Vilniaus universitetas
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
Programų sistemų katedra

Ontologiškai teisingas koncepcinis modeliavimas
Ontologically correct conceptual modelling

Magistro baigiamasis darbas

Atliko: Ana Griščenko

Darbo vadovas: lektorius Donatas Čiukšys

Recenzentas: Prof. Dr. Romas Baronas

Vilnius – 2009

Santrauka

OntoClean – tai metodika, skirta ontologijų korektiškumo vertinimui atlikti. Tačiau, dėl šios metodikos sudėtingumo, ja gali naudotis tik maža žmonių grupė. Šio darbo tikslas yra ištirti, kaip galima būtų supaprastinti šios metodikos taikymą. Tam, kad pasiekti šį tikslą buvo padaryta metodikos analizė, o taip pat buvo apžvelgta, kaip galima atlikti ontologijų vertinimą įvairių įrankių pagalba. Buvo nustatyta, kad didžioji darbo, susijusio su vertinimu, dalis – savybių priskyrimas klasėms – daroma rankiniu būdu. Šiai problemai spręsti buvo pasiūlytas algoritmas, pilnai arba dalinai automatizuojantis OntoClean metodiką. Pagrindinė algoritmo idėja – meta-savybių įrodymų paieška internete panaudojant paieškos frazių šablonus, kurie išreiškia teigiamą arba neigiamą meta-savybės įrodymą.

Raktiniai žodžiai: ontologija, meta-savybė, šablonas, kalbos analizatorius, klasifikatorius, reguliari išraiška.

Summary

OntoClean is a methodology for assessing corectness of ontology. However, because of the difficultness of the methodology, only a small group of people can use it. The goal of this work is to find out how to facilitate usage of OntoClean. To reach the goal an anglysis of OntoClean was made, as well as a survey of tools, that support usage of OntoClean. A conclusion was that a major task of ontology assessment – assignment of mata-properties to classes – was done manually. To solve this problem, an algorithm that fully or partially automates this process, was introduced. The main idea of this approach is searching for meta-property evidences in the Internet by using templates of a search phrase, that express positive or negative evidence of meta-property.

Keywords: ontology, meta-property, template, linguistic analyser, classifier, regullar expression.

Turinys

Įvadas

Šiuo metu ontologijų – viešų formalių dalykinės srities koncepcijų ir sąryšių tarp jų aprašymų - kūrimas labai išpopuliarėjo. Dabar tai jau nebe mistinė sąvoka, kurią galima buvo tyrinėti tik dirbtinio intelekto laboratorijose. Kodėl atsirado poreikis kurti ontologijas? Pirma, vienodas informacijos struktūros supratimas tarp žmonių ir programinių agentų. Tarkime, jeigu dvi interneto svetainės naudojasi ta pačia ontologija, kitos programos gali užklausti informaciją iš tų dviejų svetainių, pavyzdžiui, vartotojo poreikiams patenkinti, naudodami tą pačią terminologiją. Antra, dalykinės srities žinios gali būti atskirtos nuo sistemos realizacijos, o tai pat jos gali būti pakartotinai panaudojamos kitose programose.

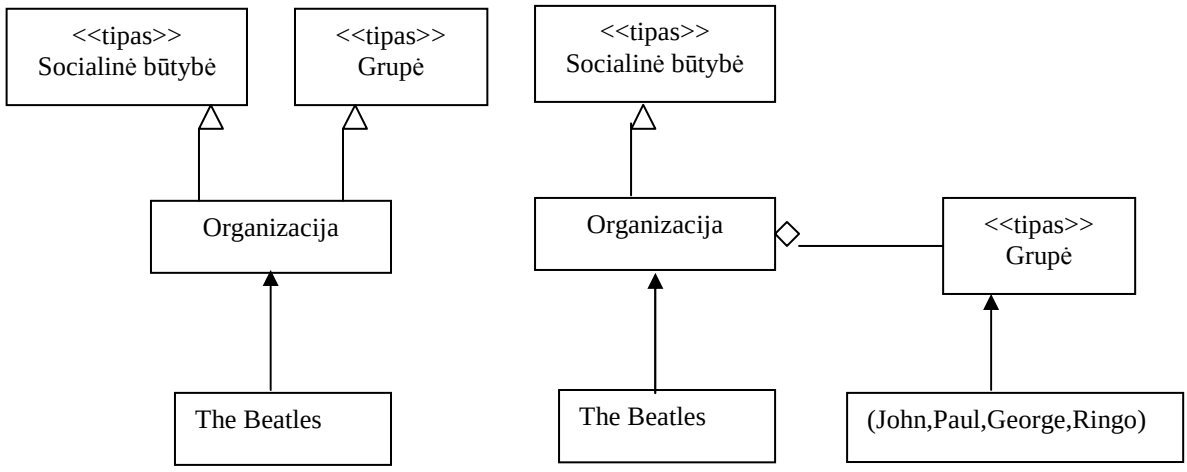
Ontologijos yra plačiai naudojamos Internete, pavyzdžiui, interneto svetainė, reklamuojanti ir klasifikuojanti siūlomas prekes ar paslaugas Amazon.com. Daugelyje dalykinių sričių yra kuriamos standartinės ontologijos, kad dalykinių sričių ekspertai galėtų tomis ontologijomis naudotis ir publikuoti informaciją apie savo sritį.

Kas gi yra ontologija? Filosofijoje Ontologija (didžioji „o“) - tai mokslas apie pačias bendriausias charakteristikas, kurias gali turėti esybė arba jos rūšis. Informatikoje ontologija (mažoji „o“) - tai formalus dalykinės srities terminų ir sąryšių tarp jų aprašymas, reikalingas žinių apie dalykinę sritį sistematizavimui [4]. Su šiuo terminu yra susietas terminas „ontologinis modeliavimas“.

Kuo skiriasi koncepcinis modeliavimas nuo ontologinio? Koncepcinis modeliavimas susijęs su esminių dalykinės srities koncepcijų ir jų apribojmų identifikavimu, analize ir aprašymu naudojant modeliavimo kalbą [1]. Tuo tarpu ontologinis modeliavimas susijęs su dalykinės srities esybių įtraukimu į ontologiją ir remiasi ontologijų specifikavimo kalba, kuri sudaryta iš ontologinių klasių, nepriklausomų nuo dalykinės srities [1]. Pavyzdžiui, objektiškai orientuotame programavime projektuotojas priima projektinius sprendimus remdamasis tuo, kokius metodus turi klasė, t.y. remdamasis klasės operacinėmis savybėmis, tuo tarpu ontologijų projektuotojas priima sprendimus atsižvelgiant į klasės struktūrines savybes. Todėl gaunasi, kad klasės struktūra ir ryšiai tarp klasių ontologijoje skiriasi nuo panašios dalykinės struktūros objektiškai orientuotoje programoje.

Tam, kad ontologijos būtų pakartotinai panaudojamos, jas reikia kurti gerai, todėl reikalingi metodai padedantys kurti ontologijas su teisingais ryšiais tarp esybių. Tam, kad patikrinti ryšių tarp dalykinės srities esybių ontologinį korektiškumą gali būti naudojamos įvairios metodikos (OntoMetric [8], FIGO [9], OntoClean). OntoClean buvo pirmą kartą viešai pristatyta 2000 metais ir nuo to laiko vis populiarėja. Šios metodikos pradininkas yra N.

Guarino ir C. Welty. OntoClean metodika turi gilų filosofinį pagrindą ir išskiria kelias meta-savybes, pavyzdžiui, *tapatumas* (angl. *identity*) ir *vienybė* (*unity*), kurios būdingos kiekvienai esybės rūšiai (klasei), ir nustato ryšių apribojimus tarp klasių priklausomai nuo to, kokiomis meta-savybėmis jos pasižymi.



1 pav. **Ontologiškai neteisinga (a) ir teisinga (b) diagramos [1].**

1 pav. parodyta, kaip OntoClean pagalba galima pertvarkyti ontologiškai neteisingą modelį į ontologiškai teisingą. Tam buvo įvesti keli klasių tipai, tokie kaip „rūšis“, „porūšis“, „kategorija“ ir t.t. OntoClean nusako pagal aukščiau aprašytas meta-savybes, kokie ryšiai gali būti tarp klasių, pavydžiui, „rūšis“ negali paveldėti iš „porūšies“ Pirmoje diagramoje yra pažeidžiamas identiškumo principas, kuris nusako, kad kiekvienas objektas gali paveldėti tik iš vienos klasės tipo „rūšis“. Šios problemos sprendimas pavaizduotas antroje diagramoje.

Darbo tikslas ir uždaviniai

Vienas didelis OntoClean trūkumas, dėl kurio projektuotojas gali atsisakyti ją taikyti, yra šios metodikos sudėtingumas. Tam, kad pritaikytų metodiką, projektuotojas turi pirmiausia, suprasti apibrėžimus, kas nėra labai lengva, mokėti modalinę logiką, bei turėti labai geras dalykinės srities žinias. Be to, ontologijos vertinimas pagal OntoClean yra daug laiko atimantis procesas. Pavyzdžiui, jeigu ontologijos dydis yra virš 200 klasių, vertinimas gali užtrūkti kelias dienas.

Magistrinio darbo uždaviniai:

1. Ištirti metodiką OntoClean, suprasti, kaip atliekamas vertinimas.

2. Padaryti šiuo metu rinkoje esančias ontologijų kūrimą palaikančias priemones. Atlikti kai kurių labiausiai naudojamų įrankių, palaikančių OntoClean, analizę, išskirti ontologijų vertinimo problemas.

Magistrinio darbo tikslas:

1. Rasti aukščiau aprašytos problemos sprendimą, t.y. pasiūlyti tokį algoritmą, kuris palengvintų ontologijų vertinimą pagal OntoClean; sugalvoti tokį algoritmą, kurio naudojimas reikalaus iš naudotojo tik pačių bendriausių žinių apie ontologijas ir apie metodiką.

1. Metodika OntoClean

1.1Metodikos pagrindas

Pagrindinės OntoClean sąvokos jau kurį laiką egzistavo filosofijoje, todėl nėra naujos. Iš tikrųjų, dar Aristotelio laikais filosofai bandė formaliai ir logiškai aprašyti pasaulį. Jie stengėsi išspręsti paslaptingas egzistencijos problemas, tokias kaip Dievas, gyvenimas ir mirtis, arba pvz., ar statula ir marmuras, iš kurio ji sudaryta yra ta pati esybė. Nors šie klausimai neturi nieko bendro su informacinių sistemų modeliavimu, tačiau koncepcinė analizė ir metodai, naudojami atsakyti į juos turi daug analogijų. Jie ir formuoja OntoClean pagrindą.

1.1.1 Sistematika

Sistematika tai svarbiausia visų koncepcinių modelių dalis. Modelis, sukurtas naudojant tinkamą sistematiką gali būti pakartotinai panaudojamas, jį lengviau integruoti su kitais modeliais, jis gali būti labai naudingas nagrinėjant atskiras modelio dalis. Netinkamai sudarytos sistematikos daro priešingą įtaką, t.y. modeliai tampa sudėtingi, painūs, bei juos sunku pakartotinai panaudoti ir integruoti į kitus modelius.

Todėl žinios apie tai, kaip teisingai sudarinėti sistematiką labai praverstų. Projektuojant sistematikas padidinto dėmesio objektas buvo taksonominių ryšių semantika (pavyzdžiui is-a, class inclusion ir pan.), įvairūs ryšių tipai (pavyzdžiui, apibendrinimas, specializacija, poaibių hierarchija) priklausomai nuo apribojimų, taikomų taksonominiams ryšiams ir t.t.

Esminis skirtumas tarp OntoClean sistematikos ir aukščiau išvardintų sistematikų yra tai, kad OntoClean koncentruojasi ties argumentu, t.y. svarbu kokias meta-savybes jis įgija [10]. Yra tik vienas ryšys, jungiantis du argumentus, tai yra „apėmimo“ (angl. *subsumption*) ryšys. Sakoma, kad savybė Φ *apima* savybę ψ tada ir tik tada, kai kiekvienas galimas Φ atvejis taip pat yra ir ψ atvejis [10]:

$$\forall x \Phi(x) \rightarrow \psi(x). \qquad (1)$$

Taigi, OntoClean tikslas yra įvertinti ryšio patikimumą ir pagrįstumą remiantis dviejų savybių Φ ir ψ ontologine prigimtimi.

1.1.2 Pagrindinės sąvokos

OntoClean pagrindą sudaro keturios filosofinės sąvokos: *tapatumas* (angl. *identity*), *vienybė* (*unity*), *esmė* (*essence*) ir *priklausomybė* (*dependence*). Tapatumo sąvoka šiame kontekste susieta su mūsų, kaip pažintinių agentų, sąveika su atskiromis esybėmis ir pasauliu aplink mus. Nepaisant jos fundamentalios svarbos filosofijoje, tapatumo sąvoka negreitai buvo pritaikyta koncepciniame modeliavime, kadangi koncepcinio modeliavimo ir pasaulio apsakymo tikslai iš esmės vienodi – sistematizuoti informaciją.

Aptarkime *tapatybė ir vienybės* panašumus ir skirtumus. Šios sąvokos skiriasi, tačiau vienybė kartais gali būti supainiojama kaip bendra tapatumo sąvoka. Griežtai kalbant, tapatybė susijusi su problema kaip atskirti klasės objektą nuo kitų tos pačios klasės objektų (atsakymas į klausymą „Ar tai *mano* šuo?“). Šią problemą galima išsspręsti kiekvienam klasės objektui priskirus unikalią savybę (pavyzdžiui, žmogus turi asmens kodą). Vienybė, iš kitos pusės, yra susijusi su problemą kaip atskirti objekto „dalis“ nuo kitų pasaulio „dalių“ („Ar apykaklė yra mano šuns dalis?“) panaudojant sujungiamąjį ryšį, kuris susieja tiktai šio objekto dalis.

