond\text{ F : }[q_i,q_m]}{\text{Jei C, tai F : }q_m,}$$

(○) .

. (K)

.

$$\frac{\text{Jei C, tai }\circ\text{F :}q_i}{\text{Jei C' F : s(qi)}}$$

$$\frac{\text{Jei C, tai }\circ\text{F :}[q_i, q_m]}{\perp}$$

Kur C'= 0, jei h(K, q_i) ├ C, priešingu atveju C' = C.

8.3.1. Sąlyginio skaičiavimo pavyzdys:

Tikslas: ◇I(įeiti),

Sąlygų aibė: {A – kambario durys atidarytos,

¬A – kambario darys uždarytos,
U – kambario durys užrakintos,

¬U – kambario durys neužrakintos,
R – turime raktą,

¬R – neturime rakto}.

Veiksmų aibė: {I → A,

D(atidaryti duris) → ¬A&¬U,
T(atrakinti) → U&R}.

Pradiniai duomenys: {¬A, U, R}.

Visas galimas sąlygas išreiškiame formule:

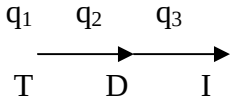
F = (jei A, tai I)&(jei (¬A&¬U), tai D&○I)&(jei (U&R), tai T&○(D&○I))

Sprendimas:

(jei A, tai I)&(jei (¬A&¬U), tai D&○I)&(jei (U&R), tai T&○(D&○I)) :q_1
(jei (U&R), tai T&○(D&○I)) :q_1
T :q_1
○(D&○I)) :q_1
D&○I :q_2
D :q_2

○I :q_2
○I :q_3

Rezultatas:



Veiksmų seka: atrakinti, atidaryti, įeiti.

9. Iteratyvus planavimas(žr. [HL07])

Paprasčiausias yra sąlyginis modelis, kur planą galime pavaizduoti kaip medį. Tačiau taip pat yra alternatyva - **iteratyvus** planavimas, kur planą galime pavaizduoti grafu su ciklais.

9.1. Problemos

Naudojant iteratyvų planavimą iškyla keletas problemų:

1. **proceso užbaigtumas**: kadangi egzistuoja ciklai, tai galima patekti į amžiną ciklą.

2. **programos korektiškumas:** vykdant ciklus galima gauti nekorektiškas duotam uždaviniui reikšmes.

3. **sunku realizuoti**: net ir trumpą iteratyvią programą gali buti sunku išreikšti ir įgyvendinti.

9.2. Iteratyvaus planavimo pavyzdys

Iteratyvus planavimas naudingas, kai keletą kartų reikia kartoti tą pačią veiksmų seką. Uždavinio su ciklais pavyzdys:

Pradinė būsena: stovime prie kambario durų,

Tikslas: ◇G- įeitį į kambarį(gali būti kelios durys).

Sąlygų aibė ***{ kambario durys atidarytos,***

durys uždarytos, durys užrakintos,

durys atrakintos,

turime raktą,

neturime rakto,

....}

Veiksmų aibė: ***{ (įeiti → durys atidarytos)***

(atidaryti duris → durys uždarytos & durys neužrakintos)

(atrakinti → durys užrakintos)

