VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Ontologijos kūrimas bei jos teisingumo užtikrinimas

Atliko:2M kurso, 9 grupės studentė

Jelena Liachovskaja

Darbo vadovas:

Asist. L. Būtėnas

Vilnius
2007

Turinys

Įvadas

Temos aktualumas

Internetinis pasaulis plečiasi beveik į visas gyvenimo sferas. Tinklalapis tampa vis naudingesniu ir svarbesniu informacijos šaltinių. Duomenų apdorojimo priemonėms vis sudėtingiau apdoroti informacinį krūvį, kuris pildomas kiekvieną dieną. Be to, duomenys Internete neorganizuoti ir nesistemiški. Toks dezorganizavimas apsunkina reikalingos informacijos paieškos procesą. Todėl reikalingas naujas žingsnis Interneto panaudojimui - tai yra semantinis informacijos vaizdavimas ir aprašymas. Paieškos sistemos, tokios kaip „Yahoo!“, „Google“ darbo metu naudoja paieškos mechanizmą, kuris naudoja raktinius žodžius ir nenagrinėja konteksto, kuriame egzistuoja informacija. Būtent dėl to, tokių sistemų darbo rezultatai gali būti tūkstančiai nuorodų. Dėl šios priežasties, tolimesnį Interneto vystymą mokslininkai numato Semantinio Web koncepcijoje ir ontologijų panaudojime. Semantinio Web koncepciją pasiūlė Tim Berners-Li – vienas iš World-Wide Web kūrėjų ir WWW-konsorciumo (W3C) pirmininkas. Pagal Semantinio Web kūrėjus, Semantinis Web ir jame taikomos ontologijos turi aprūpinti kompiuterius informacijos supratimu ir pateikti tinkamą informaciją vartotojams.

Magistrinio darbo tikslai, tyrimo užduotys

Šio magistrinio darbo tikslai yra:

- Išanalizuoti esančias ontologijas;
- Išnagrinėti ontologijų kūrimo principus bei taisykles;
- Aprašyti ontologijos kūrimo metodologiją;
- Panaudoti metodologiją praktiškai kuriant vaistų ontologiją, pasirinkus kūrimo įrankį;
- Atlikti esamų ontologijų tikrinimo priemonių analizę;
- Atrinkti tinkamiausią tikrinimo priemonę ontologijos teisingumo užtikrinimui.

1 Problemos konkretinimas

Dirbtinio intelekto mokslas ontologiją apibūdina kaip „dalykinės srities konceptualizacijos specifikacija“, arba paprasčiau, tai yra duomenų struktūra, kuri formaliai užduoda santykius tarp terminų. Tai tam tikras dalykinės srities supratimų žodynas ir akivaizdžiai išreikštų sakinių visuma. Skirtingose dalykinėse srityse vienodi supratimai gali būti išreikšti skirtingais terminais. Šiais atvejais ontologijų mechanizmas leidžia formuoti prasmingus hierarchinius ryšius tarp objektų, apibendrinti ir kartu naudoti globalinę informaciją, t. y. realizuoti kontekstinę dalykinės srities paiešką, kuri gali rasti tokius vartotojui reikalingus resursus, kuriuose nebus nei vieno žodžio iš pradinės užklausos.

Ontologijos kūrimas reikalingas:

- Kad žmonės arba programų agentai turėtų bendrą informacijos struktūros supratimą;
- Kad dalykinės srities žinios galėtų būti naudojamos pakartotinai šioje dalykinėje srityje;
- Tam tikros dalykinės srities žinių analizei;

Dalykinės srities ontologija dažnai savaime nėra tikslas. Ontologijos kūrimo procese apibrėžiamas duomenų rinkinys ir jų struktūros, kurios naudojamos kitose programose. Užduočių spendimo metoduose, programiniai agentai vietoj duomenų naudoja ontologijas ir duomenų bazes, kurios yra sukurtos pagal šias ontologijas.

2 Sprendimo būdų analizė ir pagrindimas

Nėra vienintelės „teisingos“ ontologijos kūrimo metodologijos. Darbe bus nagrinėjamas iteracinis būdas kuriant ontologijas: ontologijos kūrimas pradėdamas nuo pačių abstrakčiausių terminų nagrinėjimo. Toliau ontologija tikrinama ir tikslinama papildomomis detalėmis.

Yra tam tikros fundamentalinės ontologijos kūrimo taisyklės, kurios bus naudojamos darbe. Šios taisyklės yra gana kategoriškos, bet jos gali padėti priimant projektinius sprendinius.

1. Nėra vienintelio teisingo dalykinės srities modeliavimo būdo – visada egzistuoja galimos alternatyvos. Geriausias sprendimas beveik visada priklauso nuo planuojamos taikomosios programos ir laukiamo sprendimo.
2. Ontologijos konstravimas – tai būtinai iteracinis procesas.
3. Ontologijos supratimai turi būti artimi objektams (fiziniams arba loginiams) ir santykiams tam tikroje dalykinėje srityje. Supratimai - tai objektai arba veiksmažodžiai (santykiai) sakiniuose, kurie aprašo tam tikrą dalykinę sritį.

Kitaip tariant, priklausomai nuo to, kam bus naudojama ontologija ir kiek detali ji bus, gali turėti įtakos sprendžiant modeliavimo problemas. Tarp kelių galimų alternatyvų reikia nuspręsti kuri geriausiai padės išspręsti duotą uždavinį ir bus akivaizdesnė, labiau plečiama ir paprasčiau aptarnaujama. Taip pat reikia atsiminti, kad ontologija – tai realaus pasaulio modelis ir ontologijos supratimai turi atspindėti šią realybę. Apibrėžus pradinę ontologijos versiją, ją galima įvertinti ir patobulinti, panaudojant ją programose arba užduočių sprendimo metoduose ir/arba aptariant ontologiją su dalykinės srities ekspertais. Rezultate, tikriausia reikės peržiūrėti pradinę ontologiją. Šitas iteracinio projektavimo procesas bus tęsiamas viso ontologijos gyvavimo ciklo metu.

3 Mokslinės literatūros ir egzistuojančių ontologijų nagrinėjimas

Darbe nagrinėjamos 3 literatūroje apibrėžtos ontologijos. Ontologijos paimtos iš skirtingų dalykinių sričių, tačiau visose išlaikomas hierarchinės struktūros principas. Nagrinėjant ontologijas pateikiami skirtingi jų formavimo etapai, t. y. pirmoje ontologijoje nagrinėjama vyno ontologiją ir jos formavimo schema. Sekanti ontologija nagrinėjama realizavimo etape, aprašant suformuotas schemas ontologijos kalbos pagalba. Trečiame pavyzdyje nagrinėjamas ontologijos vystymo ir pildymo procesas tinklalapio dalykinėje srityje.

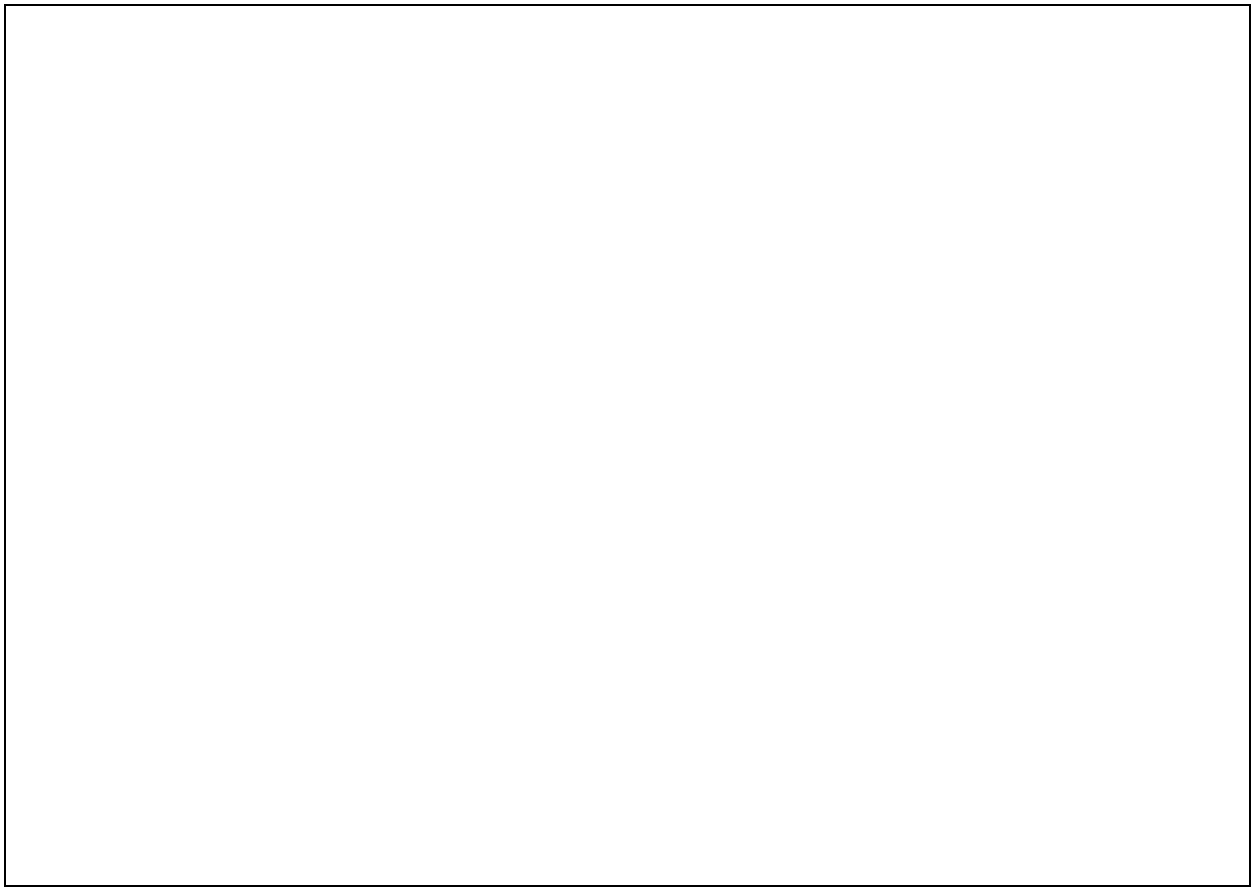
3.1 Vyno ir maisto ontologija

Literatūroje nagrinėjama vyno ir maisto ontologija, o taip pat patiekalų ir vynų kombinacijų kūrimo procesai. Ši ontologija gali būti panaudota kaip pagrindas dirbant su restorano instrumentais: Vienas instrumentas-programa gali sudarinėti vynų sąrašą valgiaraščiui einamajai dienai arba gali atsakyti į padavėjų ir lankytojų užklausas. Kitas instrumentas gali nagrinėti vyno sandėlių inventorių ir siūlyti trūkstamų vynų kategorijas, kurias reikia nupirkti perpardavimui arba patiekalų paruošimui.

Visų ontologijų pagrindas yra klasės. Klasės apibūdina dalykinės srities supratimus. Vynų ontologijoje vynų klasė aprašo visus vynus. Šios klasės egzemplioriai gali būti tam tikras vynas bokale arba vyno butelis. Klasė gali turėti poklasius, kurie apibūdina konkretesnius supratimus. Vynų ontologijoje – vynų klasę gali išskirti į raudonus ir baltus. Taip pat kiekviena klasė ir klasės egzempliorius gali turėti savybes (slotes). Vynų ontologijoje tokių savybių pavyzdžiai yra vyno stiprumas, vyno gamykla, cukraus koncentracija ir t. t.

