Liet. matem. rink., **44**, *spec. nr.*, 2004, 351–356

Dviejų žinių bazių vidinių ir situacinių skirtumų nustatymas

Laima PALIULIONIENĖ (MII)
el. paštas: laipal@ktl.mii.lt

Reziumė. Žinių bazių palyginimo uždavinys iškyla įvairiose dalykinėse srityse, kai analizuojami skirtumai tarp įvairių autorių sukurtų tos pačios paskirties žinių bazių arba skirtumai tarp įvairių žinių bazių versijų, atsiradusių dėl jų pakeitimų. Straipsnio tikslas yra pateikti žinių bazių automatizuoto palyginimo metodą, naudojantį tiesioginį išvedimą su modelių generavimu, bei pasiūlyti šio metodo realizavimo būdą. Eksperimentams pasiūlyta naudoti ekspertinių sistemų kūrimo paketą Jess, jo kalba modeliuojant žinių vaizdavimo formalizmo F-logikos konstrukcijas.

Raktiniai žodžiai: žinių vaizdavimas, išvedimas žinių bazėse, modelių generavimas, žinių bazių palyginimas.

Įvadas

Žinių bazių palyginimo uždavinys iškyla dviem pagrindiniais atvejais: 1) kai nagrinėjamos skirtingos žinių bazės versijos, atsiradusios dėl jos keitimo ir 2) kai nagrinėjamos skirtingų autorių sukurtos tos pačios paskirties žinių bazės. Šis uždavinys gali iškilti įvairiose dalykinėse srityse. Ankstesniuose autoriaus darbuose [1, 2, 3] jis buvo nagrinėtas tiriant išvedimą teisinių žinių bazėse, pasiūlytas jo sprendimo metodas. Šis metodas yra bendro pobūdžio ir tinka įvairioms dalykinėms sritims. Pažymėtina, kad žinių bazių automatizuotas palyginimas iki šiol nėra išnagrinėtas taisyklinių žinių bazių teorijoje. Pavyzdžiui, [4] ir [5] nagrinėjamas žinių bazių ekvivalentiškumas, tačiau tikslas yra ne išryškinti skirtumus, bet suprastinti žinių bazę ekvivalentiškomis transformacijomis; be to, apsiribojama teiginių logika.

Straipsnyje pateiktas žinių bazių automatizuoto palyginimo metodas, naudojantis tiesioginį išvedimą su modelių generavimu, bei pasiūlytas šio metodo realizavimo būdas. Žinioms vaizduoti pasirinkta F-logika. Šis formalizmas jungia objektinį žinių vaizdavimo būdą su loginio išvedimo mechanizmu, grindžiamu rezoliucija, tačiau F-logikos programinėse priemonėse nerealizuotas tiesioginis išvedimas. Todėl eksperimentams su tiesioginiu išvedimu panaudota kita priemonė – ekspertinių sistemų kūrimo paketas Jess. Straipsnyje palygintos F-logikos ir Jess kalbų konstrukcijos, aptarti kai kurie jų privalumai bei trūkumai siūlomo metodo realizavimo kontekste.

Žinių bazių palyginimo metodas

Žinių bazių palyginimą apibrėšime kaip procesą, kurio metu nustatomos situacijos (faktų aibės), kurių pasekmės, išvestos dviejose nagrinėjamose žinių bazėse, yra

skirtingos. Toks palyginimas yra prasmingas tik panašiose žinių bazėse, turinčiose bendrą ontologiją.