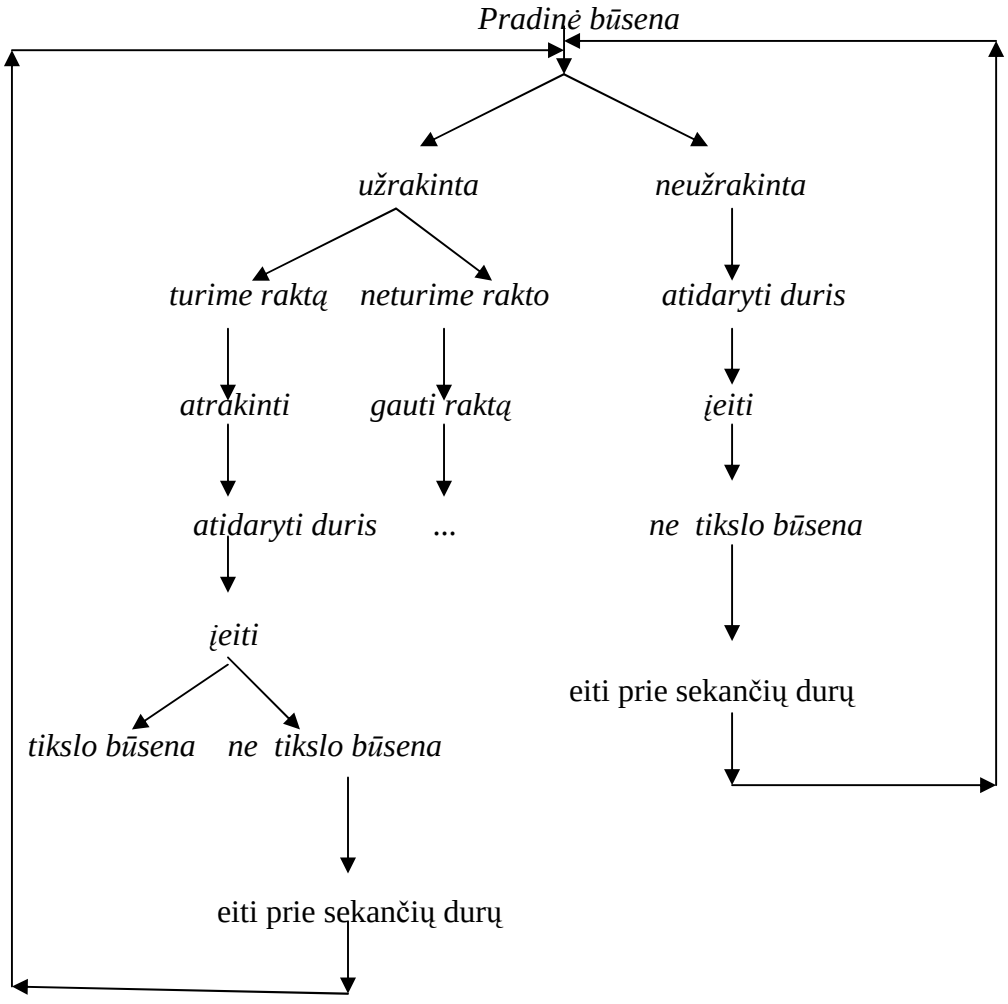
(gauti aktą → durys užrakintos & neturime rakto)

..... }

Atsakymas galėtų būti : ***atrakinti, atidaryti duris, įeiti, prieiti prie kitų durų, atidaryti duris, įeiti)***

9.3. Grafo su ciklais fragmentas

Šį uždavinį galime pavaizduoti ir grafu su ciklas:



10. Planavimas įvertinant sekas ir veiksmus

Svarbu atkreipti dėmesį į planavimo modelį(žr. [MLOP01]), kuriame skaičiuojamos galimybės išreikšti veiksmus ir sekas ne vien tik su teisingumo reikšmėmis{teisinga, klaidinga}. Remiantis šiuo modeliu, planavimo problema yra apibrėžiama naudojant ir tiesos laipsnius ir klasikines teisingumo reikšmes. Be to gali būti apibrėžtos klasės turinčios skirtingą taikomumo laipsnį.

Klasikinis modelis, kai veiksmai ir sekos išreiškiamos teisingumo reikšmėmis, yra ribotas, nes dažnai reikšmė nėra visiškai teisinga, arba visiškai klaidinga. Yra tarpinės teisingumo reikšmės arba **tiesos laipsniai**. Šiame modelyje tiesos laipsniai apibrėžia mūsų žinias, būsenos neapibrėžtumą arba įvykio sėkmės tikimybę.

Naudojantis šiuo modeliu gali būti apibrėžti veiksmai turintys skirtingus taikomumo laipsnius (t.y. veiksmai turintys reguliuojamą intensyvumą, o tai gali paveikti sekų proporcijas, kaip jos gali būti taikomos). Šiuo būdu galima lengvai parodyti veiksmų efektyvumą.

