*Liet. matem. rink,* **46***, spec. nr.*, 2006, 107–111

# Žinių bazių lyginamoji analizė: sistemos architektūra

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

**Reziumė.** Lyginant panašias žinių bazes kyla klausimas, kokiose situacijose skiriasi išvados iš šių žinių bazių. Šis uždavinys gali iškilti įvairiose dalykinėse srityse, pavyzdžiui, teisėje, kai reikia palyginti skirtingas teisės akto versijas arba skirtingų šalių analogiškus teisės aktus. Straipsnyje pateikta programų sistemos, skirtos žinių bazių palyginimui, architektūra, realizuojanti metodą, grindžiamą elementariųjų testuojamųjų situacijų konstravimu.

*Raktiniai žodžiai:* žinių bazių palyginimas, išvedimas teisinių žinių bazėse, modelių generavimas, elementariosios testuojamosios situacijos.

## Įvadas

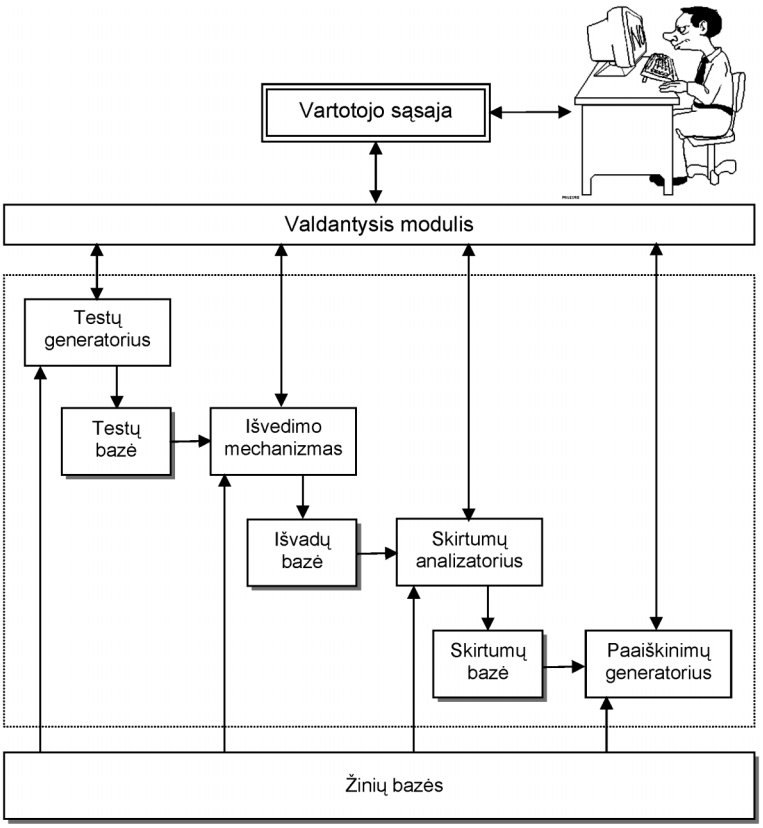
Žinių bazių palyginimo uždavinys gali iškilti įvairiose dalykinėse srityse dviem pagrindiniais atvejais: 1) kai nagrinėjamos skirtingos žinių bazės versijos, atsiradusios dėl jos keitimo ir 2) kai nagrinėjamos skirtingų autorių sukurtos tos pačios paskirties žinių bazės. Lyginant panašias žinių bazes kyla klausimas, kokiose situacijose skiriasi išvados iš šių žinių bazių.

Žinių bazių automatizuotas palyginimas iki šiol nėra išnagrinėtas taisyklinių žinių bazių teorijoje. Panašiu uždaviniu galima laikyti žinių bazių ekvivalentiškumo nustatymą, nagrinėjamą darbuose [1] ir [2], tačiau tuo atveju tikslas yra ne išryškinti skirtumus, bet suprastinti žinių bazę ekvivalentiškomis transformacijomis; be to, šiuose darbuose apsiribojama teiginių logika.