Mūsų tyrime žinioms vaizduoti naudojamas freimų logikos formalizmas F-logika [6]. Šis formalizmas jungia objektinį žinių vaizdavimo būdą su loginio išvedimo mechanizmu, grindžiamu rezoliucija. Nauji faktai išvedami naudojant taisykles, kurių forma yra *antraštė←kūnas*, kur *antraštė* (taisyklės konsekventas) yra F-molekulė, *kūnas* (taisyklės antecedentas) yra F-molekulių ir jų neiginių konjunkcija. F-molekulėse naudojami klasių, objektų ir atributų vardai, atributų reikšmės, predikatų simboliai. F-molekulėse galima išskirti primityvias sudėtines dalis, vadinamas atomais. F-molekulės yra atomų konjunkcijos. Pavyzdžiui, F-molekulė *darbuotojas* :: *asmuo* [*pavardė⇒string*, *sodra_nr⇒string*] susideda iš atomų *darbuotojas* :: *asmuo*, *darbuotojas* [*pavardė⇒string*] ir *darbuotojas* [*sodra_nr⇒string*]. Be žinių bazės taisyklių nauji faktai išvedami ir pagal F-logikoje įmontuotas taisykles, pavyzdžiui, tam tikros atributų reikšmės gali būti priskiriamos pagal nutylėjimą.

Kad galima būtų palyginti žinių bazių išvadas iš situacijų, F-logiką siūloma išplėsti tiesioginiu išvedimu naudojant modelio generavimą. [1] aprašytas modelio generavimo algoritmas, pagal kurį modelis generuojamas pagal taisyklių grandines pradedant nuo faktų (t.y., nuo taisyklių be konsekvento). Naudojami trys modaliniai operatoriai: vienas operatorius susijęs su neiginiais taisyklių antecedentuose, du operatoriai – su atributų reikšmių perdavimu pagal nutylėjimą ir su išvedimu iš faktų su tokiomis reikšmėmis. Šie modaliniai operatoriai reikalingi todėl, kad neiginiai ir reikšmių paveldėjimas padaro išvedimą nemonotoniniu, nes nauji faktai gali paneigti anksčiau išvestus faktus. Modaliniai operatoriai leidžia nusakyti tam tikras prielaidas apie faktų buvimą, kurios tolesnio išvedimo metu gali būti patvirtintos arba paneigtos. Taip galima gauti kelis modelius [7], iš kurių pabaigoje atmetami netinkami, o patys modaliniai operatoriai galutiniuose modeliuose pašalinami. Tai, kad iš žinių bazės KB galima sugeneruoti modelį N, žymėsime $KB \vdash_M N$.

Nagrinėjamas žinių bazes ir situacijas apribosime kai kuriomis sąlygomis, kurios tinka daugumoje dalykinių sričių: 1) visi kintamieji taisyklių konsekventuose ir antecedentų neigiamose dalyse yra surišti, t.y., naudojami teigiamoje antecedento dalyje, 2) modelis yra tiksliai vienas, 3) modelis yra baigtinis.

Formaliai apibrėžkime, ką mes vadinsime skirtumu tarp žinių bazių.

1-*oji apibrėžtis.* Tarp žinių bazių KB_1 ir KB_2 yra vidinis skirtumas, jei $KB_1 \vdash_M R_1$, $KB_2 \vdash_M R_2$, ir rezultatų aibės R_1 ir R_2 nesutampa. Žymėsime taip:
$Diff\,Int(KB_1, KB_2) = (R_1 \cup R_2) \backslash (R_1 \cap R_2)$ – vidinio skirtumo aibė.

2-*oji apibrėžtis.* Tegu S yra faktų aibė, aprašanti situaciją. Žinių bazės KB_1 ir KB_2 yra *skirtingos situacijos S atžvilgiu*, jei $KB_1 \cup S \vdash_M R_1$, $KB_2 \cup S \vdash_M R_2$, ir rezultatų aibės R_1 ir R_2 nesutampa. Žymėsime taip:
$Diff(S, KB_1, KB_2) = (R_1 \cup R_2) \backslash ((R_1 \cap R_2) \cup Diff\,Int(KB_1, KB_2))$ – skirtumo aibė (t.y., į skirtumo aibę mes neįtraukiame faktų iš vidinio skirtumo aibės).

3-*oji apibrėžtis.* Žinių bazės yra *skirtingos*, jei tarp jų yra vidinis skirtumas arba jos yra skirtingos kurios nors situacijos atžvilgiu.