Tačiau, laikui bėgant susiduriama su tam tikromis problemomis. Pavyzdžiui, kaip įrodyti kad žmogus po 40 metų yra tas pats žmogus? Tai yra klasikinis *identity through change* klausimas. Turime pripažinti, kad individas gali išlikti tuo pačiu individu demonstruodamas skirtingas savybes laikui bėgant. Vienos savybės išlieka tos pačios, kitos keičiasi. Taigi kaip galima identifikuoti tam tikrą savybę turinti objektą po kažkiek laiko? Ši problema lydi prie *ėsminės* savybės sąvokos, kuria remiasi *griežtumo* sąvoka.

Ketvirta, ontologinės *priklausomybės* sąvoka, apima daug skirtingų ryšių, pavyzdžiui, ryšys tarp vaiko ir tėvo, ryšys tarp skylių sūryje ir sūriu ir pan. Čia eina kalba apie priklausomybę tarp objekto savybių. Iš šios sąvokos išplaukia *išorinės* ir *vidinės* savybės, priklausomai nuo to, ar jie priklauso nuo kitų objektų. Vidinė savybė yra tai, kas įgimta individui ir nepriklauso nuo kitų individų, pavyzdžiui, širdies turėjimas. Išorinės savybės nėra įgimtos, jų prigimtis yra reliacinė, t.y. jos negali egzistuoti be kito objekto, pavyzdžiui, savybė „aš esu Jovitos draugė“ yra beprasmė, jeigu Jovita neegzistuoja. Kai kurios išorinės savybės gali būti priskirtos objektui kitų objektų, pavyzdžiui vartotojo, ID mums duoda sistemos administratorius.

Svarbu pabrėžti, kad visos šitos sąvokos turi tam tikrą galiojimo sritį, t.y. priklauso nuo pasaulio konceptualizacijos. Pavyzdžiui, jeigu aš pasakysiu, kad pirštų antspaudai yra žmogaus identifikacijos būdas, tai negalima teigti, kad tai universalus žmonių identifikavimo būdas. Jis gali turėti įtaką tik kažkokiame modelyje, pasaulio konceptualizacijoje.

1.1.3 Įvadas į modalinę logiką

Meta-savybių aprašymui naudojama modalinė logika. Modalinė logika – tai samprotavimas naudojant „būtinumo“ ir „galimumo“ išraiškas. Ji remiasi Kripkė logika.

Modalinėje logikoje būtinumas žymimas □, o galimumas - ◊. Galimumas gali būti išreikštas per būtinumą: ◊A = ~□~A. Operatoriai □ ir ◊ labai panašūs į kvantorius ∀ (visi) ir ∃ (kai kurie).

Meta-savybės aprašomos vadinamąja S5 modalinę lokiką, kuri apibrėžiama dvejomis aksiomomis [7]:

(M) □A→A, ir

(5) ◊A→□◊A.

S5 logikoje eilutės sudarytos iš operatorių (kvadratų ir rombų) lygios paskutiniam operatoriui toje eilutėje. Pavyzdžiui, „įmanoma, kad A būtinas“ yra tas pats kas „A būtinas“, t.y.

00...□ = □ ir 00...◊ = ◊, kur 0 lygus □ arba ◊.

Modalinės logikos sakiniai turi prasme *galimų pasaulių* kontekste. Galimas pasaulis tai toks pasaulis, kurio charakteristikos arba istorija skiriasi nuo mūsų pasaulio [6]. Pavyzdžiui, fantastinėse romanuose aprašomi galimi pasauliai, kurie daugiau ar mažiau skiriasi nuo mūsų pasaulio. Reikalaujama, kad galimas pasaulis būtų ne bet koks, o *pastovus logine prasme*, t.y. pasaulis kuriame galioja matematikos, fizikos dėsniai. Pavyzdžiui, gali būti pasaulis, kuriame Džordžas Bušas nėra JAV prezidentas, bet negali būti pasaulio, kuriame 2+2=5.

Taigi, □φ reiškia, kas φ yra „būtina“ kiekviename galimame pasaulyje, o ◊φ reiškia, kad φ yra „galima“ bent viename galimame pasaulyje. Predikatų galiojimo sritis nėra apribota tik tuo kas egzistuoja tikrame pasaulyje, bet tuo, kas egzistuoja kiekviename galimame pasaulyje. Visi pasauliai yra vienodai pasiekiami.

Pavyzdžiui, predikatas „Vienaragis“ bus netuščias nors faktiškai jis neegzistuoja. Faktinis egzistavimas skiriasi nuo loginio egzistavimo ir bus žymimas *E(x,t)*, kas reiškia, kad *x* tikrai egzistuoja momentu *t* šitame pasaulyje [10].

Analizės metu kai kuriais atvejais bus naudojamasi laiko logikos (angl. *temporal logic*) formulėmis, kurioje predikatuose atsiranda laiko parametras. Jeigu laiko parametras praleidžiamas reiškia, predikatas pastovus laike: $\exists t\phi(x,t) \rightarrow \forall t\phi(x,t)$.

1.2 Ontologinės analizės įrankiai

1.2.1 Griežtumas

Lowe [11] apibrėžia esminę savybę kaip savybę, kuri yra būtina objektui, t.y. objektas turi šią savybę turi kiekviename galimame pasaulyje. *Esmingumas* - tai ryšys tarp individo ir savybės. *Griežtumo* sąvoka labai susijusi su esmingumu ir, kaip paaiškėjo, yra labai naudinga koncepciniame modeliavime [10]. Toliau, apibrėžimuose bus naudojama *savybės* sąvoka, kuri yra panaši į klasės sąvoka objektiniame programavime. O atvejis yra šios savybės realizacija, pavyzdžiui, x yra savybės Φ realizacija.

Apibrėžimas 1. Savybė φ vadinama *griežta* jeigu ji yra esminė *visiems* jos egzemplioriams, t.y. $\Box(\forall x t \phi(x,t) \rightarrow \Box \forall t' \phi(x,t'))$.

Šia formulę galima taip paaiškinti: Iš to, kad viename galimame pasaulyje visi x turi savybę Φ laiku t išplaukia, kad visuose galimuose pasauliuose x visada turės savybę Φ.

Apibrėžimas 2. Savybė φ vadinama *negriežta* jeigu ji yra neesminė *kai kuriems* egzemplioriams, t.y. $\Diamond(\exists x, t \phi(x,t) \wedge \Diamond \exists t' \neg \phi(x,t'))$.

Apibrėžimas 3. Savybė φ vadinama *anti-griežta* jeigu ji yra neesminė *visiems* egzemplioriams, t.y. $\Box(\forall x t \phi(x,t) \rightarrow \Diamond \exists t' \neg \phi(x,t'))$.

Sakome, kad „būti žmogumi“ yra esminė savybė. jeigu *x* yra savybės ‚žmogus‘ egzempliorius, tai jis turi būti jo egzemplioriumi kiekviename galimame pasaulyje. Savybė ‚studentas‘, iš kitos pusės, yra neesminė, kadangi žmogus gali būti, o gali ir nebūti studentu kitame galimame pasaulyje.

Anti-griežtumas yra negriežtumo apribojimas. Anti-griežtumas apriboja visus elementus kurie turi šią savybę, tuo tarpu negriežtumas yra paprastas griežtumo paneigimas ir nėra labai informatyvus. Anti-griežtumas reiškia, kad visi elementai turintys savybę galbūt gali neturėti šios savybės, t.y. savybė nera būtina. Pavyzdžiui, žmogus nebūtinai turi būti studentu.

Negrieštumas reiškia, kad kai kuriems savybės egzemplioriams ši savybė yra privaloma, o kitiems neprivaloma. Pavyzdžiui, „uodega“ yra negriežta savybė. Kengūrai ar žuviai ši savybė esminė, nes be uodegos jos negali judėti, o reiškia ir egzistuoti. Užtat žmogui uodega visai nereikalinga. Iš kitos pusės šunys iš prigimties turi uodegą, bet gali gyventi ir be jos.

Modalinis būtinumas (kvadrato operatorius) dažnai supainiojamas su laikinu pastovumu, tačiau tai bendresnė sąvoka. Pavyzdžiui, žmogus gali būti studentu visa savo gyvenimą, tačiau,

tai nesulaužo savybės ‚žmogus‘ griežtumo.Tai, kad žmogus visa gyvenimą studijuoja daugiau yra išimtis, negu dažnai pasitaikantis atvejis, t.y. jis *nebūtinas*. Įtaka informacinėms sistemoms yra tokia, kad anti-griežta savybė galbūt gali keistis, nors nėra reikalavimo kad visi egemplioriai turi keistis. Tuo tarpu griežta savybė keistis negali. Negriežta savybė neturi apribojimų, t.y. vieniems jos egzemplioriams ji gali būti ėsminė, kitiems gali keistis.
Girežtos savybės žymimos raide +R, negriežtos žymimos kaip –R ir anti-griežtos žymimos ~R.

1.2.2 Tapatumas

Filosofijoje tapatumo kriterijus (angl. *identity condition, IC*) savybei φ apibrėžiamas kaip ryšys ρ tenkinantis formulę [12]

$$\phi(x) \wedge \phi(y) \rightarrow (\rho(x, y) \leftrightarrow x = y) \quad (2)$$

Pavyzdždiui, savybės ŽMOGUS tapatybės kriterijumi gali būti savybė ‚turėti tą patį asmens numerį‘arba ‚turėti tuos pačius pirštų antpaudus‘, jeigu jos tenkina (2). Jeigu ryšys ρ tenkinantis lygybę (2) egzistuoja savybei φ, sakoma kad φ turi tapatumo kriteriju. Čia tisklas rasti tokį ryšį, kuris visiems savybės egzemplioriams bus prilygstantis tapatumui. Informacinėse sistemose tam naudojami dirbtinės arba išorinės savybės, tačiau ontologijoje tai pats svarbiausias klausimas plečiant esybių klasių egzistavimą: kaip mes galime sužinoti, kad esybė egzistuoja, kaip ji gali būti identifikuojama?

Svarbu atskirti tapatumo kriterijaus *turėjimą* (angl. *carrying an IC*) ir tapatumo kriterijaus *tiekimą* (*supplying an IC*). Pavyzdžiui, negriežtos savybės, tokios kaip STUDENTAS, gali tik turėti (kitų esybių joms suteiktą) tapatumo kriterijų, tuo tarpu griežtos, tokios kaip ŽMOGUS, savybės gali tiekti jį (ir tuo pačiu turėti).

Yra dvi susietos su laiku tapatumo kriterijaus rūšys: diachroninis ir sinchroninis tapatumo kriterijus [2]. Diachroninis tapatumo kriterijus nusako pvz., kaip nustatyti, ar individas po tam tikro laiko yra tas pats individas? Sinchroninis tapatumo kriterijus atsako į klausimą kaip atskirti vieną individą nuo kito tam tikru laiko momentu.

Kartais būna sunku nustatyti ar savybė turi tapatumo kriterijų, kadangi reikia rasti tokį, kuris yra būtinas IR pakankamas [10]. Ypač tai sunku nustatyti gamtos rūšims ir artefaktams. Todėl buvo įvestas sekantis apibrėžimas.

Apibrėžimas 4. *Tapatumo kriterijumi* vadinama tapatybės formulė Σ kuri tenkina (3) ir (4), pašalinus visus trivialius atvejus. Čia E - tai faktinis buvimas (skyrelis 2.1.3) [10]:

$$\Box(E(x,t) \wedge \phi(x,t) \wedge E(y,t') \wedge \phi(y,t') \wedge x = y \rightarrow \sum(x,y,t,t')), \quad (3)$$

$$\Box(E(x,t) \wedge \phi(x,t) \wedge E(y,t') \wedge \phi(y,t') \wedge \sum(x,y,t,t') \rightarrow x = y). \quad (4)$$

Formule (3) norima pasakyti, kad jeigu yra du objektai x ir y egzistuojantys laiko momentais t ir t‘, abu turintys savybę φ, ir x = y, reiškia x ir y turi tapatumo kriterijų. Tapatumo kriterijus yra būtinas, jeigu jis tenkina (3) lygybę ir pakankamas, jeigu tenkina (4), ir nebūtinai turi tenkinti abi lygybes. Šis apibrėžimas yra silpnesnis negu (2) lygybė, tačiau jis daugiau tinka informacinems sistemoms.

Yra daug galimų trivialų formuluočių, kurios tenkina šias lygybes, tačiau jos nėra informatyvios. Taigi tinkama tapatumo kriterijus tai toks kriterijus, kuris yra informatyvus.

Remiantis šiuo apibrėžimu buvo apibrėžtos šios meta-savybės [10]:

Apibrėžimas 5. Sakoma, kad savybė *turi* tapatumo kriterijų, jeigu ją apima sąvybė, *tiekianti* tapatumo kriterijų (įskaitant atvejį, kai savybė pati tieka tapatumo kriterijų).

Apibrėžimas 6. Sakoma, kad savybė φ *tiekia* tapatumo kriterijų, jeigu (1) ji yra griežta, (2) jai egzistuoja tapatumo kriterijus; (3) šį tapatumo kriterijų turi visos savybės, kurios apima φ.