Vynų ontologijos pavyzdys: kelios vynų srities klasės ir santykiai tarp jų. Raudona spalva pažymėtos klasės, mėlyna – poklasiai, geltona – klasės egzemplioriai, ryšiai – savybės arba santykiai tarp klasių: [FH97]

1 pav. „Vynų ontologijos schema“:



3.2 Floros ir faunos ontologija

Kitas ontologijos pavyzdys yra ontologija aprašanti Afrikos gamtą. Šioje ontologijoje aprašomos tokios klasės kaip gyvūnai, augalai. Šių klasių poklasiai yra tokie kaip medžiai, šakos, žolė ir t. t. Klasės ir poklasiai turi savybes, pavyzdžiui gyvūnai gali maitintis kitais gyvūnais arba augalais. Tokią floros ir faunos ontologiją naudoja gamtos mokslininkai atliekant mokslų tyrimus. [SD01]

```
class-def animal          % animals (“gyvūnai”) — tai klasė
class-def plant                   % plants (“augalai”) — tai klasė
  subclass-of NOT animal          % t.y. skiriasi nuo gyvūnų
class-def tree
  subclass-of plant                       % trees (“medžiai”) — tai augalų tipas
class-def branch
  slot-constraint is-part-of  % branches (“šakos”) — tai medžių dalys
  has-value tree
```

3.3 Tinklalapio ontologija

Sekanti nagrinėjama ontologija yra tinklalapio ontologija. Nagrinėjamas WEB-tinklalapis su straipsniais, kurie grupuojami į skyrelius ir diskusijas pagal kiekvieną straipsnį. Taip yra, egzistuoja klasės ir poklasiai tokios kaip rubrikos, replikos, materialai bei ryšys tarp jų. Šioje ontologijoje galimos sekančios ryšys: „tekstas yra iš rubrikos“, „tekstas sukūrė repliką/naują diskusiją“, „tekstas yra atsakymas į“ ir t. t. Kuriant naujus straipsnius arba diskusijas, arba komentarus prie straipsnių, automatiškai padidinama ontologija ir jos hierarchijos medis.[AŽ00]

4 Specialių ontologijos kalbų nagrinėjimas

Šiuo metu yra sukurta daugybė kalbų ontologijoms kurti. Visų pagrindinių ontologijos kalbų pagrindas yra XML struktūros.

XML – tai išplėstinė žymių kalba. Ji skirta dokumento turiniui, o ne formatui ar maketui užkoduoti, t.y. akcentuojamas teksto dalies atpažinimas ir pavadinimo priskyrimas bei nuoroda, kokios struktūros turi būti tekstas. XML pagrindinis tikslas buvo palengvinti struktūrinių tekstų ir informacijos mainus internetu sujungtuose kompiuteriuose. [CG05]

Viena iš paskelbtų standartų, kurie pripažįstami semantinio interneto pagrindu yra RDF standartas. RDF (Resource Description Framework) – tai išteklių aprašymo sistema [ter03], kuri padeda, naudojantis XML kaip mainų sintaksę, užkoduoti, apsikeisti ir dar kartą panaudoti struktūrinius meta duomenis. Sistema yra galinga dėl to, kad ji įveda struktūrinius apribojimus, kurie leidžia nuosekliai ir vienareikšmiškai užkoduoti ir keistis standartizuotais meta duomenimis.

Naujausi ontologijos kalba OWL (Web Ontology Language) buvo pasiūlyta WWW-konsorciumu (W3C) 2002 metais. Ši kalba duoda daugiau galimybių, negu XML ar XML

Schema. Ontologijoje vaizduojamos žinios, skirtingai nuo XML schemų, kuriuose vaizduojamas pranešimų formatas. Daugybė WEB-standartų yra sudaryti iš pranešimų formatų kombinacijų ir protokolų specifikacijų. Bet specifikacijos negali palaikyti sistemas už konteksto ribų. Viena iš OWL ontologijų pirmenybių yra lengva prieiga prie instrumentų, kurie gali šias ontologijas „aptarinėti“. Ši kalba grindžiama RDF ir RDF-Schema.

WEB ontologijų kalba OWL buvo sukurta, kad būtų galimybė aprašinėti ontologijos klases bei santykius tarp šių klasių. [SWM04]

4.1 Ontologijos kalbos OWL (Ontology Web Language) nagrinėjimas

Darbe buvo parinkta OWL ontologijos kalba, nes šios kalbos pagalba galima:

1. formalizuoti klasių bei klasių savybių apibrėžimo sritį;
2. apibrėžti individus bei nustatyti jų savybes;
3. patikslinti šias klases ir individus iki tam tikro laipsnio panaudojant OWL formalią semantiką.

Tam kad aprašyti ontologiją, kurią programos agentai supras vienareikšmiškai ir galės naudoti, reikia naudotis teisingą OWL kalbos sintaksę bei formalią semantiką. Pagrindiniai OWL komponentai įtraukia klases, savybes ir individualinius elementus. [SWM04]

4.2 Klasės

Klasės yra pagrindiniai OWL ontologijos blokai. Klasės paprastai sudaro taksonominę hierarchiją (t. y. poklasiai-viršklasės). Klasės apibrėžiamos panaudojant elementą owl:Class. OWL kalboje egzistuoja dvi iš anksto apibrėžtos klasės: owl:Thing ir owl:Nothing. Pirma yra daugiau bendra ir įtraukia viską, antra – tai tuščia klasė. Bet kuri klasė, kurią aprašo vartotojas, yra klasės owl:Thing poklasis ir klasės owl:Nothing viršklasė. OWL klasių pavyzdžiai:

```
<owl:Class rdf:ID="SavingsAccount">
 <rdfs:subclassOf rdf:resource="#Account"/>
</owl:Class>
```

Elementas SavingAccount – tai klasės Account poklasis.

OWL palaiko 6 pagrindines klasių aprašymo galimybes. Paprasčiausias – tai klasė su vardu (named). Kiti tipai – tai klasių sankirtos (intersection), sujungimai (union), papildymai

(complement), apribojimai (restrivtions) ir įvardijimų klasės (enumerates). Pavyzdyje yra du klasių aprašymo tipai: apribojimų klasė apibrėžia SavingAccount kaip klasės su vardu Account poklasį. [NB05]

4.3 Savybės

Savybės yra 2-jų kategorijų:

- objekto savybės (Object properties), kurios suriša individualinius objektus tarpusavyje;
- duomenų tipų savybes (Datatype properties), kurios suriša individualinius elementus su duomenų tipų (sveikieji skaičiai, skaičiai su kablelių ir eilutės) reikšmėmis. Duomenų tipų nustatymui OWL naudojama XML schema.

Kiekviena savybė priklauso vienai iš sekančių kategorijų:

- funkcinė: kiekvienam objektui savybė gali turėti tik vieną reikšmę (pvz. žmogaus amžius, ūgis arba svoris);
- atvirkščiai-funkcinė: du skirtingi individualiniai elementai negali turėti vieną ir tą pačią reikšmę. Pavyzdžiui kiekvienas žmogus turi savo unikalų banko sąskaitos numerį.
- simetrinė: jeigu savybė suriša elementą A su elementu B, tai galima padaryti išvada, kad ši savybė taip pat suriša elementą B su elementu A. Simetrinių savybių pavyzdys yra reiškiniai „yra brolis (sesuo)“ arba „toks pat, kaip“;
- tranzityvinė: jeigu savybė suriša elementą A su elementu B, o elementą B su elementu C, tai galima padaryti prielaidą, kad šį savybė taip pat suriša elementą A su elementu C. Tarkim jeigu A aukščiau už B, o B aukščiau už C, tai A aukščiau už C. [NB05]

Klasės ir savybės gali turėti įvairių apribojimų.

5 Ontologijos srities ir masto apibrėžimas

5.1 Kuriamos ontologijos sritis

Darbe nagrinėjama ontologija apima medicinos sritį – vaistus (med. preparatus). Informaciją struktūrizuota pagal preparatų receptus. Nagrinėjama tokia receptų informacija kaip pavadinimas, gamintojas, kiekis, sudėtis, indikacijos, kontraindikacijos, dozavimas, nepageidaujamas poveikis. Medicininių preparatų informacijos šaltiniai yra Lietuvos vaistų tinklalapių informaciją.

5.2 Kuriamos ontologijos naudojimo sritis

Vaistų ontologija gali būti naudojama optimaliausio preparato pasirinkimui nurodytam pacientui atsižvelgiant į jo diagnozę, bei jau vartojamus vaistus.

Ontologija gali atsakinėti į sekančių tipų klausimus:

1. Koks preparatas reikalingas pagal duotą diagnozę?
2. Kaip reikia vartoti nurodytą preparatą?
3. Kokius nepageidaujamus poveikius turį nurodytas preparatas?
4. Koks preparatas yra tinkamiausias atsižvelgiant į paciento papildomas esančias lygas?
5. Kokius preparatus negalima vartoti su duotų preparatų?

Ir t .t.

Esant reikalui, ontologija gali būt papildyta, patobulinta arba net sujunkta su kitomis esančiomis ontologijomis. Gali būti panaudota specialiam programiniam agentui bei informacijos paieškai Semantiniame tinklalapyje.

6 Pagrindiniai žingsniai kuriant vaistų ontologiją

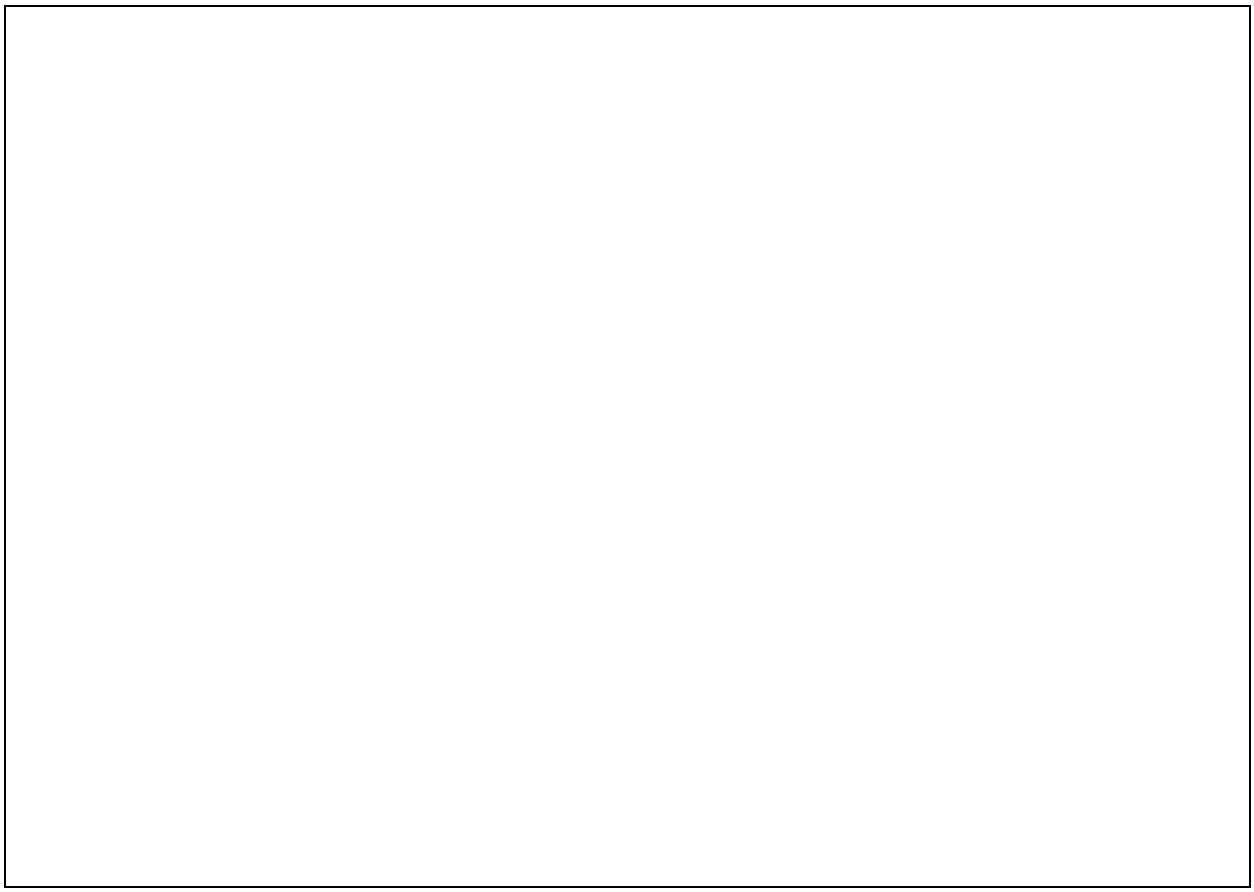
Bendras ontologijos kūrimo procesas:

- Egzistuojančių ontologijų nagrinėjimas ir panaudojimas;
- Ontologijos klasių apibrėžimas;
- Klasių išdėstymas į taksonominę hierarchiją (poklasis - viršklasė);
- Savybių apibrėžimas bei savybių galimų reikšmių nustatymas;
- Savybių reikšmių priskirimas egzemplioriams;

Po šių veiksmų sudaroma ontologijos žinių bazę.