Žinoma, nustatyti tiesos laipsnius yra daug būdų ir juos galime pasirinkti išnagrinėjus uždavinio sąlygas bei tikslą.

10.1. Teisingumo reikšmių žymėjimai

Pats paprasčiausias žymėjimo būdas yra:

1. teisingumo reikšmėmis:

sekoms: 0-teisinga,

1- klaidinga;

veiksmams - 0-veiksmas šioje situacijoje visiškai netinkamas,

1- efektyviai tinka.

2. laipsninėmis teisingumo reikšmėmis:

sekoms: reikšmių sritis yra visi realieji skaičiai iš [0;1], nurodomas sekos tinkamumo laipsnis

veiksmams: reikšmių sritis yra visi realieji skaičiai iš [0;1], nurodoma veiksmo tinkamumo laipsnis

Sprendžiant problemą yra galimybė turėti objektyvią funkciją, kuri yra tiesinė veiksmo tinkamumo laipsnio funkcija. Ja galima rasti mažiausią arba didžiausią laipsnį turintį spendimų planą. Tai aktualu ieškant kelio iki tam tikro taško, arba norint sumažinti sąnaudų kainą.

Sistema priima laipsnišką planavimo problemą ir jei egzistuoja sprendimas, tai grąžina laipsnišką sprendimo planą P. Priešingu atveju grąžina rezultatą: „Sprendimo nėra“.

10.2. Sistemos architektūra

Architektūra susideda iš trijų pagrindinių dalių, tai : plano plėtėjas, sistemos konstruktorius, sprendėjas.

Plano plėtėjas randa galimus kelius($A_1(x_1)$, $A_2(x_2)$, $A_3(x_3)$, ..., $A_m(x_m)$). Čia taikomumo ir sekos laipsniai yra nežinomi. Yra žinomi tik x_i kintamieji.

Tada **sistemos konstruktorius** suskaičiuoja pasaulių kitimą naudojant veiksmų laipsnius(tikslo būsena priklauso nuo veiksmų laipsnių), taip pat pritaiko struktūrą sprendėjui, kad būtų galima patikrinti korektiškumą. Ši struktūra perduodama sprendėjui. Sistemos konstruktorius vykdymo fazėje kai kurios preliminarių veiksmų sąlygos tiesiškai patikrinamos ar jos neveda į nesėkmę, ir ar nereikia ieškoti kito galimo plano/kelio. Jei randamas veiksmas su sąlyga sukeliantis nesėkmę, tai toliau planas konstruojamas(plečiamas) nuo šio veiksmo. Galiausiai **sprendėjas** įvertina generuotos problemos sprendimą. Jei sprendimas egzistuoja, tai grąžinama reikšmių seka $\vec{g}=(g_1,...,g_m)$, o planas $P= (A_1(g_1), A_2(g_2), A_3(g_3), ..., A_m(g_m))$ yra laipsniškas sprendimo planas. Jei sprendimas neegzistuoja, tai preliminarus planas yra plečiamas toliau, generuojama nauja problema.

11. Planavimo uždavinių euristikos

Planavimo uždavinių apibrėžime teigiama, kad agentas būtinai turi žinoti pradinius duomenis ir įvykių aibę, kuriuos agentas gali vykdyti. Sprendžiant realaus pasaulio uždavinius kartais agentas ne viską žino.

11.1. Mokymo modelis

Naudojant **mokymo modelį**(žr. [LST01]) stebimas objekto elgesys: nagrinėjama kiekviena situacija ir kokie veiksmai buvo daromi. Taip sekant galima toliau naudotis **sąlygų modeliu,** sudaryti sąlygų aibę ir apibrėžti veiksmus, kuries yra vykdomi esant atitinkamoms sąlygoms.

Mokymo modelis susideda iš:

1. pradiniai duomenys,
2. duomenys gauti mokymo metu:
    a) objekto identifikavimas,
    b) pradinių duomenų panaudojimas,
    c) mokymas – nėra vieno algoritmo efektyviai apmokyti agentą. Kiekvienam uždaviniui keliami skirtingi tikslai ir gali būti naudojami skirtingi stebėjimo ir apmokymo modeliai,
    d) įgytos žinios.