Galimų situacijų gali būti labai daug, galbūt net begalinė aibė, todėl tiesiogiai visų jų patikrinti neįmanoma. [1, 2, 3] parodyta, kad vietoj visų įmanomų

situacijų tikrinimo pakanka patikrinti situacijas, tam tikru būdu sukonstruotas iš konkretizuotų taisyklių antecedentų. Tokias situacijas vadiname elementariosiomis testinėmis situacijomis. Toliau pateikta elementariosios konkretizacijos apibrėžtis, naudojama konstruojant šias situacijas.

4-oji apibrėžtis. Teigiamos (be neiginių) formulės *konkretizacija* vadinama *elementarioji*, jei keitinyje, kuriuo gauta ši konkretizacija, visi kintamieji keičiami į konstantas ir:

1) šiose konstantose nėra funkcijų simbolių,

2) konstantos nepriklauso nagrinėjamų žinių bazių Herbrand'o universumui, t.y., nenaudojamos žinių bazėse,

3) skirtingi kintamieji keičiami į skirtingas konstantas.

Pavyzdys. Tegu turime formulę $s(X, f(Y), x)$, kurioje X ir Y yra kintamieji (pagal F-logikos notaciją, kintamieji prasideda iš didžiosios raidės, o konstantos – iš mažosios). Jos elementarioji konkretizacija gali būti $s(x1, f(y), x)$. Čia kintamieji pakeisti į atitinkamas konstantas – kintamasis Y į konstantą y, o kintamasis X į konstantą $x1$, nes konstanta x jau naudojama žinių bazėje.

Horno žinių bazėse (t.y., tokiose, kurių taisyklėse nenaudojami neiginiai) elementariosios testinės situacijos yra taisyklių antecedentų elementariosios konkretizacijos. F-logikoje naudojamų klasių, objektų ir jų atributų konkretizacijos yra analogiškos predikatų konkretizacijoms. Žinių bazėse, išreikštose bendrosiomis loginėmis programomis (t.y., su neiginiais taisyklėse), elementarioji testinė situacija gerokai sudėtingesnė ir priklauso nuo formulių abiejose lyginamose žinių bazėse (žr. 5-ąją apibrėžtį). Dar sudėtingesnės bus situacijos, atspindinčios atributų reikšmių perdavimą pagal nutylėjimą, tačiau čia jų nenagrinėsime.

5-oji apibrėžtis. Tegu yra lyginamos dvi žinių bazės. Faktų aibė E yra *elementarioji testinė situacija*, gauta iš žinių bazės KB_1 taisyklės $C \leftarrow A, not\, P_1, \ldots, not\, P_n$ ($n \geqslant 0$, A susideda iš atomų arba yra tuščia aibė, $P_1, \ldots, P_n$ yra atomai), jei teisingas vienas iš šių teiginių:

1) $E = A\theta$, kur $A\theta$ yra formulės A elementarioji konkretizacija (jei $A = \varnothing$, ši situacija nenagrinėjama);

2) jei kitoje žinių bazėje KB_2 yra viena arba kelios taisyklės $D_i \leftarrow B_i, not\, R_{i1}, \ldots, not\, R_{im}$, $i = 1, \ldots, k$ (B_i susideda iš atomų arba yra tuščia aibė, $R_{i1}, \ldots, R_{im}$ yra atomai) tokios, kad $B_i\theta \setminus V \subseteq A\theta$, kur V yra vidinių KB_2 faktų aibė, tuomet

$$E = A\theta \cup R_{11}\theta \cup \ldots \cup R_{1m}\theta \cup \ldots \cup R_{k1}\theta \cup \ldots \cup R_{km}\theta,$$

kur $A\theta$ yra formulės A elementarioji konkretizacija.

Pavyzdys. Žemiau pateiktos dvi žinių bazės (1 lentelė), jų palyginimui sukonstruotos elementariosios testinės situacijos ir iš jų sugeneruoti modeliai. Žinių bazių skirtumą parodo situacija $\{S(x), P(a)\}$.