Sudarant vaistų ontologijos schemą apytiksliai aprašomos klasės, poklasiai, santykiai, savybės (galimos ir prieštaringos). Bendra ontologijos schema, kuri buvo sudaryta pirmame ontologijos kūrimo etape:

2 pav. „Vaistų ontologijos schema“:

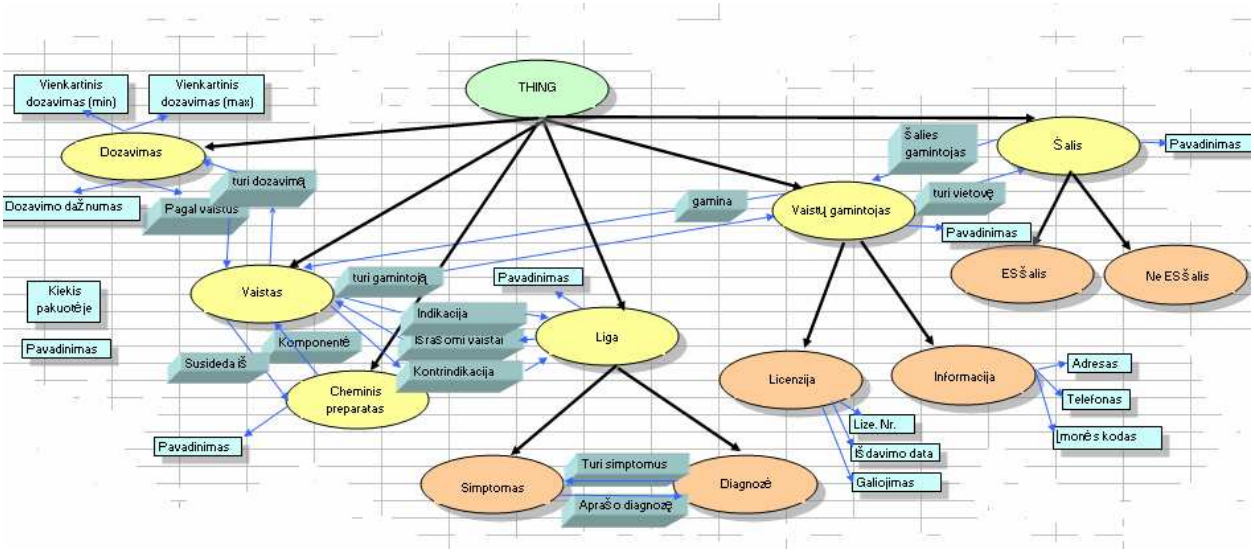


Aprašytos schemoje klasės turi tam tikrus apribojimus:

- Vienų vaistų indikacijos (savybė “Išrašomi”) ir kontraindikacijos negali sutapti;
- Kontraindikacijos neturi sutapti su vaistu, kuriam parašytos šios kontraindikacijos;
- Nustatant diagnozę pacientui siūlomas vaistas. Šio vaisto kontraindikacijos neturi sutapti su jau naudojamais vaistais bei esamomis lygomis;

Patobulinta ir papildyta ontologija, kuri buvo sudaryta sekančiame darbo etape:

3 pav. „Vaistų ontologijos patobulinta schema“:



6.1 Egzistuojančių ontologijų nagrinėjimas

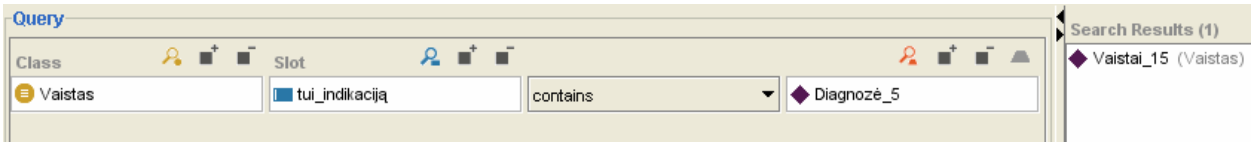
Ontologijos sukūrimui buvo atliktas tiriamasis darbas, kuriuo metu buvo išanalizuotos dvi ontologijos. Buvo ieškomos vaistų dalykinės srities ontologijos, kuriamos ontologijos papildymui, t. y. pakartotinių ontologijų panaudojimui, tačiau surasti panašių vaistų ontologijų nepavyko. Todėl buvo išnagrinėtos dvi kitų dalykinių sričių ontologijos – vynų ir maisto ontologija bei picų ontologija. Šių ontologijų dalykinės sritys skiriasi nuo pasirinktos – vaistų dalykinės srities, tačiau ontologijos formavime svarbiausia yra taisyklingos hierarchijos suformavimas, taigi sritis gali skirtis, bet pagrindiniai principai išlieka.

6.2 Protege - ontologijos kūrimo įrankis

Mokslo tiriamajame projekte ontologijos aprašymui buvo parinktas dažniausiai naudojamas įrankis Protege 3.2.1. Šis įrankis duoda galimybę sukurti ontologiją nuo pirmųjų žingsnių, kai tik formuojama hierarchija, be to, vystyti ontologiją, įvedant papildomus apribojimus bei ryšius. Protege įrankis turi patogu valdymą bei automatiškai sukuria OWL kodą, priklausomai nuo atliktų ir išsaugotų veiksmų ontologijoje. Tokiu būdu sukurta ontologija gali būti panaudota programose, kurių kalbos turi OWL biblioteką, arba kituose ontologijos įrankiuose.

Sukurta ontologija naudojama reikiamos informacijos gavimui. Paprasčiausias būdas atsakyti į pateiktus klausimus yra panaudoti Protege 3.2.1 Query įrankį. Tokios užklausos pavyzdys gali būti „surasti vaistų pavadinimą, kuris yra nurodomas pagal tam tikrą diagnozę“ (žr. 4 pav.).

4 pav. „Užklausos realizavimas panaudojant Protege 3.2.1 Query įrankį“:



Šis įrankis leidžia formuoti sudėtingesnes užklausas, bei saugoti jas failuose sekančiam panaudojimui kitose programose, kurios palaiko OWL bibliotekas.

6.3 Svarbių ontologijos terminų išvardinimas

Naudinga sudaryti visų terminų, kuriuos reikia paaiškinti vartotojui, sąrašą. T. y. kokius terminus norima išnagrinėti, kokias savybes turi šie terminai, ką reikia papildomai pasakyti apie šiuos terminus. Vaistų ontologijoje visa dominanti informacija susieta su vaistais, todėl pirmiausiai reikia sudaryti pilną sąrašą sąvokų, nesirūpinant dėl konkrečių terminų sąryšių, savybių sąvokų ir klasių aprašymų.

Sekantys du žingsniai yra klasių hierarchijos kūrimas bei savybių sąvokų aprašymas. Šie žingsniai glaudžiai susiję tarpusavyje. Sunku atlikti pirmiausiai vieną iš jų, o paskui kitą. Paprastai suformuluojamos kelios hierarchijos sąvokos, po to aprašomos šių sąvokų savybės ir t. t. Be to, minėti du žingsniai yra svarbiausi žingsniai ontologijos projektavimo procese.
Darbe nagrinėjama vaistų ontologija, todėl terminų sąrašą sudarė tokie terminai kaip:
Cheminiai preparatai, vaistai, naudojami vaistai, gamintojai, gamintojų licenzijų galiojimas, liga, ligos aprašymai, ligos simptomai, gydytojo diagnozė, vaistų naudojimas pagal receptą, šalutiniai poveikiai ir t. t.

7 Ontologijos surinkimas

7.1 Klasių bei hierarchijos klasių apibrėžimas

Hierarchiją konstruojama pagal vieną iš galimų konstravimo būdų:

- Nusileidžiančio proceso kūrimas prasideda nuo bendriausių dalykinės srities sąvokų apibrėžimų. Kiekviename žingsnyje sąvokos konkretizuojamos;
- Kylančio proceso kūrimas prasideda nuo konkrečiausių dalykinės srities klasių, hierarchijos lapų apibrėžimų. Kiekviename žingsnyje klasės grupuojamos į bendresnes sąvokas;

- Kombinuoto proceso kūrimas – tai nusileidžiančio ir kylančio procesų derinys. Pirmiausiai aprašomos svarbesnės sąvokos, toliau jos apibendrinamos bei tikslinamos, aprašant sąvokų apribojimus.

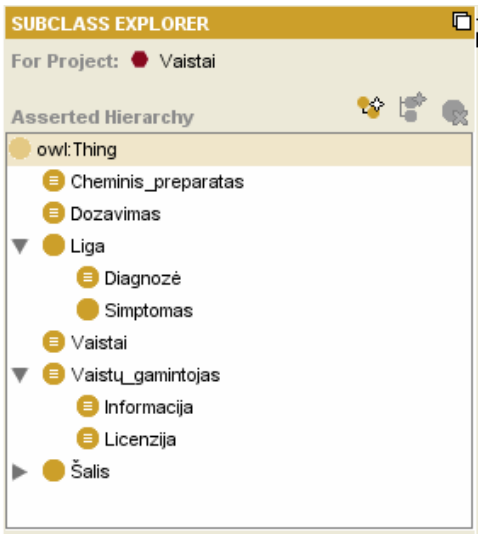
Nei vienas iš aprašytų procesų nėra geriausias ir nėra geresnis už kitą. Proceso pasirinkimas priklauso nuo asmeninio požiūrio į dalykinę sritį. Dažniausiai praktikoje naudojamas kombinuotas procesas, nes sąvokos, kurios yra hierarchijos „viduryje“ būna akivaizdžiausios dalykinėje srityje. Darbe buvo pasirinktas kombinuotas ontologijos kūrimo procesas, nes naudojant šį procesą nebūtina nagrinėti pirmaisiais konkrečiausius arba bendriausius objektus, juos galima aprašyti darbo eigoj, pildant hierarchiją.