11.2. Numatomumo modelis

Numatomumo modelis naudojamas sprendžiant uždavinius su nepilna informacija. Modelio esmė: sudaryti(generuoti) veiksmų seką, kurią įvykdžius pasiekiamas tikslas. Šis modelis gali būti naudojamas ir su sąlyginius planavimu, kur planą galime pavaizduoti kaip paieškos medį. Taip pat gali būti taikomas ir iteratyviame planavime, kur planas vaizduojamas grafu su ciklai. Abu pastarieji metodai apima planavimo problemą, kai yra uždaviniai su nepilna informacija, t.y. pradiniai duomenys nėra pilni, o reikiama informacija yra gaunama(išmokstama) sprendimo metu. Šis modelis turi numatyti, kaip turi būti elgiamasi kiekvienu atveju, priklausomai nuo to kokia informacija bus įgyjama sprendimo metu.

PVZ.: Lakmuso popierius: jei yra rūgštinis skystis, popierius nusidažo raudonai, priešingu atveju – mėlynai, neutralioje aplinkoje jis nekeičia spalvos. To mes sprendimo pradžioje galime ir nežinoti, tačiau jei veiksmai kartojasi, mes galime numatyti, kad taip yra visada ir tiesiog įsiminti, kad kaip taisyklę.

Gali būti ir taip , kad naujosios taisyklės prieštaraus pradiniams duomenims (pvz.: buvo duota, kad lakmuso popierėlis rūgštiniame skystyje nusidažo žaliai). Ką tokiu atveju daryti? Planas turi numatyti iš anksto kas turi prioritetą: ar pradiniai duomenys ar suformuotos naujos taisyklės. Prioriteto nustatymui taip pat gali būti naudojami keli metodai. Galima prioritetą nustatyti tiesiog prieš pradedant spręsti uždavinį, arba sprendimo metu: (pvz.: atsižvelgiant į taisyklės patikimumą, t.y. gal būt 50 kartų pamerkiant popierių buvo gaunama ta pati spalva, kuri prieštarauja pradiniams duomenis, tai logiška būtų manyti, kad pradiniai duomenys yra klaidingi), tokiu atveju prioritetas skiriamas taisyklei.

Tačiau šis planas yra gana ribotas. Šis modelis netinka planams, kuriuose yra reikalingas ciklas (kai neaišku kiek pakartojimų jame). Įvedus kai kurias modifikacijas galima planavimą pritaikyti ir planams su ciklais, tačiau prarandamas jo tikslumas.

11.2.1. Numatomas planavimas

Numatomame planavime problema apibrėžiama:

ß = {B, A, T, b_0, G}, kur:

B – **numatomų veiksmų seka**,

A – **baigtinė veiksmų seka**,

T – **transakcijos ryšys**: T ≤ B×A×B, T(b, a, b‘) reiškia, kad b būsenoje atlikus veiksmą a bus sėkmė ir pateksime į būseną b‘. Šis reiškinys yra sėkmingas, jei atsiras bent vienas toks b‘. Reiškinys T turi būti baigtinis.

b_0 – **pradinė būsena,** b_0 B,

G – **būsenų seka**, kuriomis pasiekiamas tikslas. G ≤ B

Planavimo uždavinio sprendimas reiškia veiksmų sekos radimą(a_1, a_2, ..., a_n), kurią įvykdžius pasiekiamas tikslas. Svarbiausia yra bet kuriame žingsnyje žinoti sekantį veiksmą, kurį mes turime daryti.