Žinių bazės, kurios išreikštos bendrosiomis loginėmis programomis, yra skirtingos tuomet ir tik tuomet, kai tarp šių žinių bazių yra vidinis skirtumas arba egzistuoja elementarioji testinė situacija, kurios atžvilgiu žinių bazės yra skirtingos [3]. Taigi,

1 lentelė

KB_1	KB_2
$R(X) \leftarrow S(X), not\, P(a)$ $Q(X) \leftarrow S(X), not\, R(X)$	$R(X) \leftarrow S(X)$
$\{S(x)\} \cup KB_1 \vdash_M \{S(x), R(x)\}$ $\{S(x), P(a)\} \cup KB_1 \vdash_M \{S(x), P(a), Q(x)\}$ $\{S(x), R(x)\} \cup KB_1 \vdash_M \{S(x), R(x)\}$ $\{S(x), P(a), R(x)\} \cup KB_1 \vdash_M \{S(x), P(a), R(x)\}$	$\{S(x)\} \cup KB_2 \vdash_M \{S(x), R(x)\}$ $\{S(x), P(a)\} \cup KB_2 \vdash_M \{S(x), P(a), R(x)\}$ $\{S(x), R(x)\} \cup KB_2 \vdash_M \{S(x), R(x)\}$ $\{S(x), P(a), R(x)\} \cup KB_2 \vdash_M \{S(x), P(a), R(x)\}$

pagal siūlomą metodą žinių bazės lyginamos taip: 1) nustatomas vidinis skirtumas, generuojant modelius iš faktų, esančių žinių bazėse, 2) sudaromos elementariosios testinės situacijos, 3) nustatomas skirtumas kiekvienos testinės situacijos atžvilgiu.

2 lentelė. Klasių ir objektų konstrukcijos Jess ir F-logikoje

Konstrukcija	**Jess**	**F-logika**
Klasė	(*deftemplate asmuo* (*slot vardas* (*type STRING*)) (*slot pavardė* (*type STRING*)) (*slot amžius* (*type INTEGER*))) Pastaba: kol kas atributo tipas Jess pakete nerealizuotas. Nors jį galima nurodyti (pvz., INTEGER, STRING), vykdant programą į jį neatsižvelgiama.	*asmuo* [*vardas* ⇒ *string*; *pavardė* ⇒ *string*; *amžius* ⇒ *integer*]. Pastaba: F-logikoje klasių, objektų ir atributų vardai gali būti parametrizuoti.
Objektas (klasės egzempliorius)	Nesutvarkytas faktas (su atributų vardais): (*assert* (*asmuo* (*vardas "Jonas"*) (*pavardė "Petraitis"*) (*amžius 34*))) Sutvarkytas faktas: (*assert* (*asmuo "Jonas" "Petraitis" 34*)) Pastaba: objektai neturi identifikatorių, tačiau galima susieti kintamąjį su objektu, pvz., (*bind ?Jonas* (*assert ...*)). Kitas būdas priskirti identifikatorių – skirti jam atskirą atributą, pavyzdžiui, *id*. Kelis objektus iš karto galima įvesti naudojant komandą *deffacts* vietoj *assert*.	*jonas*: *asmuo* [*vardas* → *"Jonas"*; *pavardė* → *"Petraitis"*; *amžius* → 34]. Objektas be išreikštinio identifikatoriaus: *: *asmuo* [*vardas* → *"Jonas"*; *pavardė* → *"Petraitis"*; *amžius* → 34].
Poklasis	(*deftemplate studentas extends asmuo*)	*studentas* :: *asmuo*.
Reikšmė pagal nutylėjimą	(*deftemplate dramblys* (*slot spalva* (*default balta*)))	*dramblys* [*spalva* ∗ → *balta*].
Daugiareikšmis atributas	(*deftemplate asmuo* (*multislot vaikai*)) (*assert* (*asmuo* (*vardas "Petras"*) (*vaikai Jonas Marytė*)))	*asmuo* [*vaikai* =≫ *asmuo*]; *petras*: *asmuo* [*vardas* → *"Petras"*; *vaikai* − ≫ {*jonas*, *marytė*}].