Pirmiausiai buvo apibrėžtos klasės, nepriklausomai nuo pasirinkto proceso. Iš sudaryto ontologijos terminų sąrašo buvo išrinkti terminai, kurie toliau aprašė objektus, egzistuojančius nepriklausomai, o ne terminai, kurie aprašo šių objektų savybes. Šie terminai ontologijoje yra klasės, jos priklauso ontologijos hierarchijos klasėms. Klasės įtraukiamos į hierarchiją panaudojant taisyklę:

Jeigu klasė A – yra viršklasė B klasės, tai kiekvienas B egzempliorius taip pat yra A egzempliorius.

Taigi, darbe nagrinėjama ontologija turi sekančias klases, viršklases bei poklasius:

5 pav. „Ontologijos hierarchija Protege 3.2.1 įrankyje“:



7.2 Klasių savybių apibrėžimas

Savybės yra 2-jų kategorijų:

- objekto savybės (Object properties), kurios suriša individualinius objektus tarpusavyje;
- duomenų tipų savybes (Datatype properties), kurios suriša individualinius elementus su duomenų tipų (sveikieji skaičiai, skaičiai su kablelių ir eilutės) reikšmėmis. Duomenų tipų nustatymui OWL naudojama XML schema.

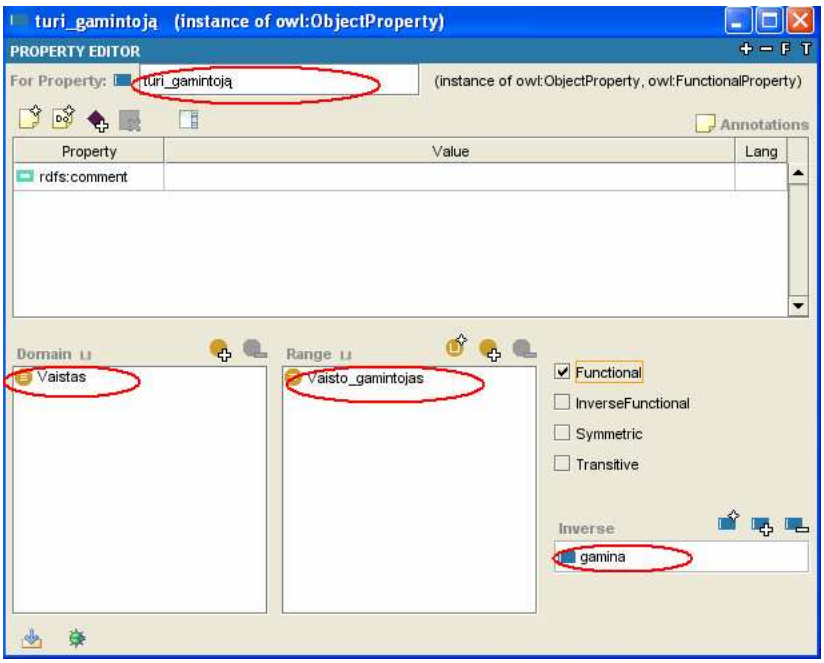
Pačios savaime klasės neduoda pakankamos informacijos atsakymams į klausimus. Po kelėtų klasių aprašymo, reikia aprašyti vidinę sąvokų struktūrą.

Terminų sąraše po klasių formavimo liko šių klasių savybės (vaisto pavadinimas, gamintojo pavadinimas, licenzijos numeris, dozavimo kiekis ir t. t.) . Todėl kiekvienai savybei reikia nustatyti kokiai klasei ji priklauso.

Ontologijoje savybės gali būti kelių tipų:

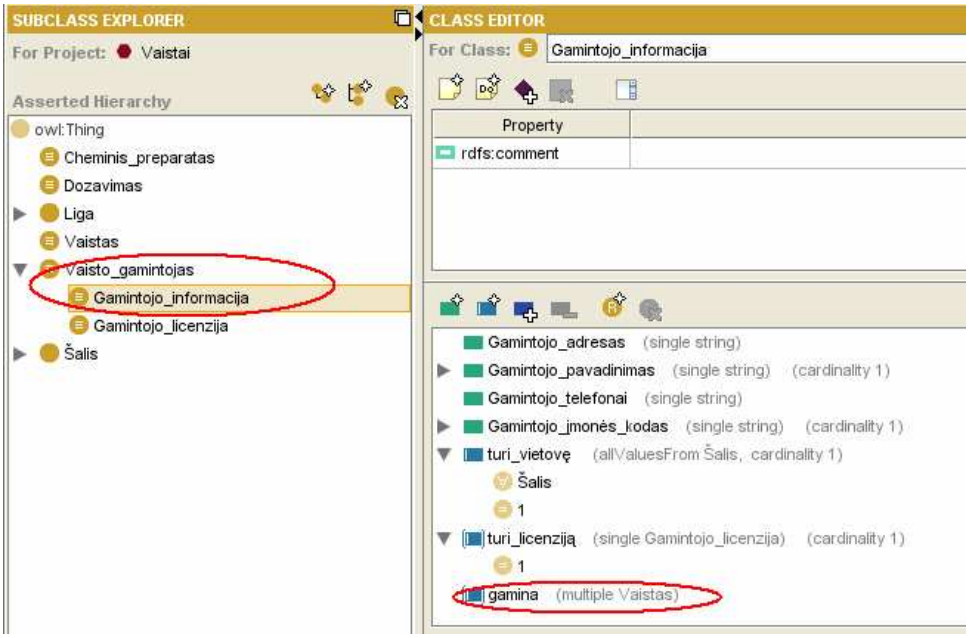
- „vidinės“ savybės, tokios kaip dozavimo dažnumas;
- „išorinės“ savybės, tokios kaip vaistų pavadinimas;
- Sąryšis tarp kitų individ. konceptų, tai sąryšis tarp atskirų klasių narių ir kitų elementų. Pavyzdžiui, savybė „turi gamintoją“ – tai sąryšis tarp Vaistų klasės ir Vaistų gamintojo, kuris gamina šiuos vaistus.

6 pav. „Sąryšių aprašymas Protege 3.2.1 įrankyje“:



Visi klasės poklasiai paveldi šios klasės savybę. Pavyzdžiui visas Vaistų gamintojo klasės savybes paveldės šios klasės poklasiai.

7 pav. „Sąryšių paveldėjimas Protege 3.2.1 įrankyje“:



Savybė reikia pririšti prie bendriausios klasės, kuri gali turėti šią savybę.

7.3 Savybių facetų apibrėžimas

Sekantis žingsnis ontologijos kūrime yra savybių falcetų apibrėžimas. Kiekvienai vaistų ontologijos savybei apibrėžiami falcetai. Savybės gali turėti skirtingus facetus, kurie aprašo reikšmių tipą, galimas reikšmes, reikšmių kiekį (galia) ir kitas reikšmių savybes.

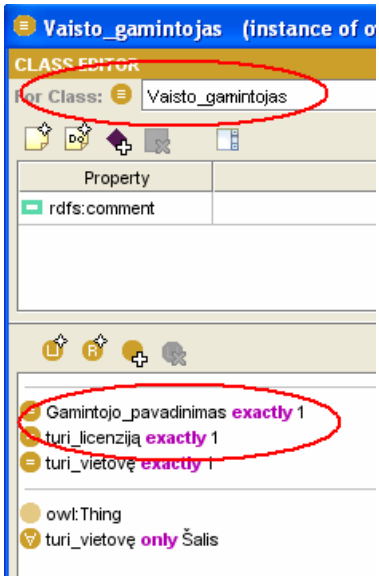
Pavyzdžiui, viena iš vaistų ontologijos savybių yra „Gamintojo pavadinimas“. Šios savybės reikšmė yra viena eilutė su tipu String. T. y. „pavadinimas“ tai savybė su String tipu. Savybė „gamina“ gali turėti daugybinę reikšmę, kuri yra Vaistų klasės egzempliorius. T. y. „gamina“ tai savybė, turinti egzemplioriaus tipą ir leidžiamą klasę Vaistai.

7.3.1 Savybės galia

Savybės galia aprašo savybės reikšmių kiekį. Galia gali būti pavienė (galima tik viena reikšmė) ir daugybinė (galimas bet koks reikšmių kiekis).

Tam tikros sistemos leidžia aprašyti minimalią ir maksimalią galią, tokiu būdu galima tiksliau aprašyti reikšmių kiekį. Darbe naudojama sistema Protege 3.2.1, leidžia nustatyti reikšmių diapazoną. Minimali galia N reiškia, kad savybė turi turėti mažiausiai N reikšmių. Tokiu būdu, nustatoma savybių galia toms savybėms, kurios turi kažkokį reikšmių apribojimą: visi pavadinimai (vaistų, lygų, gamintojų) būtinai turi nors vieną pavadinimą. Kai kurie iš šių savybių turi vienintelį pavadinimą.

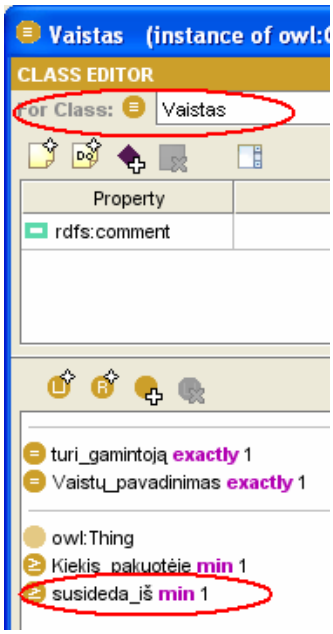
8 pav. „Savybės tikslios galios vaizdavimas Protege 3.2.1 įrankyje“:



Be to, gamintojas turi būtinai turėti vieną licenzijos numerį.

O savybė „susideda iš“ turi minimalią galią 1: kiekvienas vaistas turi būti sudarytas bent iš vieno cheminio preparato.

9 pav. „Savybės minimalios galios vaizdavimas Protege 3.2.1 įrankyje“:

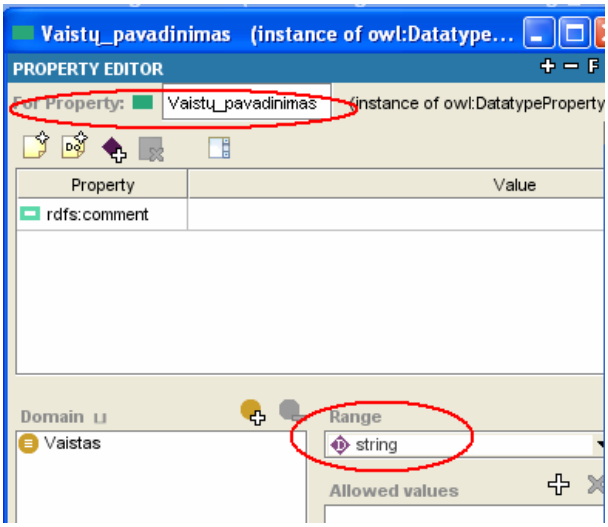


7.3.2 Savybės reikšmių tipas

Aprašant ontologiją, reikia nustatyti kokių tipų reikšmes turės vaistų, ligų, gamintojų savybės. Reikšmės tipo facetas aprašo reikšmių tipus, kurias galima įvesti į savybę. Reikšmių tipų sąrašas:

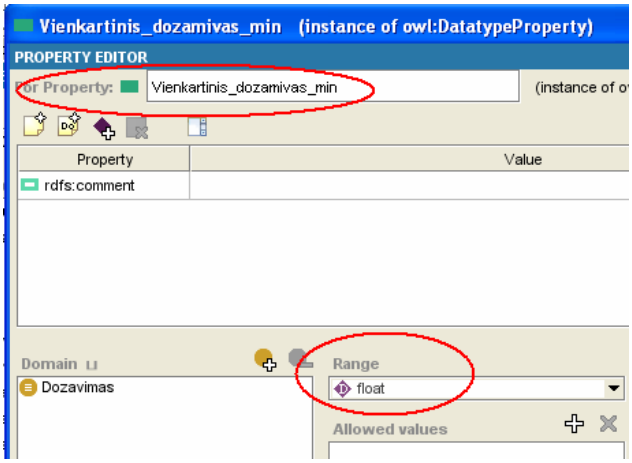
- String – paprasčiausias reikšmės tipas, naudojamas tokiose savybėse kaip „pavadinimas“: reikšmė yra paprasta eilutė;

10 pav. „*String* reikšmės tipo priskirimas Protege 3.2.1 įrankyje“:



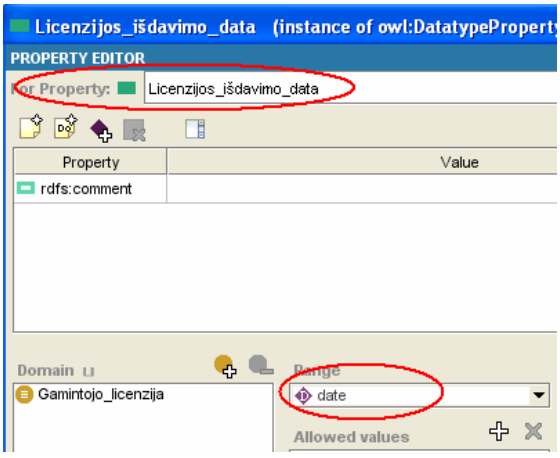
- Float ir Int – aprašo savybes, turinčias skaitmenines reikšmes. Pavyzdžiui, savybė „vienkartinis minimalus dozavimas“ turi Float tipą;

11 pav. „*Float* reikšmės tipo priskirimas Protege 3.2.1 įrankyje“:



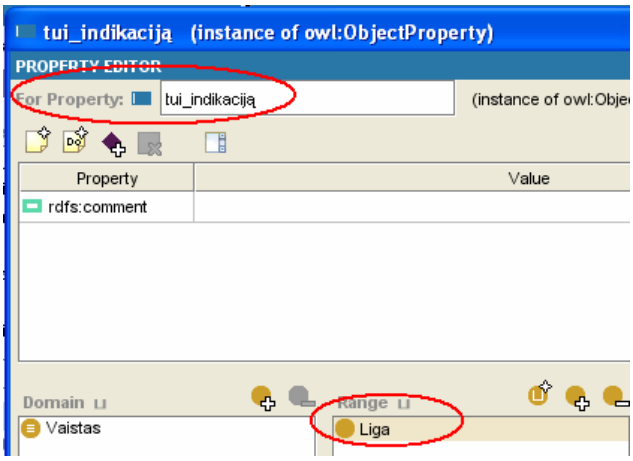
- Boolean – tai savybės su reikšme „taip“ arba „ne“ (darbe šis facetas dar nebuvo naudotas);
- Date – tai savybė su datos reikšme. Licenzijos išdavimo datos savybė turi šį tipą:

12 pav. „*Date* reikšmės tipo priskirimas Protege 3.2.1 įrankyje“:



- Savybės-egzemplioriai – aprašo sąryšius tarp individ. konceptų. Savybės, turinčios egzempliorių reikšmių tipą irgi turi turėti galimų reikšmių, t. y. klasių sąrašą, kurių egzempliorius galima naudoti. Tokios savybės yra „turi indikaciją“, pagal šią savybę kiekvienas vaistas išrašomas pagal tam tikras ligas:

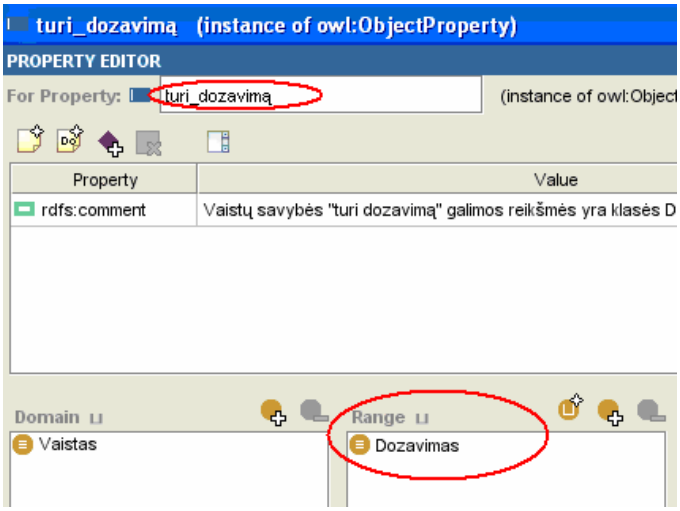
13 pav. „Savybės-egzemplioriaus vaizdavimas Protege 3.2.1 įrankyje“:



7.3.3 Savybės domenas ir savybės reikšmių diapazonas

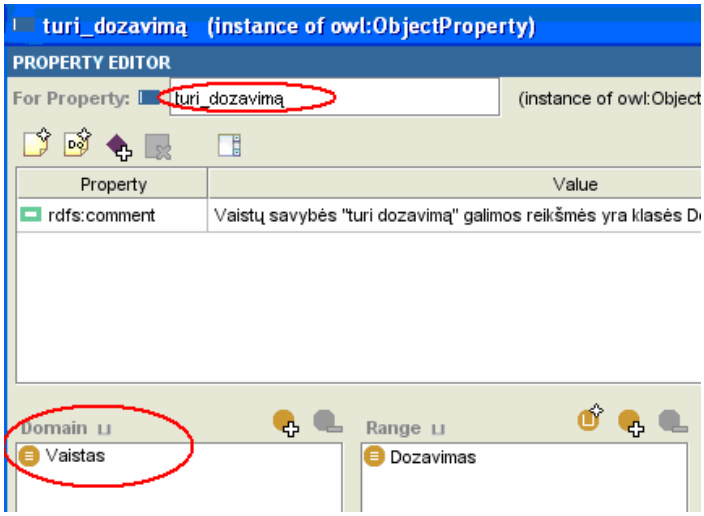
Ankstesniame poskyryje buvo aprašytas vienas iš savybės reikšmių tipų, kaip kitos klasės egzempliorius. Tokių klasių egzemplioriai literatūroje vadinami savybės reikšmių diapazonu. 14 paveikslėlyje pavaizduotas savybės „turi dozavimą“ aprašymas:

14 pav. „Savybės-egzemplioriaus vaizdavimas Protege 3.2.1 įrankyje“:



Šios savybės domenas yra klasė Vaistai. Savybės domenas – tai klasė, prie kurios yra pririšta savybė:

15 pav. „Demono vaizdavimas Protege 3.2.1 įrankyje“:



Pagrindinės savybių domenų ir reikšmių diapazonų nustatymų taisyklės yra panašios: Apibrėžiant domeną arba savybės reikmių diapazoną randamos bendriausios klasės, kurios gali būti domenu arba savybių reikšmių diapazonu atitinkamai.

Iš kitos pusės, domenas ir savybių reikšmių diapazonas neturi būti per daug bendras. Visos klasės savybės domene turi būti aprašytos su šia savybe, o visų klasių egzemplioriai savybės reikšmių diapazone, turi būti potencialia užpildomos pagal šią savybę.

Jeigu klasių sąrašas, aprašantis savybės reikšmių diapazoną arba savybės domeną turi klasę ir jos poklasį – toks poklasis naikinamas. T. y. jeigu savybės reikšmių diapazonas turi klasę Liga ir Diagnozė, tai Diagnozės diapazoną reikia panaikinti iš sąrašo, nes ši klasė neduoda papildomos informacijos (Diagnozė yra poklasis Ligos klasės).

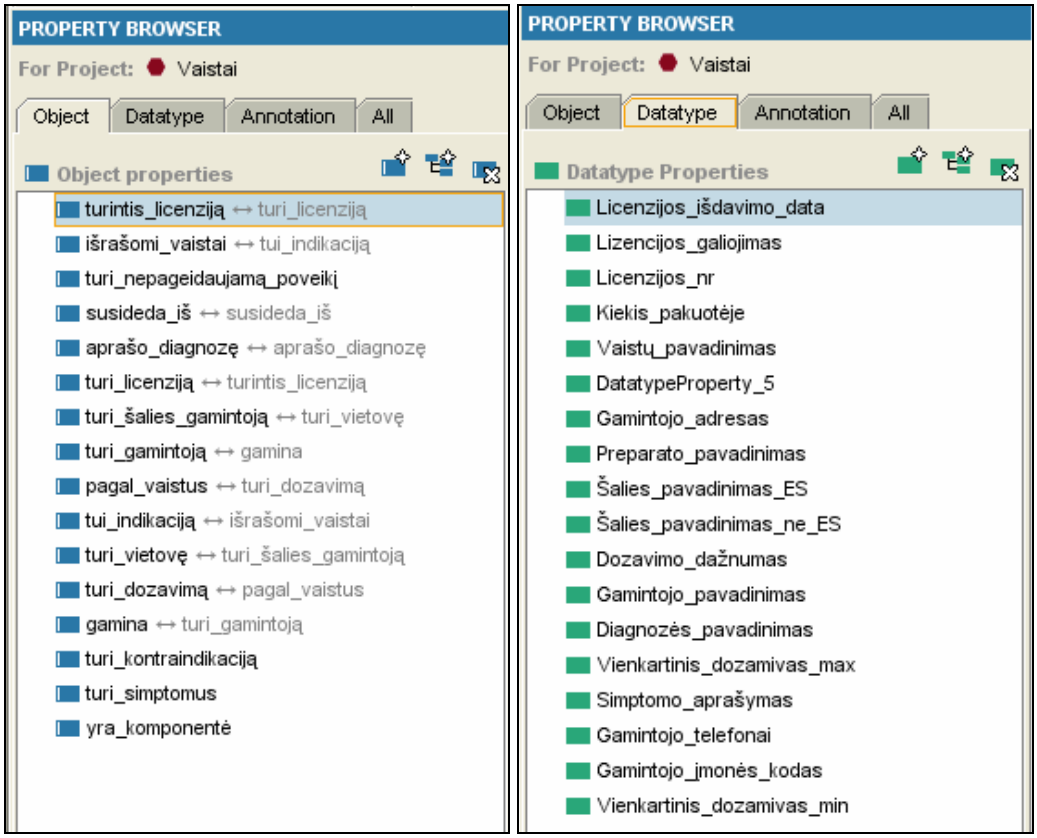
Toliau aprašant savybes vaistų ontologijoje, buvo pritaikytos sekančios taisyklės:

- Jeigu klasių sąrašas, aprašantis savybės reikšmių diapazoną arba domeną turi visus klasės A poklasius, bet neturi pačios klasės A, tai į reikšmių diapazoną įtraukiama klasė A, o jos poklasiai panaikinami iš reikmių diapazono.

- Jeigu klasių sąrašas, aprašantis savybės reikšmių diapazoną arba domeną turi beveik visas klasės A poklasiui, galbūt reikšmių diapazoną geriau aprašyti kaip A klasę.