Apibrėžimas 7. Bet kokia savybė, kuri turi tapatumo kriterijų vadinama *rūšimi.*

Savybė, kuri *turi* tapatumo kriterijų žymima +I (-I, jei neturi). Savybė, kuri *tiekia* tapatumo kriterijų žymima kaip +O(-O priešingu ateju). Iš aukščiau pateiktų apibrėžimų matosi, kad jeigu savybė tiekia tapatumo kriterijų (+O) ji taip pat turi jį (+I) ir yra griežta (+R), pavyzdiui, savybės ŽMOGUS ir STUDENTAS abu turi tapatumo kriterijus, t.y. jie yra +I, tačiau tik ŽMOGUS tiekia tapatumo kriterijų (+O). Savybė RAUDONAS neturi tapatumo kriterijaus, nes nėra tokios būtinos ir pakankamos formulės, kuri leidžia identifikuoti raudonus daiktus tik todėl, kad jie yra raudoni.

1.2.3 Vienybė

Su vienybe siejamas klausimas kaip atskirti objektą nuo kitų objektų, t.y. kaip aprašyti objekto ribas ar dalis, kad galima būtų aiškiai pasakyti kas yra jo dalis o kas ne. Su vienybe siejama pilnumo sąvoka.

Apibrėrimas 8. Objektas x vadinamas *pilnuoju* naudojant ryšį ω, jeigu ω yra toks kad sujungia visas objekto x dalis ir daugiau nieko [10].

Lentelėje pateikiamos formulės, kurios aprašo dalinį ryšį. Čia *P(x,y,t)* reiškia, kad x yra (tinkama ar netinkama angl. *proper/improper*) y dalis laiku t, o žodis *def* reiškia dalies apibrėžimą.

$PP(x,y,t) =_{def} p(x,y,t) \wedge \neg x = y$	Tinkama dalis
$O(x,y,t) =_{def} \exists z(P(z,x,t) \wedge P(z,y,t))$	Uždengimas
$P(x,y,t) \wedge P(y,x,t) \rightarrow x = y$	Anti-simetrija
$P(x,y,t) \wedge P(y,z,t) \rightarrow P(x,z,t)$	Tranzityvumas
$PP(x,y,t) \rightarrow \exists z(PP(z,y,t) \wedge \neg O(z,x,t))$	Silpnas papildymas

1 lentelė. Dalinio ryšio aksiomos [12].

Apibrėžimas 9. Sakoma, kas savybė φ *įgyja vienybės kriterijų* jeigu egzistuoja toks ryšys ω, kad kiekvienas φ egzempliorius, kurio dalys surištos ryšiu ω yra *būtinai* pilnas.

Priklausomai nuo ontologinės ryšio ω prigimties galima išskirti tris pagrindines vienybės rūšis [10]:

- Topologinė vienybė. Tai topologinis ar fizinis sujungimas, pavyzdžiui ryšys tarp kreidos ar obuolio gabaliukų.
- Morfologinė vienybė. Tai topologinės vienybės ir formos kombinacija, pavyzdžiui, kamuolys arba tai gali būti morfologinis ryšys tarp „pilnųjų“ objektų, pavyzdžiui žvaigždynas.
- Funkcinė vienybė. Tai topologinio ir morfologinio ryšių kombinacija su tam tikra paskirtimi, pavydžiui, plaktukas. Arba funkcinis ryšys tarp pilnųjų objektų, pavyzdžiui, maudymosi kostiumėlis iš diejų dalių.

Taigi, kaip rodo pavyzdžiai, „pilnasis“ objektas gali būti sudarytas iš dalių, kurios taip pat yra „pilnosios“.

Apibrėžimas 10. Savybė įgyja anti-vienybę, jeigu kiekvienas jos egzempliorius nebūtinai yra pilnasis. (Pavyzdžiui, vanduo).

Savybė, turinti vienybės kriterijų žymima **+U**(**-U** priešingu atveju). Savybė, turinti anti-vienybę žymima **~U**. Iš čia matosi, kad **~U** reiškia ir **–U**.

1.2.4 Priklausomybė

Apibrėžimas 11. Sakoma, kad savybė φ yra išoriškai priklausoma nuo savybės ψ jeigu, kiekvienam jos egzemplioriui x būtinai egzistuoja ψ egzempliorius, kuris nėra nei x dalis, nei x yra substancija, iš kurios jis sudarytas [10]:

$$\forall x\Box(\phi(x) \rightarrow \exists y\psi(y) \wedge \neg P(y,x) \wedge \neg C(y,x)) \quad (5)$$

Funkcija C reiškia konstitucija, kas reiškia, kad y sudarytas iš x. Pavyzdžiui, pilis pastatyta iš plytų, statula padaryta iš marmuro. Konstitucija dažnai supainiojama su apėmimu (angl. *subsumption*).

Jeigu neišskirsime dalis ir substancijas lygybėje (5), tada gaunasi kad visos savybės, žyminčios tam tikrų esybių klases yra priklausomos, nes visos neatominės esybės sudarytos iš tam tikros medžiagos, t.y. mažiausiu nedalomu medžiagos vienetu laikomas atomas. Taip pat reikėtų išskirti kokybės metrikas (pvz., spalvas), dalykus kurie būtinai egzistuoja (pvz., visata) ir atvejus, kai φ apima ψ (tada gaunasi kad φ priklauso nuo savęs). Sakome, kad pvz., TĖVAS išoriškai priklauso nuo VAIKO (asmuo negali būti tėvu neturėdamas vaikų), bet ŽMOGUS negali būti išoriškai priklausomas su širdies ar kūno, nes kiekvienas žmogus turi širdį ir žmogų sudaro kūnas.

Išoriškai priklausoma savybė žymima kaip +D(-D priešingu atveju).

1.2.5 Apribojimai ir prielaidos

Apėmimo ryšys turi tam tikrus apribojimus. Jeigu *p* ir *q* yra dvi savybės, ir *q apima p*, tada galioja tokie apribojimai [2]:

1. Jeigu *q* yra anti-griežta savybė (~R), tada *p* taip pat turi būti anti-griežta;
2. Jeigu *q* įgyja tapatumo kriterijų (+I), tada *p* turi tą patį tapatumo kriterijų;
3. Jeigu *q* įgyja vienybės kriterijų (+U), tada *p* turi tą patį vienybės kriterijų;
4. Jeigu *q* įgyja anti-vienybės kriterijų (~U), tada *p* taip pat įgyja anti-vienybės kriterijų.

Tapatumo savybei galioja sekančios prielaidos [10]:

- Rūšių individualizacija (angl. *sortal individuation*) – kiekvienas dalykinės srieties elementas privalo turėti savybę, kuri įgyja tapatumo kriterijų. Kitais žodžiais turi būti tenkinamas Quine‘o sakinys: „Negali būti esybės, neturinčios tapatumo kriterijaus“.

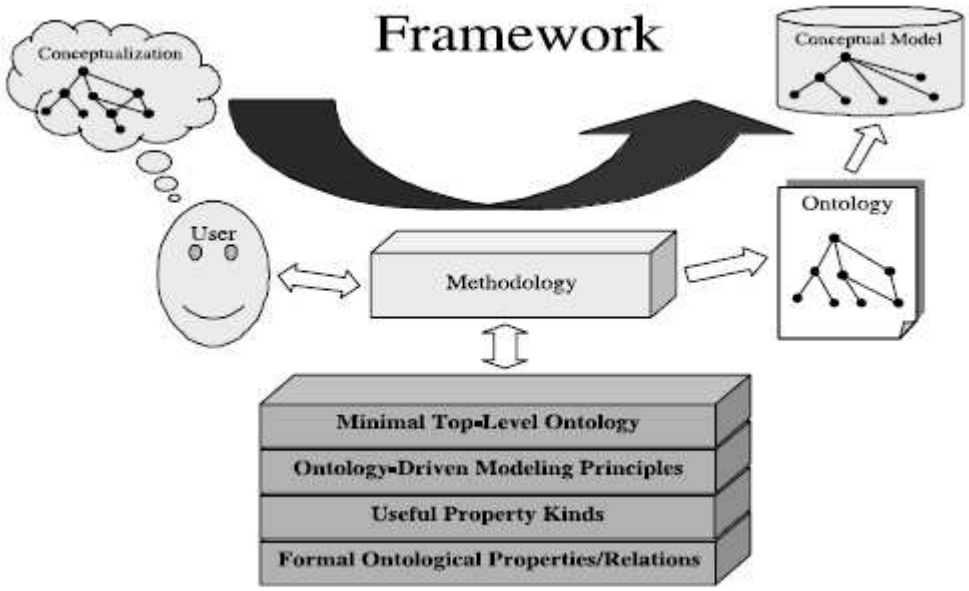
- Rūšių išplečiamumas (angl. *sortal expandability*) – jeigu kazkas yra skirtingų savybių egzempliorius, tada šitas kažkas turi būti bendresnės savybės, turinčios tapatumo kriterijų, egzmepliorius.

Kartu šios dvi prielaidos reiškia, kad kiekviena esybė turi atstovauti unikaliai savybei, turinčiai tapatumo kriterijų.

1.3 Metodikos aprašymas

OntoClean tikslas yra padėti sudaryti *pagrįstus* modelius, t.y. tokius modelius, kurių ryšiai tarp elementų neprieštarauja aukščiau išvardintiems apribojimams.

Kaip parodyta paveikslėlyje, projektuotojas naudoja OntoClean metodiką projektuodamas koncepcinį modelį. Naudodamas dalykinės srities konceptualizaciją ir metodiką jis sukuria ontologinį medį, iš kurio vėliau sudaromas koncepcinis modelis.



2 pav. Metodikos OntoClean apžvalga[10].

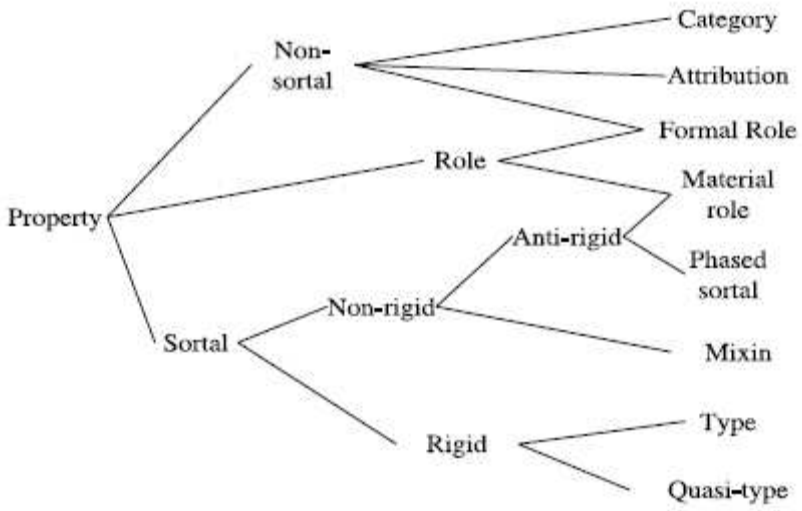
Taigi, metodiką OntoClean sudaro keturi sluoksniai. Kiekvienas sluoksnis naudojasi informacija, gauta iš žemesnių sluoksnių.

Pirmasis sluoksnis - tai metodikos pagrindinės savokos – meta-savybės aprašytos ankstesniuose skyriuose. Jos atitinka aksiomų schemas modeliavimo kalbose.

Antras sluoksnis apima visas galimas meta-savybių kombinacijas, t.y. visus galimus tipus, kurie pavaizduoti 3 ir 4 paveikslėliuose.

+O	+I	+R	+D	Tipas	Rūšis
			-D		
-O	+I	+R	+D	Kvazi-tipas	
			-D		
-O	+I	~R	+D	Materiali rolė	
-O	+I	~R	-D	Fazinė rūšis	
-O	+I	¬R	+D	Mixin	
			-D		
-O	-I	+R	+D	Kategorija	Ne-rūšis
			-D		
-O	-I	~R	+D	Formali rolė	
		¬R	+D	Savybė (angl. *attribution*)	
			-D		
+O	-I			nerišlūs	
	+I	~R			
		¬R			

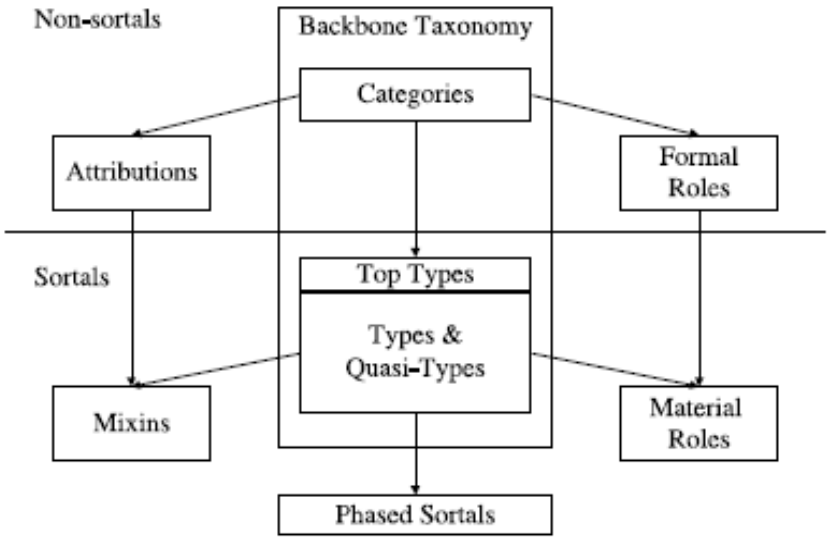
2 lentelė. Visos galimos meta-savybių kombinacijos



3 pav. Savybių sistematika [15].

Kiekvienam dalykinės srities elementui projektuotojas turi nustatyti, kokias meta-savybes (I, O, U, R, D) elementas turi.