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. Galimų situacijų gali būti labai daug, kartais net begalinė aibė, todėl tiesiogiai visų jų patikrinti neįmanoma. Ankstesniuose mūsų darbuose [3, 4] šis uždavinys buvo nagrinėtas tiriant išvedimą teisinių žinių bazėse, buvo pasiūlytas jo sprendimo metodas ir parodyta, kad vietoj visų įmanomų situacijų tikrinimo pakanka patikrinti vadinamąsias elementariąsias testuojamąsias situacijas, tam tikru būdu sukonstruotas iš konkretizuotų taisyklių antecedentų. Tęsiant šiuos tyrimus, šiame straipsnyje pateikta programų sistemos, skirtos taisyklinių žinių bazių lyginamajai analizei, architektūra.

## Sistemos architektūra

Lyginamosios analizės sistemos architektūros loginė struktūra pateikta 1 pav.




1 pav. Žinių bazių lyginamosios analizės sistemos architektūra.

Pirmiausia aptarsime žinių vaizdavimą *žinių bazėse*. Mes nagrinėjame taisyklines žinių bazes, kuriose žinios vaizduojamos kaip faktai ir taisyklės, kurios gali būti loginės arba produkcinės. Pavyzdžiui, žinioms vaizduoti gali būti naudojama freimų logika (F-logika) arba ekspertinių sistemų kūrimo paketas Jess. F-logika [5] – formalizmas, jungiantis objektinį žinių vaizdavimo būdą su loginio išvedimo mechanizmu, grindžiamu rezoliucija. F-logikos žinių bazę sudaro taisyklės, kurių forma yra *antraštė*←*kūnas*, kur *antraštė* yra F-molekulė, *kūnas* yra F-molekulių konjunkcija. Faktas yra taisyklė be kūno. F-molekulėse naudojami sąvokų vardai, atributų vardai, atributų reikšmės, predikatų simboliai. Ekspertinių sistemų kūrimo paketas Jess [6] parašytas Java kalba, jis gali būti panaudotas Java programose ir išplėstas naudojant Javą. Jess kalbos sintaksė panaši į ekspertinių sistemų kūrimo priemonės CLIPS kalbos sintaksę, kuri savo ruožtu yra specializuota LISP versija. Jess turi konstrukcijas, atitinkančias F-logikos konstrukcijas (žr. 1 lentelę). 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.

1 lentelė. Pagrindinės konstrukcijos Jess ir F-logikoje

| **Konstrukcija** | **Jess** | **F-logika** |
|---|---|---|
| Klasė | (*deftemplate asmuo*<br>(*slot vardas* (*type STRING*))<br>(*slot pavardė* (*type STRING*))<br>(*slot amžius* (*type INTEGER*)))<br>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*].<br><br>Pastaba: F-logikoje klasių, objektų ir atributų vardai gali būti parametrizuoti. |
| Objektas (klasės egzemplioriu s) | Nesutvarkytas faktas (su atributų vardais):<br>(*assert* (*asmuo* (*vardas* "*Jonas*") (*pavardė* "*Petraitis*") (*amžius 34*)))<br>Sutvarkytas faktas:<br>(*assert* (*asmuo* "*Jonas*" "*Petraitis*" *34*)) | *jonas* : *asmuo* [*vardas* − > "*Jonas*"; *pavardė* − > "*Petraitis*"; *amžius* − > 34].<br>Objektas be išreikštinio identifikatoriaus:<br>* : *asmuo* [*vardas* − > "*Jonas*"; *pavardė* − > "*Petraitis*"; *amžius* − > 34]. |
| Predikatas | (*assert* (*mirtingas Sokratas*))<br>(*assert* (*pilietis Jonas lietuvos_respublika*))<br>Pastaba: Predikatas užrašomas tiesiog kaip sąrašas (sutvarkytas faktas), kur sąrašo pradžia yra predikato vardas. | *mirtingas* (*sokratas*).<br>*pilietis* (*jonas*, *lietuvos_respublika*). |
| Taisyklė | (*defrule ar_mirtingas* (*žmogus ?X*) => (*assert* (*mirtingas ?X*))) -arba-<br>(*defrule ar_mirtingas* (*žmogus* (*vardas ?X*)) => (*assert* (*mirtingas ?X*)))<br>Pastaba: Jess naudojamos produkcijų taisyklės, taisyklės išvadoje aprašomi veiksmai. Taisyklės turi vardus. Kintamųjų identifikatoriai prasideda klaustuko ženklu. | *mirtingas* (*X*)← *X*: *žmogus*.<br><br>Pastaba: F-logikoje naudojamos atvirkštinio išvedimo taisyklės, taisyklės išvadoje yra naujas teiginys. Kintamųjų identifikatoriai prasideda didžiąja raide. |