Planas P yra ketvertas: P = {Q, q_0, Sa, Ss}, kur:
Q – **galimų būsenų seka**,
q_0 – **pradinė būsena**, kur b_0 □ Q,
Sa - **nurodo sekantį veiksmą**, Sa [Q×B→AU{stop}], kur Sa(q, b), žinant dabartinę būseną q, plano numatomą būseną b, gražina sekantį veiksmą.Su funkcija „stop“ turime patikrinti ar atlikę sekantį veiksmą bus pasiektas tikslas. Jei taip, tai iškviečiama Ss, kurios pagalba randame sekančią būseną(t.y. kurią pasieksime atlikę paskutinį veiksmą a) ir sprendimas sustabdomas.
Ss - **nurodo sekančią būseną**, Ss □ [Q×B×A→Q], kur Ss(q, b,a), žinodamas dabartinę būseną q, plano numatomą būseną b ir įvykdžius veiksmą a, gražina sekančią būseną.

Naudojant šį modelį būtina apsibrėžti, kad nebūtų generuojami neįmanomi veiksmai, t.y.:

jei Sa(q, b) = a, ir a ≠ {stop}, tai ∃b‘, toks kad T(b,a,b‘).

Apibrėžimas. Planas P = {Q, q_0, Sa, Ss}yra adekvatus numatomumo modelio planui ß = {B, A, T, b_0, G}, jei būsenų pora (q_0, b_0) R_p, kur R_p:

1 Jei Sa(q, b) = {stop} ir b□G, tai {q:b} R_p,
2 Jei Sa(q, b) = a, Sa(q, b, a) = q’, ir ∀b‘, toks kad T(b, a, b‘) atsiras {q‘:b‘} R_p, tai {q:b} R_p

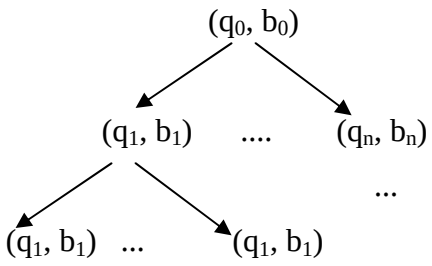
Pastaba: R_p - būsenų porų {q:b}seka, tokia, kad vykdant planą P, plano būsena q ir numatoma būsena b garantuoja, kad bus pasiekta tikslo būsena.

Paieškos medžio formavimas

Plano P = {Q, q_0, Sa, Ss}, kuris yra adekvatus numatomumo modelio planui ß = {B, A, T, b_0, G}, medžio formavimas:

1 medžio šaknis yra (q_0, b_0),
2 iš kiekvieno mazgo (q, b) išeina briauna į kitą mazgą(q’, b’), kur b‘: Sa(q, b) = a, Ss(q, b, a) = q‘, ir T(b, a, b‘).

Formuojamas medis:



Išvados

Klasikinė logika atskleidžia nagrinėjamų objektų ar reiškinių tik statinę būseną, todėl uždavinių nagrinėjimui yra labai svarbi baigtinė tiesinė laiko logika, nes ji logika atsižvelgia ir į laiko faktorių.

Remiantis straipsniais (žr. [CM97], [MSC97]) buvo suformuluotos tiesinei baigtinei logikai skaičiavimo BTLL, PLBLS taisyklės. Jomis tiksliai apibrėžti konkretūs galimi problemų atvejai, todėl taisyklų pagalba išanalizuoti norimus uždavinius jau nėra sudėtinga.

Taip pat buvo analizuojamas Turingo mašinų darbas(žr.[Nor04]) virš baigtinės laiko struktūros. Suformuluoti reikalingi predikatai, kurių pagalba aprošomos perėjimų formulės.

Planavimo uždaviniai yra plačiai nagrinėjama sritis, kurioje sprendimo metodų yra labai daug. Šiame darbe nagrinėtas planavimo uždavinių sprendimas taikant baigtinę tiesinio laiko logiką. Sprendžiant šiuo būdu atsižvelgta kaip galima analizuoti konkretų uždavinį, apibrėžti visas galimas būsenas, veiksmus ir išreikšti tai formule su laiko operatoriais.

Remiantis straipsniais (žr. [ML02], [MLOB00]) buvo suformuluotos sąlyginio skaičiavimo taisyklės baigtinei tiesinio laiko logikai. Jomis tiksliai apibrėžti konkretūs galimi planavimo problemų atvejai. Taisyklių pagalba jau nėra sudėtinga nagrinėti uždavinius ir nustatyti ar jam egzistuoja modelis.