Trečiasis sluoksnis apima pagrindinės taksonomijos arba sistematikos (angl. *backbone taxonomy*) ir sluoksniavimo (*stratification*) sąvokas. *Pagrindinę taksonomiją* sudaro trys tipai: kategorijos, tipai ir kvazi-tipai. Kategorijos negali būti apimtos kitų tipų savybėmis, t.y. pati bendriausia savybė, ir todėl yra aukščiausias lygis taksonomijoje. Taigi, pagrindinė taksonomija yra dalykinės srities bazinė struktūra. Paveiksle parodytas idealus atvejis kaip ontologijos turėtų būti struktūrizuotos.



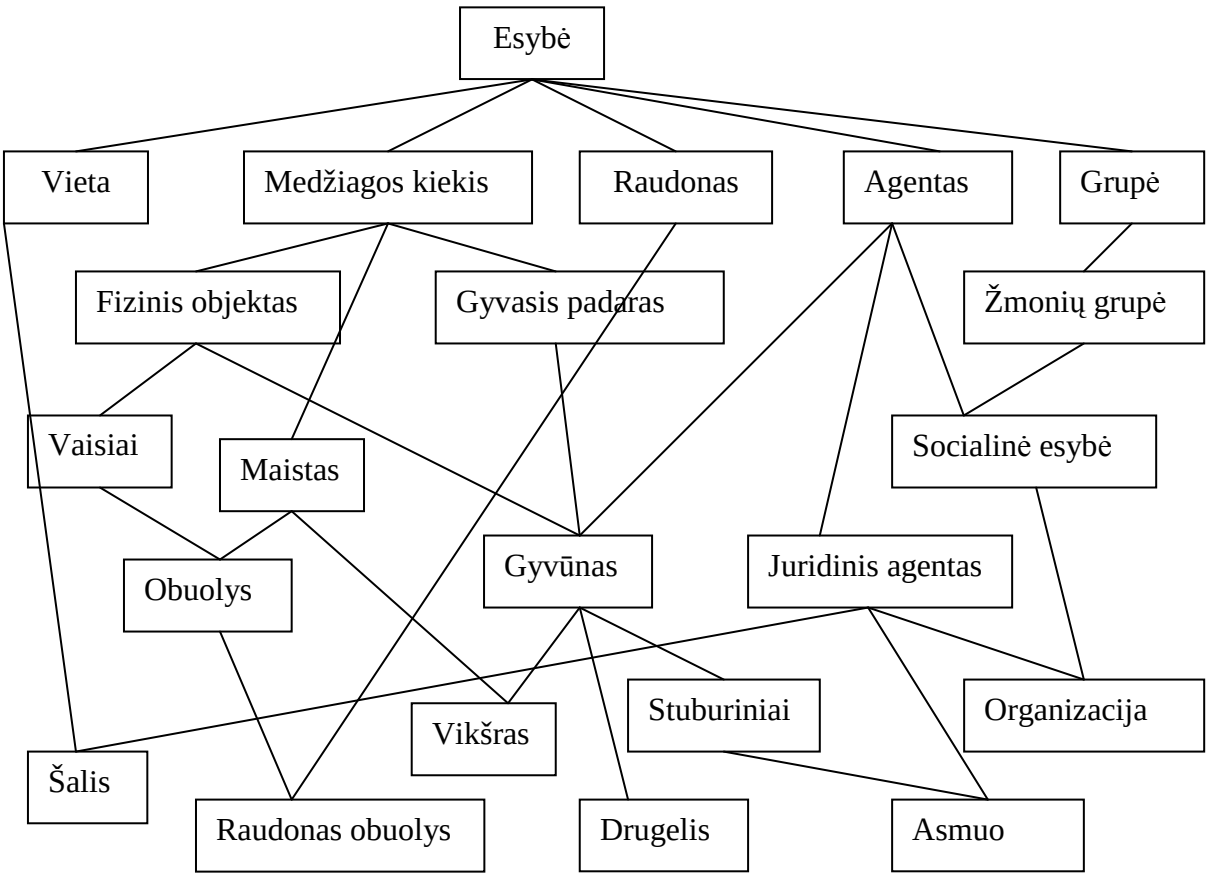
4 pav. Ideali taksonomijos struktūra [15].

Sluoksniavimas. Tai kelių savybių, turinčių skirtingus tapatumo ir vienybės kriterijus, išskyrimas iš vienos savybės [10]. Pavyzdžiui, statula ir molis, iš kurio jinai sulipdyta yra du skirtingi objektai. Sluoskniavimo rezultate gauname ontologija, kurioje savybės priskiriamos skirtingiems sluoksniams priklausomai nuo jų tapatumo ir vienybės kriterijų.

Ketvirtas sluoksnis – tai ontologijos medžio sudarymas naudojant metodus ir technikas, aprašytas žemesnuose sluoksniuose.

1.3.1 Pavyzdys

Šiame skyriuje bus parodyta, kaip gali būti naudojama OntoClean. Tarkime, turime žemiau pavaizduotą taksonomiją. Ją OntoClean pagalba pertvarkysime į ontologiškai teisingą taksonomiją.



5 pav. Nesutvarkyta taksonomija [2].

Pirmiausia, kiekvienai savybei turi būti priskirtos meta-savybės I,U,R ir D. Nustatoma, ar savybė įgyja/neįgyja arba tiekia tapatumo/vienybės kriterijų, ar ji yra priklausoma, ar ji (ne-)griežta arba anti-griežta. Kartais būna nelengva nustatyti, ar savybė turi tapatumo kriterijų, nes sunku rasti tokį požymį, kuris būtų vienu metu būtinas ir pakankamas, kad padėtų atskirti vieną savybės egzempliorių nuo kito.

Savybė	**Meta-savybės**	**Tipas**
Esybė	-I+U-D+R	Kategorija
Vieta	+O~U-D+R	Tipas
Medžiagos kiekis	+O~U-D+R	Tipas
Raudonas	-I-U-D-R	Formali savybė
Agentas	-I-U+D~R	Formali rolė
Grupė	+O~U-D+R	Tipas
Fizinis objektas	+O+U-D+R	Tipas
Gyvasis padaras	+O+U-D+R	Tipas
Žmonių grupė	+I-O~U-D+R	Kvazi-tipas

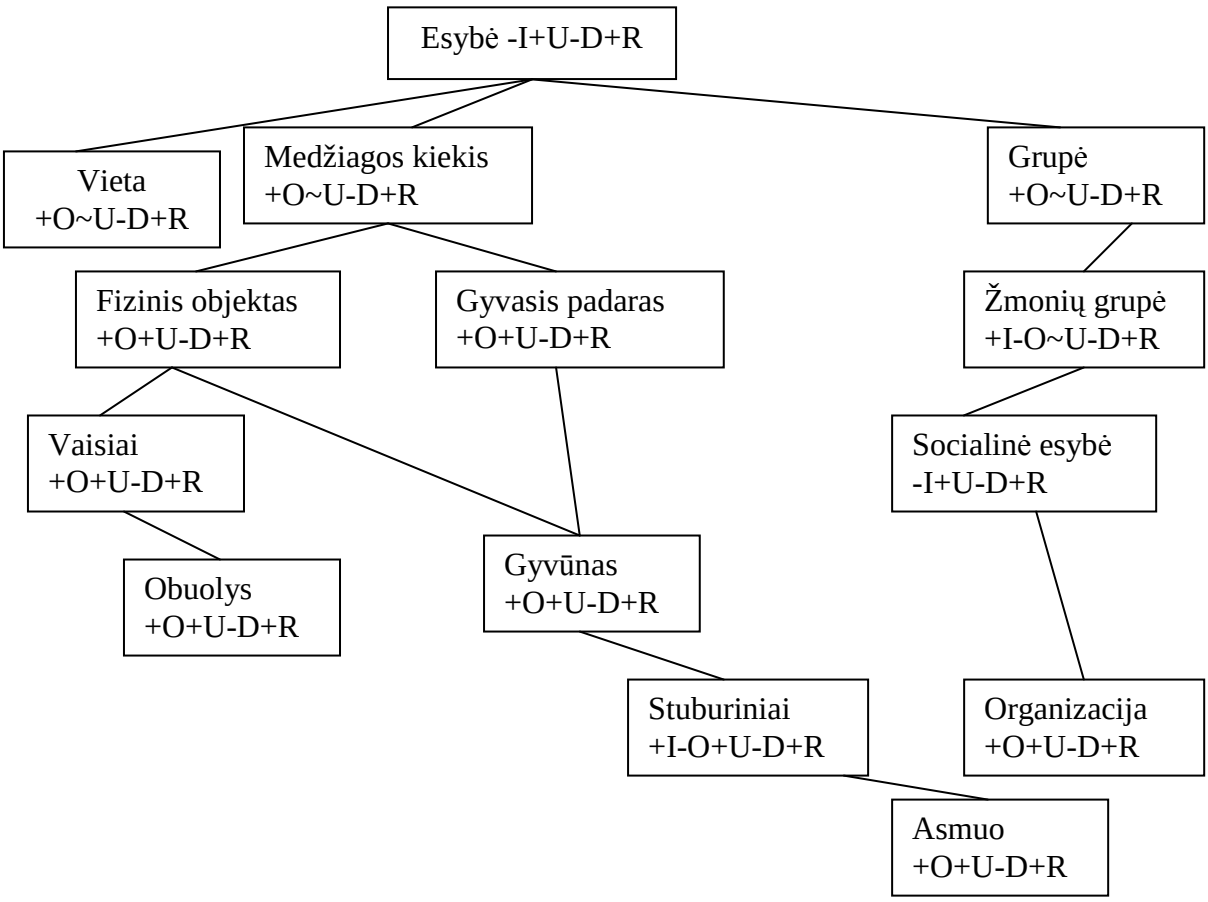
Socialinė esybė	-I+U-D+R	Kategorija
Vaisiai	+O+U-D+R	Tipas
Maistas	+I-O~U+D~R	Materiali rolė
Gyvūnas	+O+U-D+R	Tipas
Juridinis agentas	+I-U+D~R	Materiali rolė
Obuolys	+O+U-D+R	Tipas
Šalis	+I+U-D~R	Fazinė rūšis
Raudonas obuolys	+I-O-D~R	Mixin
Vikšras	+I+U-D~R	Fazinė rūšis
Drugelis	+I+U-D~R	Fazinė rūšis
Stuburinis	+I-O+U-D+R	Kvazi-tipas
Organizacija	+O+U-D+R	Tipas
Asmuo	+O+U-D+R	Tipas

3 lentelė. Meta-savybės priskirtos savybėms [10].

Toliau diagramoje pašalinamos visos negriežtos savybės, t.y. paliekama tik pagrindinė taksonomija, kuria sudaro visos griežtos savybės: kategorijos, tipai ir kvazi-tipai. Tada analizuojama, kokie yra ryšių pažeidimai.

Visų pirma, gyvasis padaras nėra medžiagos kiekis, nes pagal taisyklę savybė, turinti ~U negali apimti savybės, turinčios +U. Tas pats su fiziniais objektais ir medžiagos kiekiu, bei žmonių grupėmis ir socialinėmis esybėmis.

Dar vienas pažeidimas pagal OntoClean yra tai, kad gyvūnai nėra fiziniai objektai. Nors čia nėra jokiu pažeidimu, tačiau šių savybių tapatumo kriterijai turi skirtingą prigimtį. Gyvūno esminė savybė yra ‚būti gyvam', t.y. gyvūnas nustoja egzistuoti kai jis miršta. Tačiau jo kūnas dar kurį laiką egzistuoja, todėl tai nėra fizinio objekto esminė savybė. Jeigu gyvūnas yra fizinis objektas, kaip parodo apėmimo ryšys, gaunasi, kaip jis vienu metu yra „būtinai" gyvas ir „nebūtinai" gyvas. Taip būti negali, todėl apėmimo ryšys tarp šių dviejų savybių turi būti pašalintas.



6 pav. Pagrindinė nesutvarkyta taksonomija.

Kai visi ryšių pažeidimai tarp griežtų savybių pašalinti, analizuojamos negriežtos savybės.

1 pažeidimas. Drugeliai ir vikšrai. Nors gyvūnai iš tikrųjų apima drugelius ir vikšrus, kurie yra vienos esybės fazės. Todėl į taksonomiją reikia įtraukti dar vieną savybę, kuri apimtų tik drugelius ir vikšrus, t.y. tos pačios esybės fazes.