Taigi, darbe nagrinėjamoje ontologijoje buvo sukurtos sekančios savybės:

16 pav. „Vaistų ontologijos savybės Protege 3.2.1 įrankyje“:



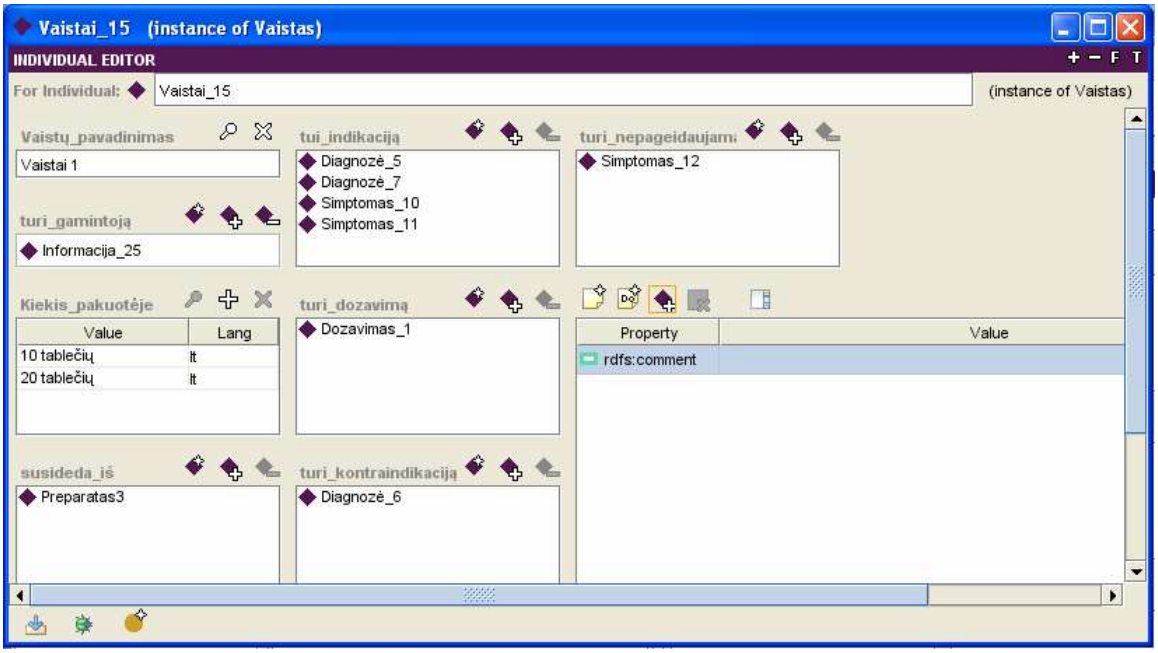
7.4 Vaistų dalykinės srities klasių egzempliorių kūrimas

Paskutinis žingsnis tai klasių egzempliorių kūrimas hierarchijoje. Atskiros klasės egzemplioriaus aprašymui reikia:

1. pasirinkti klasę;
2. sukurti atskirą egzempliorių;
3. įvesti savybių reikšmes.

Kiekvienai klasei (Vaistams, Diagnozėms, Simptomams, Gamintojams, Šalims) buvo sukurti egzemplioriai bei užpildyti šių klasių savybių reikšmės pagal nustatytą diapazoną. Vaistų klasės egzemplioriaus pavyzdys su užpildytomis savybėmis:

17 pav. „Vaistų ontologijos egzempliorius Protege 3.2.1 įrankyje“:



8 Ontologijos hierarchijos teisingumo užtikrinimas

Sudarant klasių hierarchiją, galima pridaryti klaidų, kurias bus sunku ištaisyti, todėl reikia žinoti hierarchijos kūrimo taisykles ir tikrinti hierarchiją po kiekvieno naujo žingsnio. Kaip jau minėjau, nėra vienintelės teisingos klasių hierarchijos bet kuriai dalykinei sričiai. Hierarchija priklauso nuo galimų ontologijos panaudojimo būdu, detalizavimo lygiu, reikalingu programai, o kartais nuo suderinimo su kitais modeliais. Bet yra keletą esminių, pagrindinių principų, kuriuos reikia turėti omenyje kuriant klasių hierarchiją. Sukonstravus tam tikrą naujų klasių kiekį reikia patikrinti, turimą hierarchiją pagal šiuos principus.

8.1 Hierarchijos klasių teisingumo aprūpinimas

Santykis „is-a“:

Klasių hierarchija vaizduoja santykį „is-a“: klasė A – tai poklasis B klasės, jeigu kiekvienas egzempliorius B klasės yra A klasės egzempliorius. Pavyzdžiui Europos Sąjungos šalis yra taip ir visų Šalių egzempliorius. Išvada: Šalių hierarchija sudaryta teisingai.

Dažnai pasitaikanti modeliavimo klaida – yra hierarchijos papildymas viena ir ta pačia sąvoka vienaskaitoje ir daugiskaitoje, taip kad sąvoka vienaskaitoje yra antros sąvokos poklasis. Pavyzdžiui klaidingai būtų apibrėžti klasę Vaistas kaip klasės Vaistai poklasį. Tinkamiausias

būdas išvengti tokias klaidas yra – visada naudoti klasių pavadinimus arba tik vienaskaitoje arba daugiskaitoje.

Minėtų klaidų išvengimui vaistų ontologijoje klasės pavadinimai aprašytos vienaskaitoje (Vaistas, Gamintojas, Liga ir t. t.).

8.2 Klasių vardai

Svarbu yra atskirti klasę ir jo vardą:

Klasės aprašo dalykinės srities sąvokas, o ne žodžius, kurie išreiškia šias sąvokas. T. y. aspirinas ir analginas negali būti Vaistų klasėmis, nes tai yra Vaistų klasės egzemplioriai ar net savybės „Vaisto pavadinimas“ reikšmė.

Klasės vardas gali pasikeisti, jeigu bus pasirinkta kita terminologija, bet pats terminas išreiškia objektyvią pasaulio realybę.

Iš esmės, reikia visada laikytis taisyklės: vienos ir tos pačios sąvokos sinonimai neaprašo skirtingas klases. Tokius sinonimus galima aprašyti ontologijos projekto dokumentacijoje, arba padaryti klasių asociacijų sąrašą.

8.3 Ciklišků klasių išvengimas

Hierarchijoje klasės neturi būti cikluose. Hierarchijoje yra ciklas, jeigu tam tikra klasė A turi poklasį B, tuo metu kai klasė B turi poklasį A. Tokio ciklo reikšmė yra tokia, kad klasės A ir B yra ekvivalenčios, todėl jos neturi prasmės. Darbe panaudota sistema Protege 3.2.1 automatiškai stebi tokius ryšius bei neleidžia jiems atsirasti.

8.4 Klasių/poklasių kiekis

Nėra griežtų taisyklių dėl tiesioginių poklasių kiekio vienai klasei. Tačiau dažniausiai ontologijos, turinčios tiksliai apibrėžtą struktūra turi nuo 2-jų iki 12 tiesioginių poklasių. Išplaukia du vadovaujantys principai:

1. Jeigu klasė turi vienintelį poklasį, tai tikriausiai buvo padaryta modeliavimo klaida arba ontologija yra nepilna;
2. Jeigu klasė turi daugiau už 12 poklasių, tai tikriausiai reikalingos papildomos tarpinės kategorijos.

Tačiau, jeigu nėra natūralių klasių sąvokų grupavimui, nereikia kurti dirbtines klases. T .y jeigu pasaulyje nėra šių sąvokų atskirų kategorijų, tai turi būti atvaizduota ontologijoje.
Pavyzdžiui, nėra tikslo sudarinėti dirbtinę klasę (klases) klasėms Liga, Vaistas, Dozavimas ir Cheminiai preparatai, nes šios klasės neturi natūralios vieningos kategorijos ir vienintelė vienoda savybė gali būti pavadinimas, kuris pagal prasmę yra visiškai skirtingas klasių sąvokoms.

8.5 Daugybinis paveldėjimas

Protege 3.2.1 sistema leidžia aprašinėti hierarchijos klasių daugybinį paveldėjimą: klasė gali būti kelių klasių poklasis. Pavyzdžiui, jeigu aprašyti detalesnius vaistų ontologijos lygius, tai klasė Vaistas gali turėti poklasius Stiprus vaistas ir Vaistas tabletėmis, be to gali prireikti aprašyti papildomą klasę Stiprus vaistas tabletėmis. Stiprus vaistas tabletėmis yra ir stiprus, ir tablečių pavidalo, todėl reikėtų aprašyti, kad nauja klasė turės du viršklases. Visi naujos klasės egzemplioriai bus abiejų viršklasių egzemplioriai. Be to nauja klasė paveldės abiejų viršklasių savybes.

8.6 Naujos klasės įvedimo kriterijus

Vienas iš sudėtingiausių sprendimų, kuriuos reikia priimti ontologijos kūrimo procese yra nuspręsti kada įvesti naują klasę ir aprašyti skirtumus naudojant įvairių reikšmių savybes. Yra keletas praktinių būdų, kurie naudojami nustatant ar reikalinga nauja klasė hierarchijoje:

1. Jeigu klasės poklasiai turi papildomas savybes, kurių neturi jų viršklasė;
2. Jeigu klasės poklasiai turi apribojimus, skirtingus nuo viršklasės apribojimų;
3. Jeigu klasės poklasiai turi kitus santykius, negu jų viršklasė.

Kitaip tariant, hierarchija papildoma nauja klase tik tada, kai apie klasę galima pasakyti kažką tokį, ko negalima pasakyti apie jos viršklasę. Pagal šiuos principus ir buvo sudaryta vaistų klasių hierarchija.

8.7 Naujos savybės įvedimo kriterijus

Hierarchijos klasės neprivalo turėti naujas savybes. Pavyzdžiui, yra ontologijų, kurios turi dideles paprastų dalykinės srities terminų hierarchijas, kurios neturi savybių (arba turi vienodą savybių rinkinį). Šiuo atveju taip pat naudinga sudėlioti terminus į hierarchiją, nes tai palengvins navigaciją ir padės vartotojui pasirinkti tinkamą terminų lygį. Taip pat ir vaistų ontologijoje, sudaryta klasė Liga bei jos poklasiai Diagnozė ir Simptomai. Ligos klasė neneša jokios naudingos informacijos, bet ją patogu sukurti diagnozes ir simptomus nuo kitų terminų.

8.8 Pasirinkimas tarp naujos klasės ir savybės reikšmės

Modeliuojant dalykinę sritį, dažnai reikia spręsti ar tam tikrą apribojimą (tabletė, kapsulė, butelis) reikia modeliuoti kaip savybės reikšmę arba kaip klasių rinkinį, ir vėlgi tai priklauso nuo dalykinės srities masto.

Jeigu sąvoka su skirtingomis savybės reikšmėmis tampa apribojimu skirtingoms savybėms kitose klasėse, tai atskirimui reikia sukurti naują klasę. Priešingu atveju atskirimas

daromas savybių reikšmėse. Taip, vaistų ontologijoje sukurti du skirtingi šalies poklasiai, kurių egzemplioriai priklauso arba nepriklauso Europos Sąjungai.

Dažniausiai spalvos, skaičiai, vietovės yra sąvokų reikšmės ir nesukuria naujas klases. Kaip ir vaistų ontologijoje visos aprašytos šalis turi tik pavadinimo sąvoką, kurioje aprašomas šalies pavadinimas, tam nereikia kurti klases kiekvienai šaliai.

8.9 Disjunktiniai poklasiai

Klasės yra disjunktinės, jeigu jos negali turėti bendrų egzempliorių. Pavyzdžiui, vaistų ontologijoje ES šalies ir ne ES šalies klasių egzemplioriai yra disjunktiniai. Nes viena šalis gali būti arba iš Europos Sąjungos are nepriklausyti jai. Disjunktinių klasių aprašymas duoda galimybę geriau tikrinti ontologijos teisingumą.