3 lentelė. Predikatų ir taisyklių konstrukcijos Jess ir F-logikoje

Konstrukcija	**Jess**	**F-logika**
Predikatas	(*assert* (*mirtingas Sokratas*)) (*assert* (*pilietis Jonas lietuvos-respublika*)) Pastaba: Predikatas užrašomas tiesiog kaip sąrašas (sutvarkytas faktas), kur sąrašo pradžia yra predikato vardas.	*mirtingas* (*sokratas*). *pilietis* (*jonas*, *lietuvos-respublika*).
Taisyklė	(*defrule ar_mirtingas* (*žmogus* ?*X*) ⇒ (*assert* (*mirtingas* ?*X*))) -arba- (*defrule ar_mirtingas* (*žmogus* (*vardas* ?*X*))⇒ (*assert* (*mirtingas* ?*X*))) Pastaba: Jess naudojamos produkcinės taisyklės, taisyklės išvadoje aprašomi veiksmai. Taisyklės turi vardus. Kintamųjų identifikatoriai prasideda klaustuko ženklu.	*mirtingas* (*X*) ← *X*: *žmogus*. Pastaba: F-logikoje naudojamos atvirkštinio išvedimo taisyklės, taisyklės išvadoje yra naujas teiginys. Kintamųjų identifikatoriai prasideda didžiąja raide.
Loginės operacijos taisyklėse	(*defrule ar_skraido* (*paukštis* ?*X*) (*not* (*pingvinas* ?*X*)) ⇒ (*assert* (*skraido* ?*X*))); (*defrule ar_palikuonis* (*or* (*sūnus* ?*X*, ?*Y*) (*dukra* ?*X*, ?*Y*)) ⇒ (*assert* (*palikuonis* ?*X*, ?*Y*)))	*skraido* (*X*) ← *paukštis* (*X*) *AND NOT pingvinas* (*X*). *palikuonis*(*X*, *Y*) ← *sūnus* (*X*, *Y*) *OR dukra* (*X*, *Y*).

Tiesioginio išvedimo realizacija

Kaip minėta aukščiau, F-logikos programinėse priemonėse kol kas nėra tiesioginio išvedimo. Todėl kompiuteriniams eksperimentams paimta kita priemonė – ekspertinių sistemų kūrimo paketas Jess [8]. Šis paketas parašytas Java kalba, jis gali būti panaudotas Java programose ir išplėstas naudojant Javą. Be to, Jess kalba yra ir bendro pobūdžio programavimo kalba, leidžianti pasinaudoti visomis Java klasėmis ir bibliotekomis. Galimybė išplėsti Jess kalbą leidžia įtraukti į ją F-logikos sintaksę.

Jess turi konstrukcijas, atitinkančias F-logikos konstrukcijas (žr. 2 ir 3 lenteles). Skirtingai nuo F-logikos ir Prologo, Jess išvedimo mechanizmas įgalina atlikti tiesioginį išvedimą ir taip sugeneruoti žinių bazės modelį. Tam naudojamas Rete algoritmas, kuriam būdingas palyginus didelis greitis. Jis pasiekiamas intensyviai naudojant atmintį, kurioje saugomas specialiu būdu sudarytas taisyklių prielaidų tinklas, skirtas greitai patikrinti naujų faktų atitikimą kurios nors taisyklės prielaidai.

Jess pakete dirbama su sąrašais. Tam tikrais atvejais jie įgalina atlikti daugiau veiksmų, negu F-logikoje naudojamos aibės. Pavyzdžiui, galima suskaičiuoti sąrašo elementus, o tai svarbu, pavyzdžiui, teisės dalykinėje srityje modeliuojant bausmių kiekio ribojimus.