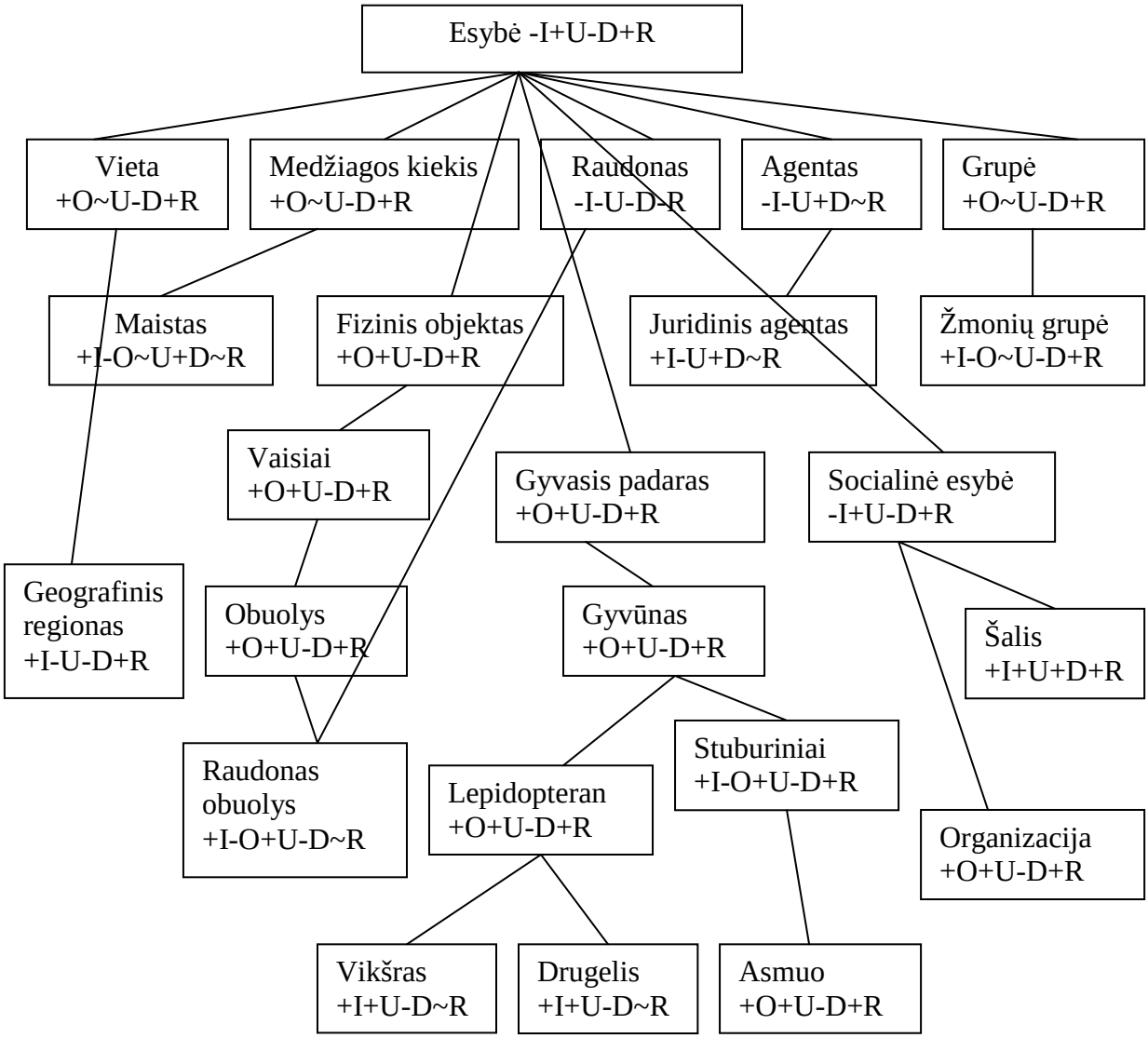
2 pažeidimas. Šalys. Šaliai buvo priskirta +O~R, tačiau kas konkrečiai turima omeny? Ar tai regionas, kuris laikinai tampa šalimi ir kas atsitinka, kai šalies daugiau nebėra? Ar ji nustoja egzistuoti, ar tiesiog pasikeičia savybę? Šiuo atveju viena esybė apima kelias reikšmes, kurias reikia atskirti. Jeigu šalis nustoja egzistuoti kaip geografinis regionas, ji vis dar gali egzistuoti kaip geopolitinė esybė. Todėl šalį reikia išskirti į dvi esybes: geografinį regioną (+I-U-D+R) ir šalį (+O+U+D+R). Šalis įgyja savo unikalų tapatumo kriterijų, tuo tarpu geografinis regionas paveldi tapatumo kriterijų ir vietos.

3 pažeidimas. Agentai. Pagal taisyklę savybė, kuri yra anti-griežta negali apimti savybės, kuri yra griežta. Gaunasi, kad gyvūnas ir socialinė esybė privalo būti agentas, nors iš tikrųjų neprivalo.

4 pažeidimas. Juridiniai agentai. Kadangi legalus agentas paveldi iš agento anti-griežtumą, jis taip pat negali apimti griežtų savybių. Todėl apėmimo ryšys tarp legalaus agento juo apimtu griežtų savybių „asmuo“ ir „organizacija“ būti pašalintas.

5 pažeidimas. Maistas. Apėmimo ryšį tarp savybės „maistas“ ir savybių „obuolys“ ir „vikšras“ reikia pašalinti, nes maistas yra rolė, o rolės negali apimti tipų ir fazinių rūšių.

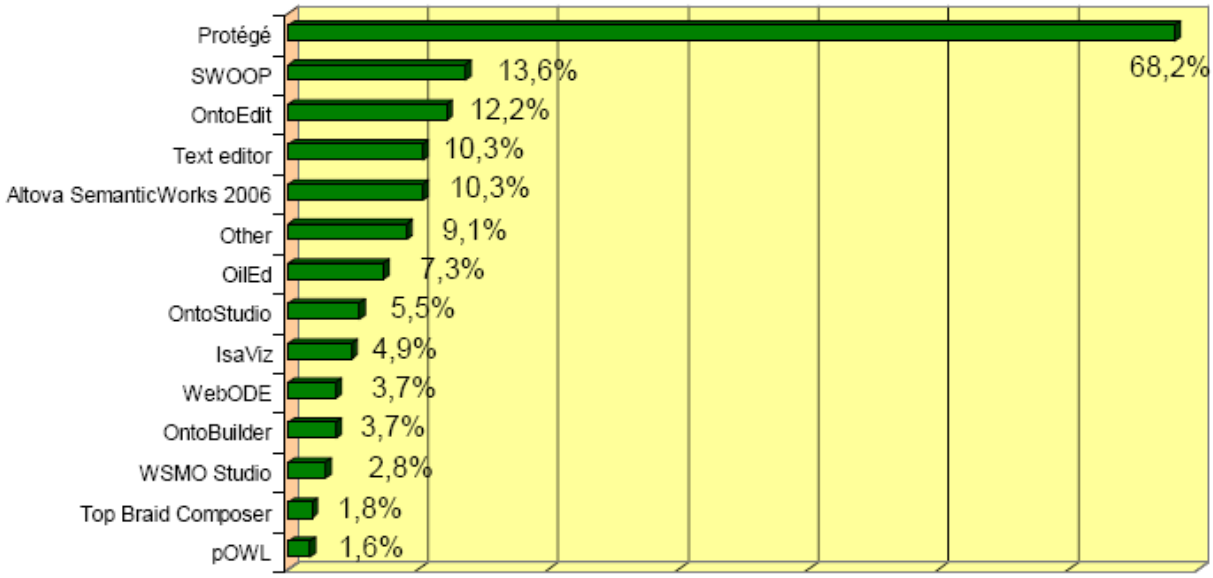
Išanalizavus visus apėmimo ryšius ir jų pažeidimo atvejus, savybės pertvarkomos į galutinę ontologiją.



7 pav. Galutinė švari ontologija [2].

2. Įrankių apžvalga

Šiuo metu pasaulyje yra nemažai įrankių, skirtų ontologijų kūrimui. Vieni įrankiai apima tik vieną ontologijų kūrimo stadiją, pavyzdžiui ontologijų kūrimą, kiti – kelias stadijas, pavyzdžiui, specifikavimą, kūrimą ir vertinimą. Žemiau parodytas įrankių panaudojamumas 2007 metų pabaigoje:



8 pav. Įrankių panaudojamumas [18].

Labiausiai naudojami yra Protégé, SWOOP ir OntoEdit. Visi šitie įrankiai palaiko OntoClean, kas parodo šios metodikos svarbą ontologijų kūrimo procese. Kaip gi realizuotas ontologijų vertinimas šituose įrankiuose?

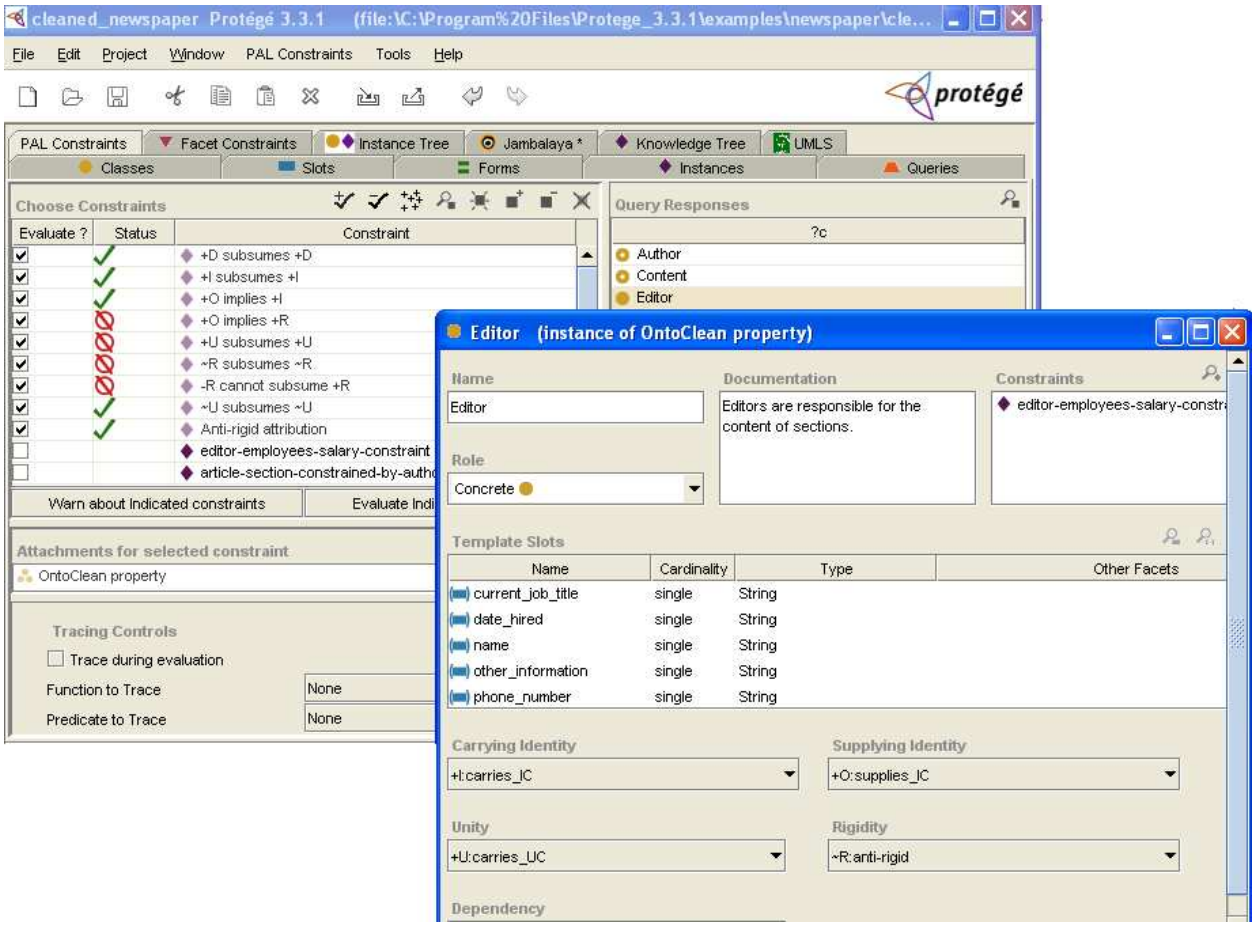
2.1 Protégé

Protégé [19] – tai atviro kodo ontologijų kūrimo įrankis. Jį sukūrė Stenfordo universitetas bendradarbiaudamas su Mančesterio universitetu. Protégé turi praplėtimų architektūrą, tai reiškia kad jo funkcionalumas gali būti praplečiamas pridedant naujus plug-in‘us. Protege gali naudotis ir kitos programos, norinčios pasinaudoti jos žinių baze.

Skirsnyje „Classes“ vartotojas gali aprašyti klases (koncepcijas) ir klasių hierarchiją. Klasė turi tipą: abstraktų arba konkretų, ir gali turėti apribojimus. Skirtuke „Slots“ įvedamos klasės savybės ir ryšiai tarp klasių. Kitas skirtukas „Intances“ skirtas klasių objektų ir jų savybių aprašymui. Protege ontologija vaizduojama kaip direktorijų hierarchija Windows op. sistemoje, tačiau praplėtimų pagalba, pavyzdžiui, su Jambalaya ją galima atvaizduoti į diagramą.

Pagrindinė Protege prielaida yra ta, kad ontologijų kūrimas reikalauja daug žinių, patirties ir pastangų. Ontologijas gali kurti tik dalykinės srities ekspertai ar programuotojai. Todėl Protege tikslas yra vadovauti ekspertams ir programuotojams sistemos kūrimo procese. Protege leidžia pakartotinai naudoti ontologijas ir problemų sprendimo metodus, ir tokių būdu sumažina ontologijų kūrimo ir palaikymo laiką.

Protege OntoClean metodas naudojamas kaip meta-klasė, kuri gali būti priskiriama kiekvienai ontologijos klasei. Priskirus klasei meta-klasę OntoClean Poperty, ši klasė įgauna papildomų savybių I, O, U, R, D ir apribojimų, kurie remiasi šiomis savybėmis (žr. 2 skyrių). Tada, naudojant Protege aksiomų kalbos plug-in‘ą (PAL Constraints) ontologija gali būti įvertinta jos apribojimų atžvilgiu. Paspaudus „Evaluate selected constrants“ patikrinami visi pažymėti apribojimai. Pasirinkus apribojimą, turintį jį netenkinančių klasių pateikiamas tų klasių sąrašas. Čia pat, pasirinkus klasę, gali būti atliekamas meta-savybių koregavimas.