Šio tikslo realizavimui Protege 3.2.1 buvo sudarytos skirtingos savybių reikšmių sąrašas, tokiu būdu šalį galima pasirinkti arba iš vienos, arba iš kitos kategorijos.

18 pav. „Disjunktinių klasių reikšmių pasirinkimas Protege 3.2.1 įrankyje“:

9 Ontologijos masto rėžiai

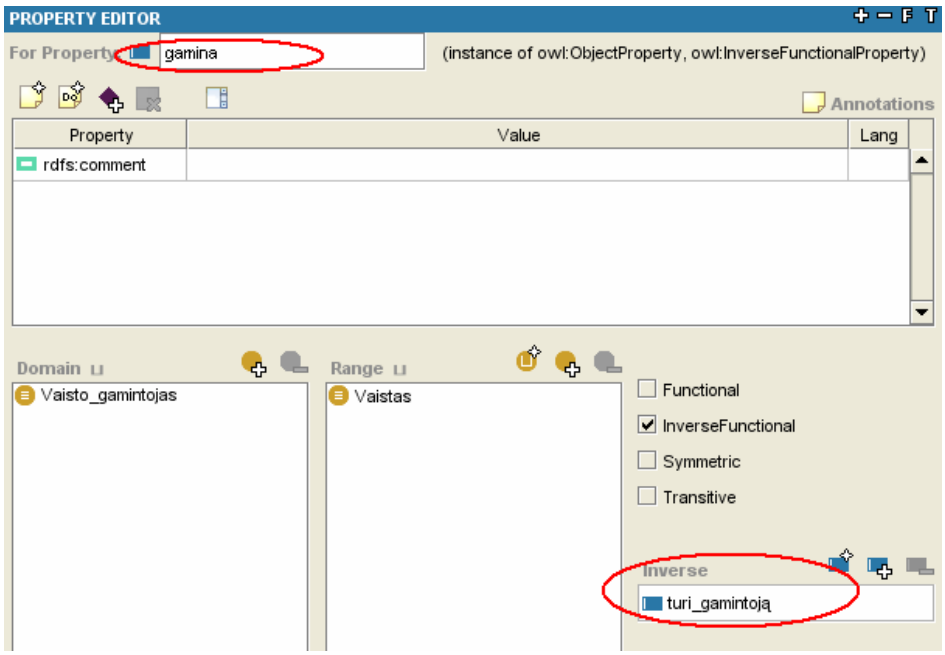
Kuriant ontologiją svarbu nustatyti, kada reikia baigti pildyti hierarchiją ir kurti naujas klases. Ontologija neturi kaupti visos įmanomos dalykines srities (vaistų) informacijos. Nereikia konkretizuoti arba apibendrinti daugiau, negu reikalauja programa, kuri naudos kuriamą ontologiją (nedaugiau vieno papildomo lygio į kiekvieną pusę). Be to, ontologijoje neturi būti visos įmanomos klasių savybės.

Vaistų ontologija be abejo neturi visas savybes, kurias galėtų turėti vaistai ir lygos. Sukurtoje ontologijoje buvo įdėtos pačios pagrindinės ir svarbiausios klasių savybės, kurios reikalingos siūlant preparatą pacientui, atsižvelgiant į jo diagnozę ir simptomus.

10 Atvirkštinės sąvokos

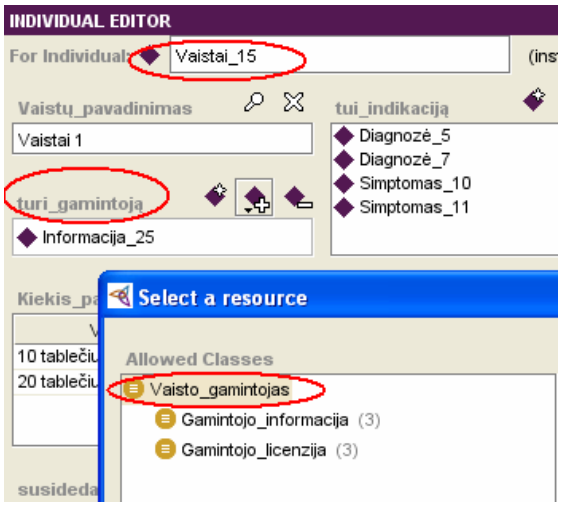
Savybės reikšmė gali priklausyti nuo kitos savybės reikšmės. Jeigu vaistai buvo pagaminti vaistų gamintoju, tai vaistų gamintojas pagamino šiuos vaistus, tokiu būdu „turi gamintoją“ ir „pagamina“ yra atvirkštiniai santykiai.

19 pav. „Atvirkštinių savybių vaizdavimas Protege 3.2.1 įrankyje“:



Nenaudinga saugoti informaciją apie šias 2 savybes. Tačiau, naudinga turėti šiuos blokus prieinamus, kad būtų galimybė dirbti su duomenimis iš bet kurios pusės. Tokiu būdu, vartotojas turės galimybę nustatyti vaistų gamintoją, aprašant vaistus. Iš kitos pusės, jeigu užpildoma gamintojo informacija, gaminamus vaistus galima pasirinkti gamintojo pusėje.

20 pav. „Egzemplioriaus reikšmių pildymas klasėms su atvirkštinėmis savybėmis Protege 3.2.1 įrankyje“:



11 Ontologijų analizė

Šiuo metu skiriama didelė svarba ne tik ontologijos kūrimui, bet ir ontologijos analizei. Kuo daugiau ontologijų bus sukurta ir pakartotinai panaudota, tuo daugiau bus instrumentinių įrankių ontologijos analizei. Ontologijos analizė gali būti atlikta panaudojant specialias tikrinimo priemones (angl. reasoner). Tokių įrankių pagalba galima tikrinti ontologijos loginį teisingumą bei daryti tipinių ontologijos kūrimo metų padarytų klaidų diagnostiką. Atrastas ontologijos klaidas reikia taisyti. Po šių veiksmų ontologijos analizė turi būti atlikta pakartotinai.

Pagrindiniai šiuo metu naudojami OWL ontologijų tikrinimo priemonių produktai yra išnagrinėti šiame darbe. Yra pristatytas šių produktų funkcionalumo analizė, privalumai bei trūkumai, be to padaryta šių produktų palyginimo analizė.

Atliekant tiriamąjį darbą, nebuvo rasta galimybių pratestuoti ir panagrinėti realųjį tikrinimo priemonių veikimą, dėl esamų produktų neužbaigtumo bei vartotojo dokumentacijos trūkumo. Todėl buvo nuspręsta atlikti produktų analizę pasinaudojant tikrinimo priemonių aprašymais.

Darbe pristatytos keturios OWL ontologijos tikrinio priemonės: FaCT++, KAON2, Pellet ir RacerPro.

11.1 FaCT++ tikrinimo priemonė

FaCT ++(greita terminologijų klasifikacija iš angl. Fast Classification of Terminologies) FaCT++ tai loginio OWL DL[1] išvedimo mechanizmas, kuris realizuotas su C++. FaCT++ yra naujos kartos tikrinimo priemonė, kurios tėvas yra FaCT OWL DL tikrinimo priemonė.

FaCT - yra DL klasifikatorius, kuris dar gali būti panaudotas testuojant modalinės logikos [2](žr. apačioj) įvertinimą.

Be to, FaCT++ realizuotas panaudojant C++, efektyvesnės programinės įrangos sukūrimui bei mobilumo padidinimui. [DT06]

[1] OWL DL (aprašymo logika iš angl. Description Logic) – šis OWL detalizavimo lygis skirtas vartotojams, kuriems reikalingos maksimalios kalbos galimybės, be apskaičiavimų bei semantinių išvadų praradimų. OWL DL lygis orientuotas į šiandien egzistuojančias žinių aprašymo sistemas bei į loginio programavimo sistemas.

[2] Modalinė logika – tai logikos sritis, kurioje tyrinėjami modalumai, apskaičiavimų kūrimai, kuriuose modalumų pasakymai taikomi kartu su loginėmis operacijomis.

11.1.1 FaCT++ ypatybės

FaCT++ gali būti panaudotas sekančiais tikslais:

- Tikrinant ontologijos neprieštaringumą;
- Tikrinant vienos klasės/grupės sąvokų įvedimą;
- Tikrinant hierarchijos santykius tarp dviejų klasių;
- Klasifikuojant pilną ontologiją (kuriant taksonomiją[3]);
- DIG[4] sąsają, numatančią priėjimą iš įrankių, tokių kaip Protege;
- Sąsają programiniam valdymui, ontologijoms realizuotoms OWL-DL, SWRL[5] ir F-Logic[6] aplinkose; [DI04]

11.1.2 FaCT++ dokumentacija

Vartotojo instrukcijos arba įrankio naudojimo dokumentacijos produkto kūrėjai kol kas neplatina.

11.1.3 FaCT++ problemos

Ribotas egzempliorių palaikymas:

Šiuo metu FaCT++ turi ribotą ontologijos egzempliorių palaikymą. Visi egzemplioriai laikomi kaip klasės ir tikrinimas atliekamas modifikuotoje ontologijoje. Toks principas išvengia išvadų praradimą, jeigu ontologija neturi ryšių tarp egzempliorių. Net tuo atveju, kai santykiai tarp egzempliorių egzistuoja, kartais įmanoma gauti teisingą atsakymą į užklausą dėl hierarchijos santykių. Šiuo atveju, FaCT++ duoda atsakymą kartu su teisingumo informaciją.

Domenų rolė ir diapazono sumažinimas:

Jeigu patikrinimo priemonė nepalaiko ontologijų reikšmių diapazonų ir domenų konstravimo (kaip FaCT), ji turi traktuoti juos kaip bendras aksiomas, kurių esmė lieka nepakeista. Tai smarkiai sulėtina tikrinimo procesą. FaCT++ buvo realizuotos konstrukcijos, kurios pagreitina tokio tikrinimo procesą dvigubai kai kuriose ontologijose. Be to, nauja pakeitimo metodika

[3] Taksonomija – hierarchinių pavidalų sukonstruota tikslų bei rezultatų sistema, kur konstravimas vyksta nuo paprasto iki sudėtingo lygio. Taksonomijų kūrimas duoda naujas perspektyvas įvairių sferų testavimams.

[4] DIG (DL Implementation Group) DL realizavimo grupė tai tyrinėtojų ir kūrėjų surinkimas, susijęs su DL sistemų diegimais. DIG nariai turi nereguliarus neoficialius susitikimus dėl įvykių susijusių su DL, tokius, kaip kasmetinis DL seminaras. Pagrindinis DIG indėlis – yra DIG sąsaja. Ji tiksliai nusako bendrą sąsają DL tikrinimo priemonėms ir leidžia įterpinėti jas į ontologijos redaktorius, kad DL tikrinimo priemonės būtų laisvai prieinamos ir naudojamos. Sąsajos specifikacijos apibrėžia sąvokų kalbą ir minimalų operacijų seką.

[5] SWRL (Semantic Web Rule Language) – semantinio voratinklio taisyklių kalba kuri derina OWL ir RuleML.

[6] F-logic (frame logic) –tai žinių bei ontologijos vaizdavimo kalba.

pagrįsta domenų bei diapazonų konstravimu buvo patobulinta ir tikrinimo efektyvumas akivaizdžiai padidėjo. [DI04]

11.1.4 FaCT++ panaudojimas

FaCT++ suteikia galimybę naudoti savo technologiją per Protege-OWL redaktorių.