Tarp Jess trūkumų lyginant su F-logika galima paminėti tai, kad struktūra turi būti aprašyta iš karto, o F-logikoje tai galima daryti pažingsniui. Be to, sintaksės trūkumu galima laikyti angliškus raktinius žodžius (F-logikoje apseinama be jų) ir LISP stiliaus prefiksinę funkcijų notaciją.

Išvados

Žinių bazių palyginimo uždavinys yra svarbus įvairiose dalykinėse srityse, kai analizuojami skirtumai tarp įvairių autorių sukurtų tos pačios paskirties žinių bazių arba skirtumai tarp įvairių žinių bazių versijų, atsiradusių dėl jų pakeitimų. Pasiūlytas žinių bazių palyginimo algoritmas leidžia išvengti begalybės situacijų tikrinimo ir tikrinti tik specialiu būdu sudarytas elementariąsias testines situacijas, kurios reprezentuoja visas situacijas su galimai skirtingomis pasekmėmis. Šis algoritmas pritaikytas Prologo arba F-logikos tipo taisyklinių žinių bazėms, tačiau reikalauja išvedimo mechanizmą papildyti tiesioginiu išvedimu su modelių generavimu. Eksperimentams su tiesioginiu išvedimu galima pasirinkti ekspertinių sistemų kūrimo paketas Jess. Nors tai nėra išbaigtas paketas ir jame trūksta kai kurių svarbių F-logikos galimybių, eksperimentams jis tinka, juolab kad jį galima tobulinti naudojant Javą ir pradinius tekstus, kurie akademiniam naudojimui pateikiami nemokamai.

Literatūra

1. L. Paliulionienė, Ontologijų naudojimas teisinių žinių bazėms lyginti, Lietuvos mokslas ir pramonė. *Informacinės technologijos 2004*, Konferencijos pranešimų medžiaga, Technologija, Kaunas (2004), pp. 569–574.
2. L. Paliulionienė, Požiūrių vaizdavimas ir jų skirtumų analizė teisinių dokumentų rengimo sistemų žinių bazėse, Lietuvos mokslas ir pramonė, A. Otas, V. Petrauskas, H. Pranevičius, R. Šeinauskas (red.), *Informacinės technologijos-99*, Konferencijos pranešimų medžiaga, Kauno technologijos universitetas, Technologija, Kaunas (1999), pp. 94–98.
3. L. Paliulionienė, Kai kurie teisinių žinių bazių lyginamosios analizės aspektai, *Lietuvos matematikų draugijos XL konferencijos darbai*, Vilnius, 1999 m. birželio mėn. 21–22d., Technika, Vilnius (1999), pp. 230–235.
4. V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, *ACM Transactions on Computational Logic*, **2** (2001), 526–541.
5. P.L. Hammer, A. Kogan, Essential and redundant rules in Horn knowledge bases, *Decision Support Systems*, **16**(2), 119–130 (1996).
6. M. Kifer, G. Lausen, J. Wu, *Logical Foundations of Object-Oriented and Frame- Based Languages*, Technical Report 93/06, Department of Computer Science, University SUNY at Stony Brook (1993).
7. R. Hasegawa, Parallel theorem-proving system: MGTP, in: *Proc. of FGCS'94*, ICOT, Tokyo (1994), pp. 51–56.
8. E.J. Friedman-Hill, *Jess, The Rule Engine for the Java Platform.* Distributed Computing Systems, Sandia National Laboratories, Livermore, CA, Version 6.1p7 (7 May 2004). `http://herzberg.ca.sandia.gov/jess/docs/61/`,

SUMMARY

L. Paliulionienė. Finding internal and situational differences between two knowledge bases

The task of comparing knowledge bases can emerge in different domains. It is sometimes essential to compare knowledge bases created by different experts for the same goal, or different versions of a knowledge base after amendments. The article presents a method of the automated comparing of knowledge bases, and offers an approach to its implementation. The method uses forward inference and model generating. Jess, the expert system shell for the Java platform, is proposed as an experiment basis, and constructs of F-logic are modelled by Jess.

Keywords: knowledge representation, inference in knowledge bases, model generation, comparison of knowledge bases.