9 pav. Ontologijų vertinimas su Protege

Pagrindinės savybės:

- ✓ Import format

- XML, RDF(S) and XML Schema

✓ Export format
- XML, RDF(S), XML Schema, FLogic, CLIPS and Java HTML

✓ Dviejų ir daugiau vartotojų palaikymas
- Kliento-serverio režimas leidžia daugeliui vartotojams dirbti su ta pačia ontologiją bei keisti tuos pačius duomenis.

✓ OWL palaikymas
- Per Protégé-OWL praplėtimą

2.2 OntoEdit

OntoEdit leidžia kurti ir palaikyti ontologijas panaudojant grafines priemones. Interfeisas palaiko daug kalbų. OntoEdit buvo sukurtas 2002 metais ir buvo nemokamas. Neseniai jis tapo komercinių produktų (dabar vadinamas OntoStudio), tačiau jo nemokama versija vis dar plačiai naudojama (trečia pagal vartotojų panaudojamumą). OntoEdit remiasi W3C standartais ir gali transformuoti ontologiją į pačias žinomiausias ontologijų pavaizdavimo kalbas: RDF(S), DAML+OIL, F-Logic, XML.

OntoEdit apima 3 ontologijų gamybos stadijas: specifikavimą, kūrimą ir vertinimą.

Ontologijų specifikavimą palaiko praplėtimai OntoKick ir Mind2Onto5. OntoKick padeda aprašyti svarbius ontologijos aspektus: dalykinę sritį ir tikslus, pasiekiamus žinių šaltinius (dalykinės srities ekspertus, ontologijas, kurios gali būti pakartotinai panaudojamos), ontologijos vartotojus, naudojimo scenarijus ir programas, kurios naudos ontologiją. Mind2Onto5 vadovauja brainstorming'ui ir diskusijoms dėl ontologijos struktūros.

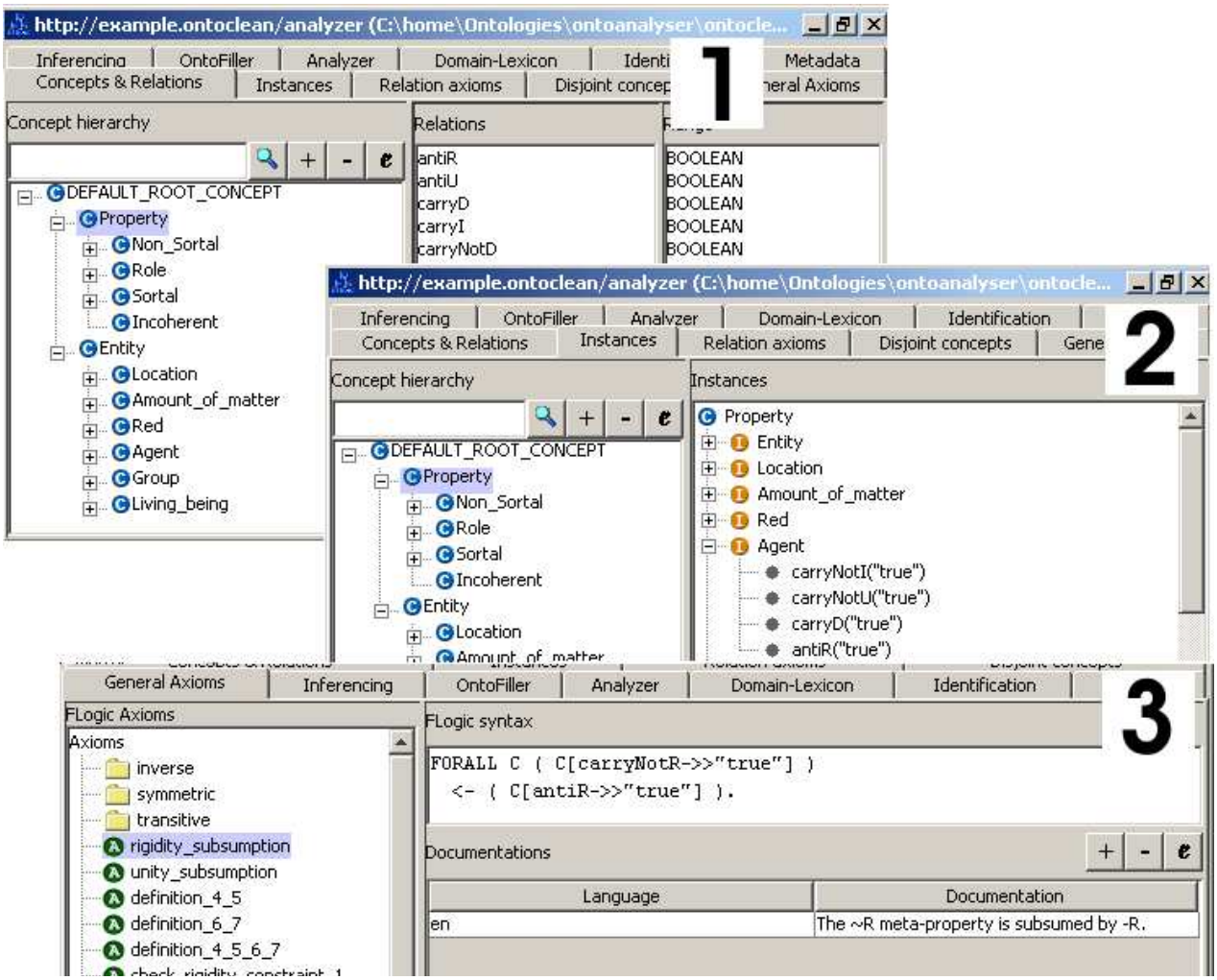
Ontologijų kūrimo stadijoje ontologija formalizuojama taip kad ji galėtų būti pritaikyta programose. Vartotojas gali sukurti ontologijos klasių hierarchiją nepriklausomai nuo konkrečios ontologijų pavaizdavimo kalbos. Koncepcinio modelio saugojimui OntoEdit naudoja galingą ontologijos modelį.

OntoEdit pagalba galima redaguoti koncepcijų ir klasių hierarchijas. Ontologijos modeliuojamos kaip „is-a" ryšių hierarchija, kas reiškia, kad poklasis turi visas tėvinės klasės savybes. Koncepcija gali turėti kelis vardus. Kiekviena koncepcija, klasė ir jos atributas yra dokumentuojami. Tai yra ypač svarbu vykstant ontologijų apsikeitimui. Palaikomas ontologijos konfigūravimas: išsaugoma paskutinio modifikavimo data ir autorius. Visi ontologijos pakeitimai yra sekami. Dar vienas įrankio privalumas yra tas, kad vartotojas gali dirbti vienu metu su keliomis ontologijomis.

Ontologijos vertinimas pagal OntoClean yra panašus į vertinimą Protege. Jis susideda iš trijų žingsnių (10 pav.):

1. Sukurti ontologijas
2. Priskirti koncepcijoms meta-savybes (pvz., carryR (+R) ir t.t.)
3. Aprašyti OntoClean apribojimus kaip aksiomas (F-Logic kalba)

Padaręs visus 3 žingsnius vartotojas gali rašyti užklausas, kad rasti aksiomų pažeidimus ontologijoje.



10 pav. Ontologijų vertinimas su OntoEdit [20].

- Pagrindinės savybės :
  - ✓ Importo/eksporto formatas
    - o XML, RDF(S), F-Logic ir DAML+OIL (nemokama versija)
    - o XML, RDF(S), F-Logic, DAML+OIL ir SQL-3 (tik eksportas) komercinei versijai

- ✓ Palaiko informacijos atvaizdavimą grafikais ir diagramomis
- ✓ Suderinamumo tikrinimas
- ✓ Web palaikymas
  - o URI

- Papildomos savybės :
  - ✓ Ontologijos saugojimas
    - o File (nemokama) vs. File and DBMS (komercinė)
  - ✓ Samprotaujantis variklis (angl. *inference engine*)
    - o Nepalaiko (nemokama) vs. palaiko per OntoBroker (komercinė)
  - ✓ Daug vartotojų ir Ontologijų biblioteka
    - o Nepalaiko (nemokama) vs. palaiko (komercinė)

2.3 Išvados

Ontologijų vertinimą pagal OntoClean abu įrankiai Protege ir OntoEdit daro panašiai. Vertinimas su OntoEdit reikalauja daugiau žinių, kadangi ten OntoClean aksiomas vartotojas turi apibrėžti pats su F-Logic kalba. Iš kitos pusės, Protege šios aksiomos įtraukiamos į projektą pagal nutylėjimą (po meta-klasės OntoClean priskyrimo klasėms), todėl vartotojas turi tik varnele pažymėti, kokius apribojimus jis nori patikrinti.

Ontologijos koregavimas irgi nėra labai patogus. Diagramoje meta-savybės nėra parodomos. Nepalaikomas objektų perkėlimas drag-and-drop principu. Tai sukelia tam tikrų nepatogumu, nes jeigu vartotojas nori pakeisti klasės vietą hierarchijoje, jis turi ištrinti ir iš naujo ją sukurti.

Galu gale, didžiausias šių įrankių trukumas yra tas, kad sudetingiausia vertinimo dalis – meta-savybių priskyrimas klasėms – lieka neautomatizuota. Todėl vertinimą priversti daryti tik dalykinės srities ekspertai. Be to, kaip rodo eksperimentai [17], ekspertų nuomonės dažnai nesutampa. Šios veiklos pilnas ar dalinis automatizavimas būtų didelis šuolis į priekį ontologijų vertinimo procese.

3. Automatinis ontologijų vertinimas

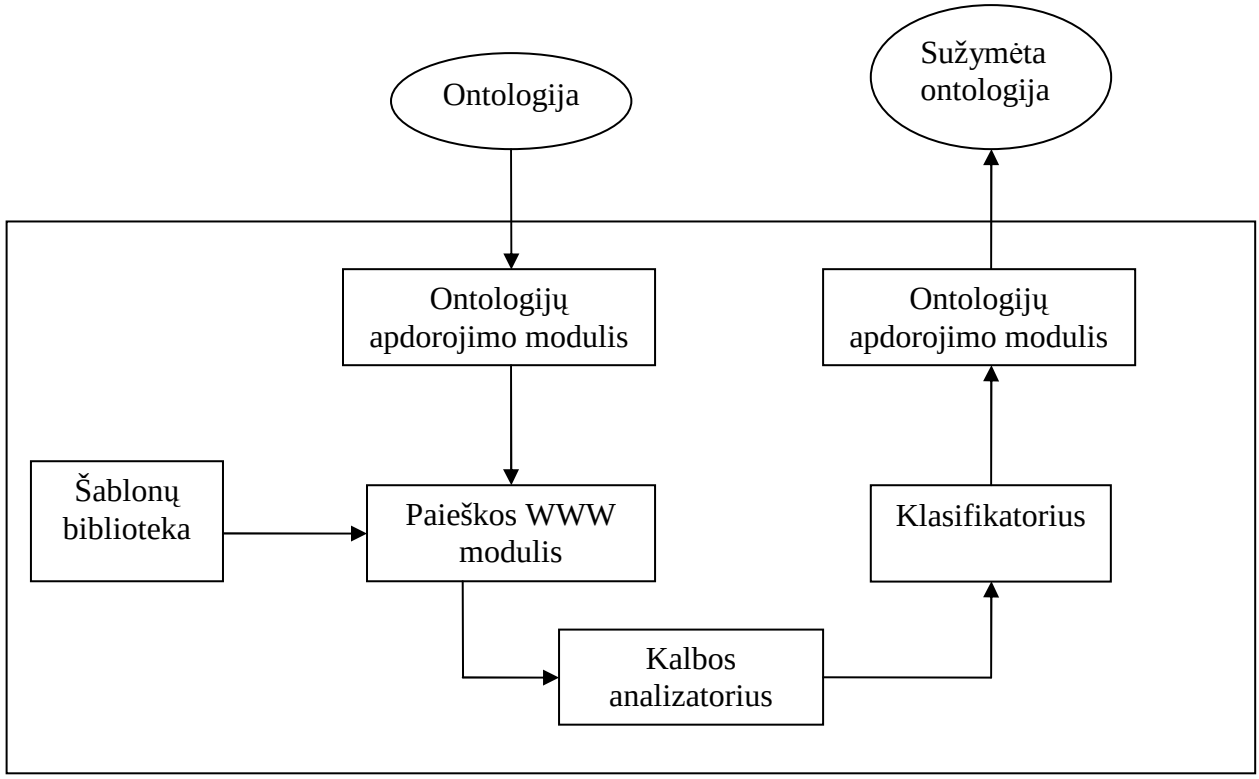
OntoClean metodikos taikymas, ypač meta-savybių priskyrimas koncepcijoms, nemažai kainuoja, todėl ši metodika retai taikoma. Be to, eksperimentų metu [17] buvo nustatyta kas pats meta-savybių priskyrimo procesas yra gana sudėtingas, t.y. vertintojų nuomonės gali stipriai skirtis. Todėl bent dalinis šio proceso automatizavimas labai praverstų.

3.1 Idėja

Automatinio ontologijų vertinimo metodas remiasi dvejomis prielaidomis, aprašytomis [17, 24]. Pirmoji prielaida yra ta, kad meta-savybių prigimtis kažkiek atsispindi žmogaus kalboje ir kas apie meta-savybes yra parašyta informacijos šaltiniuose! Todėl galima laikyti, kad paieškos rezultatų pagal leksinius-sintaksinius šablonus statistika yra priemonė gauti meta-savybės teigiamą arba neigiamą įrodymą. Kita prielaida yra ta, kad įrodymų paieškai gali būti naudojamas internetas, kadangi jis yra pats didžiausias informacijos šaltinis. Tai padeda susidoroti su atvejais, kai trūksta informacijos apie retai pasitaikantį žodį. Įrodymų paieškai bus naudojami paieškos frazių šablonai [17].