*Vartotojo sąsaja* gali būti tiek grafinė, tiek tekstinė. Paprasčiausiu atveju tai gali būti Jess komandinės eilutės sąsaja, F-logikos užklausų kalba arba kitokia užklausų kalba grindžiama sąsaja. Vartotojo sąsajos komponentas turi būti kiek įmanoma atskirtas nuo pačios sistemos, o sistemoje turi būti realizuota vidinė kalba, kurią galima po to įvairiais būdais pateikti vartotojui.

*Valdantysis modulis* priima duomenis iš vartotojo, transformuoja juos į vidinę kalbą, valdo užklausos vykdymą, transformuoja rezultatus į vartotojo sąsajos kalbą.

*Testų generatorius* iš žinių bazių taisyklių ir užklausos duomenų generuoja faktų-testų bazę. Ankstesniuose mūsų darbuose [3, 4] šis uždavinys buvo nagrinėtas tiriant išvedimą teisinių žinių bazėse, buvo pasiūlytas jo sprendimo metodas ir parodyta, kad vietoj visų įmanomų situacijų tikrinimo pakanka patikrinti situacijas, tam tikru būdu sukonstruotas iš konkretizuotų taisyklių antecedentų. Tokias situacijas vadiname

elementariosiomis testuojamosiomis situacijomis. Faktai testų bazėje vaizduojami ta pačia kalba, kaip ir žinių bazėse.

*Išvedimo mechanizmas* generuoja naujus faktus pagal tam tikras išvedimo taisykles. Elementariųjų testuojamųjų situacijų metodas reikalauja tiesioginio, t.y. prasidedančio nuo faktų, išvedimo, kad galėtume gauti visas konkrečios situacijos išvadas. Todėl jei žinioms vaizduoti naudojama F-logika, ją būtina praplėsti formaliu įrodymu, grindžiamu *modelių generavimu* [3], nes standartinis F-logikos išvedimo mechanizmas įgalina gauti tik atsakymą į užklausą panašiai kaip Prologe. Pagal modelio generavimo taisyklę į modelį pirmiausia įtraukiami faktai, toliau pagal taisyklių grandines gaunamos išvados iš šių faktų. Skirtingai nuo F-logikos ir Prologo, Jess išvedimo mechanizmas įgalina atlikti tiesioginį išvedimą ir taip sugeneruoti žinių bazės modelį, todėl Jess tinka eksperimentams žinių bazių lyginamojoje analizės srityje. Jess išvedime 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.

Išvedimo mechanizmo darbo rezultatai (faktai-išvados) pateikiami *skirtumų analizatoriui*, kuris palygina tų pačių faktų išvadas, gautas skirtingose žinių bazėse, ir išrenka nesutampančias išvadas. Tai daroma taikant operacijas su aibėmis ir ieškant išvadų aibių skirtumo. *Paaiškinimų generatorius* pateikia šiuos skirtumus vartotojo analizei.