Taip pat apžvelgta kokios yra planavimo uždavinių sprendimo euristikos, t.y. kokias galima įvesti modifikacijas, kad būtų galima dar efektyviau spręsti šiuos uždavinius: mokymasis sprendimo metu, tiesos laipsniai, numatomumo metodas ir t.t.

Šiame darbe, žinoma, nėra atskleisto visos laiko logikos galimybės, nes jų yra labai daug. Egzistuoja daug uždavinių, kuriuose būtina atsižvelgti į laiko faktorių, todėl ši logika yra ypatingai naudinga tokių problemų sprendimui.

Summary

The aim of the present paper was to go deeper into the logics of time. By the formula of the classical logic it is possible to analyze only those problems where the condition of the objects is static. However, there are such problems where the condition of the objects is dynamic, i.e., the meanings are not always the same, they change. That is why it is not possible to solve them by the formula of classical logic.

By referring to the articles (look [CM97], [MSC97]) calculation rules for the finite linear temporal logic as well as predicates' logic above the finite temporal logic (PLBLL) were formulated. Concrete possible problem cases were defined by those rules. Therefore, it is no longer difficult to analyze the problems wanted with the help of those rules.

Moreover, the formulated rules make it possible to analyze such problems where it is necessary to pay attention to the change of meanings at appropriate time moments. The main difference of this logic is: a more exact detailed analysis of the problem, additional tools of a problem analysis, i.e. new time operators are used.

The work of the Turing machine was analyzed (look [Nor04]) over finite temporal structure. Necessary predicates were formulated, with the help of which transit formula were described.

Also, this work aims at verifying the effective possibility of using Linear Time logic as a planning language. The main advantage of such a rich and expressive language is the possibility of encoding problem specific information that can be of help both in reducing the search space and finding a better plan.

This work aims at a different planning model that takes into account the possibility to express actions and fluents with non-boolean values. Action that can have different application degrees can be defined.

Literatūros sąrašas

[CM97] S.Cerrito, M.C. Mayer A Prefixed Tableau Calculus for Plan Generation in Linear Temporal logic, KB, 1997.05.

[MSC97] M.C. Mayer, S.Cerrito, A. Cesta Planning as model construction in linear temporal logic, KB, 1997.05.

[Nor04] S.Norgėla. Matematinė logika. TEV, Vilnius, 2004m.,

[Nor07] S.Norgėla. Logika ir dirbtinis intelektas. TEV, Vilnius, 235-250psl., 2007m.,

[CMP99] S.Cerrito, M.C. Mayer, S.Praud First Order Linear Temporal Logic over Finite Time Structures, KB, 1999m.

[ML02] M.C. Mayer, C. Limongelli Linear Time Logic, Conditioned Models and Planning with Incomplete Knowledge, 2002m.

[MLOP01] M. C. Mayer, C. Limongelli, A. Orlandi,V. Pogi, Planing under Unertainty in Linear Time Logic, 2001m.

[MLOB00] M. C. Mayer, C. Limongelli, A. Orlandi, G. Balestreli, A Planner Fully Based on Linear Time Logic 2000m.

[LST05] C. Limongelli, A. Sterbini, M, Temperini, Automated course congiguration based on automated planning: framework and first experiments, 1-4 psl., DIA university Roma, 2005m.

[HL07] H. J. Levesque Planning with Loops, University of Toronto, 2007m.

[ML02] M.C. Mayer, C. Limongelli Linear Time Logic, Conditioned Models and Planning with Incomplete Knowledge, 5-10 psl., 2002m

[MLOP04] M. C. Mayer, C. Limongelli, A. Orlandi, V. Poggioni, Towards a Parallel Search Engine for Planning Systems based on Linear Time Logic, DIA university Roma , 1-8psl. 2004m.

[ML98] M.C. Mayer, C. Limongelli, UsingTemporal Logic to Model and Solve Planning problems 1998m.