3.2 Algoritmo aprašymas

Remiantis aukčiau aprašyta idėja, aš sukūriau algoritmo, automatizuojančio ontologijų vertinimą, modelį. Modelio architektūrą sudaro penki moduliai: Ontologijų analizatorius, paieškos WWW modulis, kalbos analizatorius ir klasifikatorius. Kiekvienas modulis turi tam tikrą paskirtį. Ontologijų analizatorius skirtas darbui su ontologijomis, t.y. ontologijų skaitymui ir sukūrimui. WWW paieškos modulis panaudodamas šablonus, užklausia WWW apie dalykinės srities sąvokų meta-savybių įrodymus. Šablonų biblioteką sudaro paieškos frazės, tokios kaip ‚is no longer x', kurios reikalingos meta-savybių teigiamiems ir neigiamiems įrodymams gauti. Kalbos analizatorius atfiltruoja netinkamus užklausos rezultatus. 11 paveikslėlyje yra pavaizduota šio modelio architektūra.



11 pav. Modelio architektūra

Žėmiau aprašyta, kokiais žingsniais yra vykdomas algoritmas:

1 Žingsnis. Vartotojas paduoda Ontologijų analizatoriui pirminę ontologiją OWL formatu. Ontologijų analizatorius ištraukia iš ontologijos visų klasių pavadinimus. Gautas sąrašas pasiunčiamas paieškos moduliui.

2 Žingsnis. Paieškos modulis nuskaito visus šablonus iš šablonų bibliotekos. Tada jis paima klasių sąrašą ir pagal kiekvieną klasę atlieka paiešką internete su visais šablonais. Paieškos rezultatai yra rastų HTML puslapių skaičius ir surasti HTML puslapiai kiekvienam klasės šablonui.

3 Žingsnis. Surasti HTML puslapiai ir jų skaičius patenka pas kalbos analizatorių. Kalbos analizatorius taip pat naudoja šablonų biblioteką, nes kiekvienas šablonas turi reguliarią išraišką (skyrelis 2.3), kuri reiškią kokių leksinių ir sintaksinių formų reikia ieškoti tekste. Taigi kalbos analizatorius, pereina per visą puslapio tekstą ir sužymi visų žodžių kalbos dalis. Tada jis paima iš šablono, pagal kurį šis puslapis buvo rastas, reguliarią išraišką, įrašo vietoj kintamojo klasės pavadinimą ir atlieka paiešką jau sužymetame html puslapyje. Skaičius, kiek kartų tekste

pasitaikė frazės atitinkančias reguliarojoje išraiškoje kalbos konstrukciją yra įsimenamas kiekvienam šablonui ir kiekvienai klasei.

Žodžių kalbos dalių sužymėjimui yra naudojami specialūs įrankiai – kalbos dalių priskyrėjai (angl. *part of speech taggers*). Yra sukurta nemažai tokių įrankių, bet labiausiai paplitę yra tie, kurie naudoja statistinius (pavyzdžiui, Markovo modelis) metodus. Šios grupės įrankiai gali būti priklausomi arba nepriklausomi nuo kalbos. Jeigu pasirenkamas nepriklausomas nuo kalbos įrankis, tada programuotojas turės parašyti funkciją, kuri ištraukia žodžio šaknį, o taip pat sukurti kalbos žodyną, kuriame kiekvienam žodžiui nurodyta jo kalbos dalis.

4 Žingsnis. Toliau, sužymėti ir atfiltruoti html puslapiai patenka pas klasifikatorių. Klasifikatorius priima sprendimą, kokį meta-savybės požymį priskirti klasei remiantis statistiniais duomenimis, gautais iš paieškos pagal šablonus rezultatų. Gali būti naudojami įvairūs algoritmai, kurie yra aprašyti žemiau. Taigi, klasifikatorius kiekvienai koncepcijai priskiria vienybę, tapatybę, griežtumą ir priklausomumą su jai būdingais požymiais – teigiamu ar neigiamu.

5 Žingsnis. Sužymėtosios koncepcijos vėl patenka pas ontologijų analizatorių, kuris sukuria OWL ontologiją remdamasis OntoClean taisyklėmis bei egzistuojančiais ryšiais tarp klasių.

3.3 Šablonų biblioteka

Šablonų biblioteka – tai aprašytas XML kalba abstrakčių šablonų rinkinys kiekvienai meta-savybei (U, I, R, D). Šablone nurodyta kokio tipo įrodymą jis pagamina (pvz., anti-griežtumo įrodymas), vieno ar daugiau kintamųjų deklaracijas ir web užklausų šablonai, kuriuose vietoje kintamojo įrašomas koncepcijos vardas. „Google regex“ yra reguliari išraiška ir naudojama tam, kad rasti ieškomą paieškos frazę.

Paieškos frazė ieškoma tekste, kuriame sužymėtos visų žodžių kalbos dalys. Žemiau pateiktas šablono pavyzdys, kuris ieško neigiamo griežtumo įrodymų. Tokiu būdų, naudojant šią reguliarią išraišką tekste prieš savybę (x) gali būti ne vieno arba vienas daiktavardis (kuris žymimas „NN“) ir po savybės gali eiti bet kas išskyrus daiktavardį. Tai reiškia, kad frazės panašios į „he is no longer a computer hacker“ nebus atrinktos, nes tai nėra neigiamo griežtumo įrodymas.

```
<pattern>
      <variable name=”x” />
      <evidence type=”false ” for=”R” />
            <google regex=”is \t \w + no \t\w + longer \t (DT\w + \t) ?(NN)
            x\t [^(NN)]”>
            <query string=” is no longer a x” />
            <query string=” is no longer an x” />
            <query string=” is no longer x” />
      </google>
</pattern>
```

12 pav. Šablono struktūra

Šablonai ieško tik teigiamų arba neigiamų meta-savybių įrodymų ir neieško anti- įrodymų. Tačiau, sprendžiant ar viena klasė negali paveldėti iš kitos klasės užtenka parodyti, kad koncepcija įgyja neigiamą meta-savybę.

Ši šablonų biblioteka yra lengvai plečiama. Tereikia pridėti naują šabloną.

3.3.1 Šablonai

Kadangi visų galimų šablonų aibė yra labai didelė, buvo išrinkti patys bendriausi šablonai, nepriklausantys nuo dalykinės srities. Frazių pagal šabloną paieškai yra naudojami HTML puslapiai anglų kalba. Puslapiai lietuvių kalba nebuvo nagrinėjami, kadangi „lietuviško“ interneto apimtis yra kur kas mažesnė už angliškąjį ir tam retai pasitaikantiems apibrėžimams bus grąžinamas labai mažas ar išvis negrąžinamas html puslapių skaičius, t.y. sunku bus rasti įrodymu.

Griežtumas

Pagal griežtumo apibrėžimą, jeigu kažkoks individas tapo arba nustojo būti konkrečios klasės nariu, tada galima teigti, kad šios klasės narystė, pavyzdžiui, tokia kaip *būti studentu*, nėra esminė visiems šios klasės individams. Todėl, neigiamą griežtumo požymį galima gauti iš šių šablonų.

Is no longer a/an/- student (daugiau nebe studentas)

Became a/an/- student (tapo studentu)

While being a/an/- student (kol buvo studentu)

Jeigu nebuvo rasta frazių atitinkančių šiuos šablonus, reiškia savybė yra esminė.

Vienybė

Teigiamos vienybės požymis klasei gali būti priskirtas tada, jeigu jos individo visos dalys gali būti identifikuotos/apibrėžtos ir įgyja tą patį vienybės kriterijų. Todėl, norint nustatyti ar klasė turi/neįgyja vienybės kriterijų, reikia atsakyti į klausimus Kas yra objekto dalis? Kas nėra objekto dalis? Kokiomis sąlygomis objektas yra visas? Jeigu išeis atsakyti į šiuos klausimus didžiajai daliai klasės objektų, tada galima tai laikyti teigiamu vienybės požymiu:

Part of a/an/- object (objekto dalis)

Be to, objektai, kurie nėra skaičiuojami, neįgyja vienybės kriterijaus, todėl mes galime rasti teigiamus vienybės požymius su šiais šablonais:

One/two object (vienas/du objektas (-ai))

Aišku, žodžiai ‚one' ir ‚two' turėtų būti pasirenkamieji, tačiau Google kol kas nemoka apdoroti užklausų, turinčių reguliarias išraiškas.

Neigiamas požymis gali būti gautas naudojant šabloną, kuris parodo objekto neskaičiuojamumą. Anglų kalboje su neskaičiuojamais žodžiais naudojamas žodis ‚amount':

Amount of object (medžiagos kiekis)

Tapatybė

Pagal apibrėžimą, du objektai yra tapatūs tada ir tik tada, jeigu jie turi tas pačias dalis. Tai gali būti išreikšta šablonais, tiekiančiais teigiamą tapatybės įrodymą.

Object consists of two/three parts (objektą sudaro dvi/trys dalys)

Object is composed of two/three parts

Object is identified by....

Neigiami ir teigiami įrodymai taip gali būti gauti iš šablonų, kurie patikrina ar objektas yra daiktavardis ar būdvardis. Jeigu objektas yra būdvardis (atsako į klausimą koks?), tada jis įgyja neigiamą tapatybės požymį, jeigu daiktavardis (kas?) – tada teigiamą. Tam, kad nustatyti objekto kalbos dalį yra naudojamas kalbos analizatorius, kuris sužymi kalbos dalis tekste. Google užklausias tik pagal paieškos žodį, t.y. patį objektą. Taigi, šablonas būtų tiesiog

Object

Taip pat, skaičiuojamumas reiškia, kad objektas gali būti identifikuojamas. Todėl galima panaudoti tuos pačius šablonus, kurie buvo naudojami teigiamo vienybės įrodymo gavimui:

One/two object (vienas/du objektas (-ai))

Amount of object

Priklausomybė

Tarp meta savybių griežtumas, vienybė, tapatybė ir priklausomybė, priklausomybę yra sunkiausia nustatyti automatiškai dėl to, kad šablone turi būti mažiausiai dvi koncepcijos, tarp kurių egzistuoja priklausomybės ryšys. Teigiamą priklausomybės įrodymą galima rasti pagal tokį šabloną:

Cannot be a/an/- object without (negali būti objektu be...)

3.4 Reguliarios išraiškos

Reguliarios išraiškos reikalingos rezultatų, gautų iš paieškos interneto, filtravimui. Kalbos dalių priskyrėjas pereina per HTML puslapį, sužymi visas kalbos dalis tekste, po ko automatinių ontologijų vertinimo įrankis ieško frazių, atitinkančių reguliaria išraiška. Jeigu paieškos frazė atitinka reguliaria išraiška, tada ji įtraukiama savybių požymių apskaičiavimą, priešingu atveju – neįtraukiama.

Šiuo atveju reguliarios išraiškos padeda analizuoti kontekstą ir ar tam tikros frazės nėra įrodymai. kad, pavyzdžiui, „He is no longer a computer hacker“ nėra sąvokos „computer“ griežtumo įrodymas, nes yra žodis „hacker“, kuris ir yra pagrindinis šiame kontekste. Tam reikia, kad tekste po ieškomo žodžio nebūtų dar vieno daiktavardžio, nes tada gaunasi, kad ieškomas žodis yra po jo einančio daiktavardžio savybė. Pats žodis gali būti taip pat būdvardžiu. Šią reguliarią išraišką galima būtų užrašyti taip:

is no longer (*tikrinis daiktavardis vienaskaitoje/ daiktavardis vienaskaitoje daugiskaitoje/ paprastas daiktavardis vienaskaitoje/ daiktavardis vienaskaitoje daugiskaitoje/būdvardis*) **x NE(***tikrinis daiktavardis vienaskaitoje arba daugiskaitoje/ paprastas daiktavardis vienaskaitoje arba daugiskaitoje***)**

Jeigu tai užrašyti PERL kalba, gausis toks sakinys:

is\t\w+ no\t\w+ longer\t(DT \w+\t)?(NN|NP|NNS|NPS|JJ) x\t[^(NN|NP|NNS|NPS)]

Čia simbolis ‚\t‘ reiškia tabuliaciją, ‚\w‘ bet koks simbolis iš aibės `[a-zA-Z_0-9]`. Ženklas ^ reiškia neigimą. Konstrukcija ^() reiškia „išskyrus elementus iš grupės“. Žodžiai NN, JJ ir pan. žymi kalbos dalis. PERL kalbos dalių suvestinė pateikta terminų žodynėlyje.