## Teisės aktų rengimo ir analizės sistema ir žinių bazių palyginimas

Kaip minėjome, viena iš dalykinių sričių, kurioje iškyla žinių bazių palyginimo uždavinys, yra teisė, tiksliau – teisės aktų rengimo kompiuterizavimas. Galima išskirti dvi teisės aktų lyginamosios analizės rūšis: a) skirtingų to paties teisės akto versijų palyginimas (pavyzdžiui, alternatyvių projektų, projekto ir galiojančio dokumento, dokumento iki ir po pakeitimo palyginimas), b) skirtingų valstybių analogiškų teisės aktų palyginimas. Tai yra sudėtingas uždavinys, ypač kai teisės aktas yra didelės apimties ir sudėtingos struktūros.

Taikant dirbtinio intelekto metodus teisėje, lyginamosios analizės posistemis yra viena iš intelektualizuotos teisės aktų rengimo ir analizės sistemos dalių. Be šio posistemio tokioje sistemoje turi būti dalys, skirtos atlikti teisės aktų rengimo procedūrų valdymą, tekstų ir žinių bazių redagavimą, pagalbą konvertuojant tekstą į žinių bazę ir generuojant tekstą iš žinių bazės, išvedimą žinių bazėse ir atsakymą į užklausas, suderinamumo ir išsamumo tikrinimą, koncepcinę žinių bazių analizę. Kompiuterizuota teisės aktų rengimo ir analizės sistema turėtų palengvinti įstatymus bei kitus teisinius dokumentus rengiančių grupių darbą ir sudaryti prielaidas kokybiškesniems dokumentams rengti.

## Išvados

Žinių bazių palyginimo uždavinys iki šiol yra beveik netirtas dirbtinio intelekto teorijoje, tačiau 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. Straipsnyje pateikta sistemos architektūra leidžia eksperimentuoti realizuojant ankstesniuose autoriaus darbuose pasiūlytą žinių bazių palyginimo algoritmą, kuris leidžia išvengti begalybės situacijų tikrinimo ir tikrinti tik specialiu būdu sudarytas elementariąsias testines situacijas, kurios reprezentuoja visas situacijas su galimai skirtingomis pasekmėmis. Eksperimentams pasirinkta F-logika ir ekspertinių sistemų kūrimo paketas Jess, kuris kartu su pradiniais tekstais akademiniam naudojimui pateikiamas nemokamai.

## Literatūra

1. V. Lifschitz, D. Pearce and A. Valverde, Strongly equivalent logic programs, *ACM Transactions on Computational Logic*, **2**, 526–541 (2001).
2. P.L. Hammer, A. Kogan, Essential and redundant rules in Horn knowledge bases, *Decision Support Systems*, **16**(2), 119–130 (1996).
3. L. Paliulionienė, Ontologijos teisinių dokumentų rengimo ir analizės sistemose, *Informacijos mokslai*, **26**, 176–179 (2003).
4. L. Paliulionienė, Kai kurie teisinių žinių bazių lyginamosios analizės aspektai, kn.: *Lietuvos matematikų draugijos XL konferencijos darbai*, Vilnius, 1999 m. birželio mėn. 21–22 d., Technika, Vilnius (1999), pp. 230–235.
5. M. Kifer, G. Lausen and 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).
6. E.J. Friedman-Hill, Jess, *The Rule Engine for the Java Platform*, `http://herzberg.ca.sandia.gov/jess/docs/61/`, Distributed Computing Systems, Sandia National Laboratories, Livermore, CA, Version 6.1p7 (2004).

SUMMARY

***L. Paliulionienė. Comparative analysis of knowledge bases: system architecture***

When comparing similar knowledge bases the question arises: what situations produce different conclusions in these bases? The task of comparing knowledge bases can emerge in different domains. For example, in legal domain, it is sometimes essential to compare different versions of a legal act or similar legal acts of different countries. The article presents an architecture of the system designed for the automated comparing of knowledge bases. The architecture implements the method of comparing based on the construction of elementary test situations.

*Keywords:* comparing of knowledge bases, legal knowledge bases, model generation, elementary test situations.