[MLOP04] M. C. Mayer, C. Limongelli, A. Orlandi, V. Poggioni, Planning with graded fluents and actions, 1-4 psl. DIA university Roma, 2004m.

[MLOP07] M. C. Mayer, C. Limongelli, A. Orlandi, V. Poggioni, Linear temporal logic as an executable semantics for planning languages, 2007m.

[CM98] S.Cerrito, M.C. Mayer, Bounded Model Search in Linear Temporal Logic and its Application to Planning, 1998m.

[SGLL06] S. Sardin, G. De Giacomo, Y. Lesperance, H. J. Levesque, On the Limtis of Planning over Belief States under Strict Uncertaintly, 1-9 psl, 2006 m.

PRIEDAS

1.1 Loginės operacijos

Žymėjimai: p, q, r – teiginiai, t – teisinga, k – klaidinga.

Neigimas(“ne p”) Konjunkcija(“p ir q”) Disjunkcija(“p arba q”)

p	**¬p**
t	k
k	t

p q	**p&q**
t t	t
t k	k
t t	k
t k	k

p q	**p∨q**
t t	t
t k	t
t t	t
t k	k

Implikacija(“iš p išplaukia q”)

p q	**p→q**
t t	t
t k	k
t t	t
t k	t

1.2 Ekvivalenčios formulės

1.2.1 Ekvivalenčios formulių poros

$p \rightarrow q \equiv \neg p \vee q$,

$\neg (p \& q) \equiv \neg p \vee \neg q$,

$\neg (p \vee q) \equiv \neg p \& \neg q$,

$p \leftrightarrow q \equiv (p \rightarrow q)(q \rightarrow p)$,

$(p \& q) \vee r \equiv (p \vee r) \& (q \vee r)$,

$(p \vee q) \& r \equiv (p \& r) \vee (q \& r)$.

1.2.2 Ekvivalenčių formulių pavyzdžiai

$p \& q \equiv q \& p$,

$p \& (p \& q) \equiv (p \& p) \& q$,

$(p \vee q) \equiv (q \vee p)$,

$p \vee (q \vee r) \equiv (p \vee q) \vee r$,

$\neg\neg p \equiv p$.

1.2.3 Ekvivalenčių formulių savybės

Jei $A \equiv B$, tai $B \equiv A$,

Jei $A \equiv B$ ir $B \equiv C$ tai ir $A \equiv C$,

Jei $A \equiv B$ tada ir tik tada, kai $\neg A \equiv \neg B$.

2.Tiesinės baigtinės logikos skaičiavimo taisyklės pagal [CM97] ir [MSC97]:

(&) $\dfrac{q: A \& B}{q: A \quad q: B}$ (∨) $\dfrac{q : A \vee B}{q: A \quad q: B}$

(□) $\frac{q:\ \square A\ ,\quad jei\ \Delta \vdash q <= q'}{q': A}$

(⬱) $\frac{q:\ \boxminus A\ ,\quad jei\ \Delta \vdash q <= q'}{q': A}$

(◇) $\frac{q:\ \diamond A\ ,}{q': A \parallel \{q <= q'\}}$

čia q’ žymi naują būseną;

(sinch) $\frac{\Delta}{\parallel \{ q= q'\} \parallel \{ q< q'\} \parallel \{ q'< q\}}$.

Pritaikę sinchronizavimo taisyklę naujos formulės negaunama,bet gauname tris naujas šakas, kuriose prie prijungiama atitinkamai q = q’, q< q’, q’< q. q bei q’ turi būti būsenos, sutinkamos nagrinėjamoje šakoje.Šios taisyklės tikslas yra įvesti visišką tvarką apribojimų aibėje.

3.Predikatų logika

3.1.apibrėžimas: n-viečiu predikatu aibėje A vadiname vienareikšmę n argumentų funkciją, kurios apibrėžimo sritis yra aibė A ir reikšmių aibė – {t, k}.

3.2. Kintamieji:

1) loginiai: $p_1,p_2,p_3,...,..$
2) predikatiniai: $P_i(x_1,x_2,...x_n),...,$
3) individiniai: x,y,z,...,