Išnagrinėjus kitus šablonus gavosi, kad tokios pat taisyklės galima pritaikyti ir kitiems šablonams. Todėl, pritaikius tas pačias PERL kalbos taisykles, buvo gautos reguliarios išraiškos anglų kalba:

Savybė	Regex
-R	is\t\w+ no\t\w+ longer\t(DT \w+\t)?(NN|NP|NNS|NPS|JJ) x\t[^(NN|NP|NNS|NPS)]
-R	\w+ while\t\w+ being\t(DT \w+\t)?(NN|NP|NNS|NPS|JJ) x\t[^(NN|NP|NNS|NPS)]
-R	\w+ became\t(DT \w+\t)?(NN|NP|NNS|NPS|JJ) x\t[^(NN|NP|NNS|NPS)]
+I	\t(NN|NP|NNS|NPS) x\t
-I	\t(JJ|JJR|JJS) x\t
+U	\w+ part\t\w+ of\t(DT \w+\t)?(NN|NP|NNS|NPS) x\t[^(NN|NP|NNS|NPS)]
+U	CD \w+\t(NN|NP|NNS|NPS) x(s)?\t[^(NN|NP|NNS|NPS)]
-U	\w+ amount\t\w+ of\t(NN|NP|NNS|NPS) x\t[^(NN|NP|NNS|NPS)]
-D	\w+ cannot\t\w+ be\t(DT \w+\t)?(NN|NP|NNS|NPS) x\t\w+ without

1 lentelė. Reguliarios išraiškos, naudojamos automstiniame ontologijų vertinime

3.5 Klasifikatoriai

Šio skyrelio tikslas yra pasiūlyti kelis algoritmus duotąjai klasifikavimo problemai spręsti. Kitaip tariant, reikia algoritmo, kuris nuspręstų į kurią iš dvejų klasių patenka elementas remiantis kiekybinėmis šio elemento charakteristikomis. Formaliai tai galima užrašyti taip:

Remiantis pradiniais duomenimis $\{(x_1, y_1), ..., (x_n, y_n)\}$ sukurti klasifikatorių h: $X \rightarrow Y$, kuris kiekvienam elementui x iš aibės X priskiria klasifikavimo požymį y iš aibės Y, nusakytą nežinomu atvaizdavimu g: $X \rightarrow Y$.

Pats paprasčiausias klasifikatorius duotajai problemai spręsti yra linijinis klasifikavimo algoritmas. Jis apibrėžia klasifikavimo funkciją kaip linijinę charakteristikų kombinaciją.

$$h = x_1 * w_1 + x_2 * w_2 + ... + x_n * w_n ,$$

kur x_n – klasės kiekybinė charakteristika ir w_n – atitinkamas svoris.

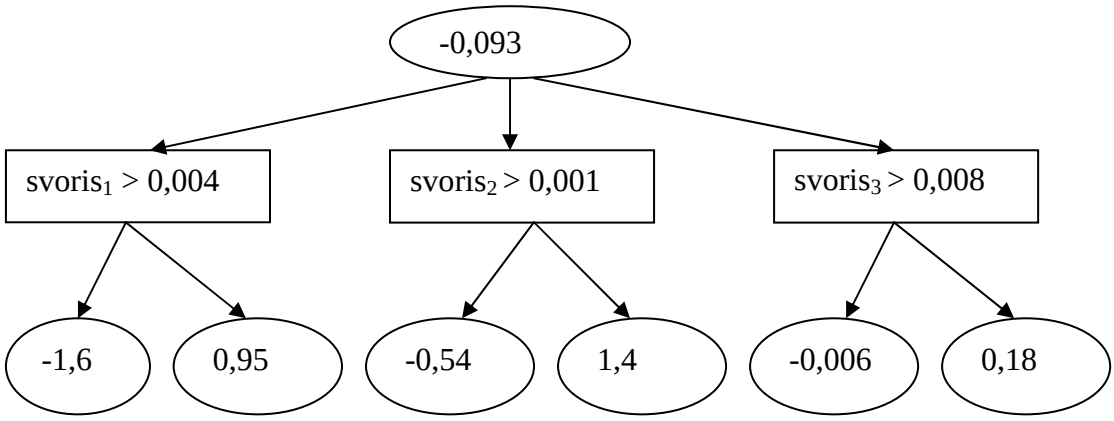
Pagrindinis linijinio klasifikatoriaus privalumas yra greitis, tačiau šis klasifikatorius nesuranda *optimalaus* sprendimo, nes remiasi tik pradiniais duomenimis.

Kitas variantas – naudoti algoritmą, nusakytą ‚mokymosi metodu'. Šios grupės klasifikatoriai po kiekvienos iteracijos perskaičiuoja koncerto charakteristikų svorius. Šių klasifikatorių veikimas gal užtrukti ilgiau, negu linijinio, tačiau jie leidžia gauti tikslesnį rezultatą remiantis naujaisiais duomenimis. Tokio tipo klasifikatoriai yra alternatyvūs sprendimų medžiai (*angl. alternating decision tree*) ir neuroniniai tinklai.

Žemiau yra pateiktas pavyzdys, kaip klasifikatorius gali būti naudojamas meta-savybės požymio nustatymo problemai spręsti.

Pavyzdys

Sprendimui, kokią meta-savybę turi koncepcija, priimti yra naudojamas alternatyvūs sprendimų medis. Šis klasifikatorius pagal koncepcijos charakteristikas nustato, kokiai iš dviejų klasių (+ ir -) priklauso ši koncepcija. Kiekvienai meta-savybei bus naudojamas atskiras klasifikatorius. Klasifikatoriaus pavyzdys:



13 pav. Klasifikavimo pavyzdys.

Šiame pavyzdyje kiekviena viršūnė kuri pavaizduota kvadratu vadinama *sprendimų viršūne*, o jos vaikai vadinami *prognozėmis*. Neigiama reikšmė reiškia teigiamą įrodymą, o teigiama – neigiamą įrodymą. Koncertas klasifikuojamas pagal charakteristikas. Charakteristikos lyginamos su atitinkamais svoriais, ir priklausomai nuo rezultato pasirenkama *prognozė.* Sudėjus visas prognozes gauta reikšmė nusako kokiai klasei priklauso ši koncepcija. Pavyzdžiui, jeigu skaičius yra teigiamas, tai koncepcija įgyja teigiamą meta-savybę įrodymą. Jeigu skaičius neigiamas – tada klasė įgyja neigiama meta-savybę. Iteracijos pabaigoje šakninė prognozė ir svoriai yra perskaičiuojami, bei pridedama nauja *sprendimų viršūnė.*

Koncepcijos charakteristikų svoriai apskaičiuojami sekančiu būdu. Charakteristika tai iš esmės kiek užklausų buvo gražinta pagal tam tikrą šabloną. Taigi, charakteristikos svoris *W*, nusakantis teigiamą arba neigiamą meta-savybės $p \in \{R, I, U, D\}$ įrodymą apskaičiuojamas pagal formule

$$W(p,i,c) = \frac{\sum_{q \in Q_i} lf\,(hits\,(q_c))}{lf\,(hits\,(c))},$$

kur Q_i yra užklausų, susijusių su tam tikru šablonu *i* rinkinys, q_c – tai užklausa su įrašytu koncepcijos vardu vietoj kintamojo, o *hits(q_c) ir hits(c)* – rastų puslapių skaičius gautų nuo q_c ir *c* atitinkamai. *lf* – funkcija, atliekanti užklausų filtravimą. Kalbant paprastai, paimami visų atsakymų į šablono užklausas skaičiai, susumuojami ir visa tai padalinama iš skaičiaus atsakymų pagal paieškos frazę „c“. Gautas skaičius yra įeities duomenys sekančiajame modulyje „klasifikatorius“.

Taigi, šiai klasifikavimo problemai spręsti gali būti naudojami įvairūs algoritmai, pavyzdžiui, linijiniai algoritmai, sprendimo medžių grupės algoritmai arba neuroniniai tinklai. Tačiau, šiuo metu nėra algoritmo kuris padėtų nustatyti, kuris klasifikatorius geriausiai tinka duotajai problemai spręsti. Todėl galbūt reikės išbandyti kelis klasifikavimo algoritmus.

3.6 Algoritmo įgyvendinimas

Kadangi algoritmo tikslas yra atlikti ontologijų vertinimą, galima sukurti atskirą įrankį, kuris užsiims tik ontologijos vertinimu, arba integruoti jį į įrankį, palaikantį pilną ontologijos gyvavimo ciklą, arba jo dalį (pavyzdžiui, ontologijos sukūrimą).

Ontologijų kūrimo įrankis pasipildo nauju funkcionalumu: klasės redagavimo lange sukuriamas naujas mygtukas „OntoClean įvertinimas“. Paspaudus jį sistema priskirs visoms klasėms atitinkamus meta-savybių požymius, bei nurodys, kokie sąryšiai tarp klasių pažeidžia OntoClean taisykles. Vartotojas gali įvertinti arba visą ontologiją, arba atskiras jos klases.

Retai pasitaikantiems klasių pavadinimams gali būti rasta per mažai duomenų arba visai nerasta. Tokiu atveju sistema gali pateikti paaiškinimus prie kiekvieno varianto (teigiamo ir neigiamo) ir leisti vartotojui pačiam pasirinkti reikšmę. Vartotojas gali pasirinkti vertinimo būdą: pilnai arba dalinai automatizuotą.

Išvados

Darbo metu buvo atlikta ontologijų vertinimo metodikos OntoClean analizė: buvo išnagrinėti meta-savybių apibrėžimai, pagrindiniai dėsniai ir pateiktas metodikos naudojimo pavyzdys. Taip pat buvo išnagrinėta, kaip šiuolaikinių įrankių, tokių kaip Protégé ir OntoEdit, pagalba galima įvertinti ontologijos korektiškumą pagal OntoClean. Buvo nustatyta, kad pagrindinis šių įrankių trūkumas yra tas, kad ontologijų vertinimo procesas yra daromas rankiniu būdu, kas reikalauja daug vertintojų pastangų ir laiko. Remiantis prielaidą, kad meta-savybių įrodymų galima ieškoti tekstiniuose informacijos šaltiniuose, pavyzdžiui, internete, aš sukūriau algoritmo, automatiziuojančio ontologijų vertinimą, modelį.

Pasiūlyto metodo privalumai:

1. algoritmas padeda greičiau atlikti ontologijos vertinimą,
2. projektuotojas neprivalo turėti OntoClean žinias,
3. šablonų biblioteka yra lengvai plečiama, kas leidžia tobulinti vertinimo tikslumą,
4. algoritmas duoda didžiausią naudą, kaip ontologija yra didelė,

Algoritmo silpnoji pusė yra ta, kad jis neskiria tarp negriežtų ar anti-griežtų meta-savybių, nuo ko šiek tiek suprastėja vertinimas, tačiau dėl lengvo šablonų bibliotekos plečiamumo galima pridėti naujus šablonus, kurie padėtų gauti geresnius ontologijos įvertinimus.

4. Literatūra

[1] Falbo, An Ontologically Well-Founded profile for UML Conceptual Models, 2003.

[2] N. Guarino, C.A. Welty, An Overview of OntoClean, In Steffen Staab and Rudi Studer, eds., *The Handbook on Ontologies.* Pp. 151-172. Berlin:Springer-Verlag.

[3] Guarino, Nicola, Chris Welty. 2000. Ontological Analysis of Taxonomic Relationships. In, Laender, A. and Storey, V., eds, *Proceedings of ER-2000: The 19th International Conference on Conceptual Modeling.* Springer-Verlag. October, 2000.

[4] Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001. http://protege.stanford.edu/publications/ontology_development/ontology101.html

[5] www.ontoclean.org

[6] A. H. Dekker, Possible Worlds, Belief, and Modal Logic: a Tutorial, Oct. 5, 2004

[7] Modal Logic, Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/entries/logic-modal/#6

[8] A. Lozano-Tello and A. Gomez-Perez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. Journal of Database Management, Vol. 15, No. 2, 15(2), 2004.

[9] F.M. Couto**,** M.J. Silva, P.M. Coutinho**,** Finding genomic ontology terms in text using evidence content, BMC Bioinformatics 2005, 6(Suppl 1)**:**S21.

[10] N. Guarino, C.A. Welty, Supporting Ontological Analysis of Taxonomic Relationships. Data and Knowledge Engineering 39 (2001) 51 – 74.

[11] E.J. Lowe, Kinds of Being. A study of Individuation, Identity and the Logic of Sortal Terms, Basil Blackwell, Oxford, 1989.

[12] Towards a methodology for ontology-based model engineering. In Proceedings of ECOOP-2000 Workshop on Model Engineering. Cannes, France.

[13] G. Guizzardi. The role of foundational Ontologies for Conceptual Modeling and Domain Ontology Representation, Laboratory of Applied Ontology, ISTC-CNR, Trento, Italy.

[14] Guarino, N., Carrara, M., and Giaretta, P. An Ontology of Meta-Level Categories, LADSEB-CNR Int. Rep. 06/93.

[15] Nicola Guarino, Chris Welty: A Formal Ontology of Properties. EKAW 2000: 97-112.

[16] G. Hirst, Existence assumptions in knowledge representation, AI 49 (1991) 199-242.

[17] J. Volker et al., Automatic Evaluation of Ontologines (AEON)

[18] Jorge Cardoso, “The Semantic Web Vision: Where are We?” IEEE Intelligent Systems, September/October 2007, pp.22-26, 2007.

[19] Protégé. http://protege.stanford.edu/

[20] York Sure, Jurgen Angele, Steffen Staab, OntoEdit: Multifaceted Inferencing for Ontology Engineering, Journal on Data Semantics 1 (1): 128-152. November 2003.

[22] Regullar expressions.http://en.wikipedia.org/wiki/Regular_expression#Syntax

[23] Ian H. Witten, E. Frank. Data mining, Elsevier, second ed.

[24] P. Cimiano et al., Towards the self-annotating web. In proceedings of the 13th WWW Conference, pp. 462-471, 2004.

[25] Wikipedia, Statistical Classification. http://en.wikipedia.org/wiki/Statistical_classification

5. Terminų žodynėlis

OWL –Web Ontology Language

W3C - The World Wide Web Consortium

URI - Uniform Resource Identifier

NN – tikrinis daiktavardis vienaskaitoje

NNS – tikrinis daiktavardis daugiskaitoje

NP – paprastas daiktavardis vienaskaitoje

NPS – paprastas daiktavardis daugiskaitoje

JJ – būdvardis

JJS – būdvardis daugiskaitoje

JJR – palyginamasis būdvardis

DT – artikelis

CD - kiekinis skaitvardis