11.2 KAON2 tikrinimo priemonė

KAON2 – ontologijos OWL-DL, SWRL ir F-Logic valdymo infrastruktūra. Užklausos gali būti suformuluotos panaudojant SPASQL[7]. KAON2 buvo pilnai realizuotas su Java 1.5.

11.2.1 KAON2 ypatybės

- Sąsają programiniam valdymui, ontologijoms realizuotoms OWL-DL, SWRL ir F-Logic aplinkose;
- Automatinį serverį teikiantį priėjimą prie ontologijų paskirstytuose metoduose naudojant RMI;
- DIG sąsają, numatančią priėjimą iš įrankių, tokių kaip Protege;
- Modelį ontologijos egzempliorių išskleidimui iš reliacinės duomenų bazės.

KAON2 sąsaja yra patogi atliekant manipuliacijas su OWL-DL ontologijomis. Šiuo metu, sąsaja supranta OWL RDF sintaksę.

KAON2 palaiko galimybę atsakinėti į sujungtas užklausas, nors ir užklausoje nėra tiksliai aprašytų kintamųjų . Tai reiškia, kad visi užklausos kintamieji yra apriboti egzempliorių reikšmėmis atsižvelgiant į žinių bazę, net jeigu kintamieji nėra grąžinami kaip atsakymo į užklausą dalis. KAON2 Algoritmas, kuris duoda atsakymus į užklausas, kuriose nėra tiksliai aprašytų kintamųjų, yra sukurtas, bet nėra įgyvendintas iki šiol.

Užklausos gali būti suformuluotos panaudojant SPARQL. Didelė dalis (bet ne visa) SPARQL reikalavimų palaikoma. Praktiškai, palaikomos tik tos užklausos, kurios yra suderinamos su sujungtomis užklausomis. Kaip alternatyva, užklausos gali būti suformuluotos su F-Logic.

[7] SPARQL (Support In MySQL) – yra kalba RDF duomenų užklausoms, taip pat kaip SQL kalba yra kalba surištų duomenų užklausoms.

Skirtingai nuo daugelių šiuo metu prieinamų DL tikrinimo priemonių, kaip FaCT, FaCT++, RACER arba Pellet, KAON2 nerealizuotas išvadų lentelių skaičiavimas. KAON2 tikrinimo priemonė realizuota pagal naują algoritmą, kuris sumažina žinių bazę iki dataloginės programos. Šie nauji algoritmai leidžia kreiptis į dedukcinės duomenų bazės[8] metodus DL sutikrinimui. Atsižvelgiant į našumo įvertinimą, algoritmai pagreitino atsakymus į KAON2 užklausas porą kartų, negu egzistuojančios sistemos. [BM06]

11.2.2 KAON2 problemos

- KAON2 šiuo metu negali vartoti nominalų (išskaičiuojamų klasių): Jeigu ontologija turi owl:oneOf klasę arba owl:hasValue apribojimą, kiekviena tikrinimo priemonės užduotis mes klaidą.
- KAON2 šiuo metu negali vartoti didelių skaičių pagrindiniuose teiginiuose. Pastebėtos problemos ontologijoje, labai dideli apribojimai: KAON2 šioje ontologijoje nesugeba atsakyti į bet kurias užklausas. Tačiau, problemos atsiradimas priklauso ne tik nuo panaudotų skaičių, bet ir nuo kitų ontologijos aksiomų. [BM06]

11.2.3 KAON2 dokumentacija

Šiuo metu nėra surašytos dokumentacijos, kurioje būtų aprašytas KAON2 panaudojimas, tačiau yra pavyzdžių, kuriuose demonstruojamas KAON2 kaip užduočių spendimo priemonė.

11.3 Pellet tikrinimo priemonė

Pellet – loginio OWL DL išvedimo mechanizmas, su atviru programiniu kodu, realizuotas su Java 1.4 versija. Pellet gali būti panaudotas kartu su Jena[9] arba OWL API[10] bibliotekomis. Be to, gali būti įtrauktas ir į kitas technologijas.

Pellet buvo sukurtas Merilendo Universitete Mindswap laboratorijoje. Ši technologija pagrįsta sprendimų lentelių algoritmais, kurie buvo išvystyti DL technologijai. Pellet palaiko pilną OWL-DL išreiškimą, įskaitant nominalų (išskaičiuojamos klasės) tikrinimą.

Pellet technologija praplečia DL su:

- Tinkamais pagrindiniais apribojimais;

[8] Dedukcinės duomenų bazės – tai bazė sudaryta iš 2-jų dalių: ekstencionalinė, kaupia faktus, ir intencionalinė, kuri kaupia naujų faktų loginio išvedimo taisykles, panaudojant ekstencionalinę dalį ir vartotojų užklausas.

[9] Jena – tai Java aplinka semantinio voratinklio programiniams įrankiams konstruoti. Ji suteikia programinę aplinką RDF, OWL, SPARQL ir turi taisyklėmis pagrįstą išvadų sistemą.

[10] OWL API – suteikia programinį priėjimą prie duomenų struktūrų, kurios pavaizduotos OWL ontologijos pavidalu.

- Kompleksinėmis posavybių aksiomomis;
- Lokaliniais atvirkštiniais apribojimais;
- Atvirkštinėmis, simetrinėmis ir anti-simetrinėmis savybėmis;
- Disjunktinėmis savybėmis;
- Vartotojo-aprašytais duomenų tipais.

Pellet teikia daug įvairių tikrinimo priemonių kurios aprašomos apačioj. Be to, technologija įtraukia įvairias ontologijos optimizavimo technikas ir turi, atsakymų į sujungtas užklausas ir padidintą/praplėstą testavimą. [CP06]

11.3.1 Pellet ypatybės

- Standartinės tikrinimo priemonės Pellet suteikia visas standartines išvadų sudarymo galimybes, kurios tradiciškai teikia DL tikrinimo priemonės:
- Neprieštaringumo užtikrinimas užtikrina, kad ontologija neturi jokių prieštaringų duomenų. Semantinio OWL dokumentas suteikia formalų ontologijos neprieštaringumo apibrėžimą, kurį naudoja Pellet.
- Sąvokų įvertinimas tikrina, ar klasė gali turėti kokių nors savybių. Jeigu klasė yra neįvertinama, bet kuri surasta šios klasės savybė bus visos ontologijos prieštaringumo priežastis.
- Klasifikacija apskaičiuoja poklasio santykius tarp kiekvienos įvardintos klasės, užbaigtos klasių hierarchijos sukūrimui.
- Įvykdymas suranda labiau apibrėžtas klases, kuriems sukuriami egzemplioriai; kitaip tariant, apskaičiuoja tiesioginius tipus kiekvienam egzemplioriui. Realizavimas gali būti įvykdytas tik po klasifikavimo, kadangi tiesiogini tipai apibrėžiami atsižvelgiant į klasių hierarchiją. Klasifikuojant hierarchiją, galima gauti visus egzempliorių tipus. [CP06]

11.3.2 Pellet problemos

Neįvertinamų sąvokų suradimas ontologijoje yra paprastas uždavinys. Tačiau klaidų nustatymas ir taisymas apskritai nepalaikomas. Pavyzdžiui, neduodamas paaiškinimas, kodėl įvyko klaida (t. y. Nurodant aksiomas ontologijoje ir kodėl jos nesuderinamos) arba kaip priklausomybės tarp klasių didina klaidas. [CP06]

11.3.3 Pellet dokumentacija

Produktas turi gerą aprašymą, kuriame surašyta kaip galima naudoti produktą, kokiais tikslais bei kaip ieškoti klaidas. Oficialiame tinklalapyje galima rasti net atsakymus į DUK (dažnai

užduodamus klausimus). Tačiau toks dokumentas nėra vartotojo instrukcija, ir jame išdėstytos informacijos neužtenka pilnam produkto išnagrinėjimui ir įvertinimui.

11.3.4 Pellet panaudojimas

Pellet suteikia daug įvairių prisijungimo prie tikrinimo priemonės būdų:

- Programa iš komandinės eilutės;
- Programinė sąsaja, kuri gali būti naudojama atskirai arba kartu su Jena ir OWL-API bibliotekomis.
- DIG serveris, kuris suteikia galimybę naudoti Pellet su skirtingais klientais, tokiais kaip Protege-OWL redaktorius;
- Integruotas į ontologijos redaktorių SWOOP. [CP06]

11.4 RacerPro tikrinimo priemonė

RacerPro – loginio OWL išvedimo mechanizmas ir Semantinio tinklalapio serveris.

11.4.1 RacerPro ypatybės

RacerPro atsiradimas yra iš DL srities vidaus. Kadangi DL suteikia tarptautinį požiūrį į standartizuotas ontologijos kalbas kontekstuose, taip vadiname Semantiniame tinklalapyje, RacerPro taip pat gali būti panaudotas kaip sistema, tvarkanti Semantinio tinklalapio ontologijas, su OWL kalba (t. y. jis gali būti panaudotas, kaip tikrinimo priemonė ontologijos redaktoriuose, tokiuose kaip Protege). Tačiau, RacerPro taip pat gali būti pritaikytas, kaip semantinio voratinklio informacijos šaltinis su optimizuota paieškos priemone, nes jis gali tvarkyti dideles duomenų aprašymų rinkinius (pvz., aprašytais su RDF arba OWL). [RS07]

11.4.1.1 Semantinio tinklalapio tikrinimo priemonė ir informacijos saugykla

Semantinis tinklalapis siekia teikti „suprantamą mašinoms“ meta duomenis. Svarbus aspektas naudojant šiuos resursus ateities sistemose yra galimybė apdoroti OWL dokumentus - OWL KBs (knowledge bases iš anlg. žinių bazės), kur OWL yra oficiali semantinio tinklalapio kalba. Ontologijos gali būti paimtos iš bet kurios vietos arba praplėstos konkretaus domeno tikslais. Tam tikslui, tikrinimo priemonė reikalauja ontologijos redaktoriaus sistemos. RacerPro gali apdoroti OWL Lite bei OWL DL dokumentus. Tačiau taikomi tam tikri apribojimai – OWL DL dokumentai apdoroja su apytiksle nominalų reikšme klasių išraiškose, be to vartotojais aprašyti XML duomenų tipai kol kas nepalaikomi.

OWL ontologijose ir RDF duomenų aprašymuose palaikomos sekančios galimybės:

- Neprieštaringumo tikrinimas OWL ontologijoje ir duomenų aprašymų rinkinyje;
- Santykių paieška tarp besąlygiškų poklasių;
- Sinonimų paieška sąvokoms (klasėms arba savybėms);
- Kadangi praplėsta informacija iš OWL dokumentų turi būti panaudota taikomųjų programų užklausose, OWL-QL[11] užklausos apdorojimo sistema prieinama RacerPro projektui kaip atviras kodas;
- HTTP klientas išrenka importuojamas sąvokas iš tinklalapio. Daugybinės sąvokos gali būti importuotos į vieną ontologiją; [RS07]

11.4.2 RacerPro dokumentacija

Pateiktas produkto aprašymas, aprašyta vartotojo instrukcija. Tačiau dokumentacija yra prieinama tik gavus produkto licenciją, kuri yra platinama tinklalapyje.

11.4.3 RacerPro panaudojimas

RacerPro tikrinimo priemonei buvo sukurtos speciali įrankiai kuriuose galima būtų pritaikyti šį produktą:

- RacerManager – užklausoms realizuoti tikrinti su OWL-QL;
- RacerPorter – grafinis interaktyvus vartotojo klientas;
- RICE(Racer Interactive Client Enviroment) – interaktyvi klientinė aplinka; [RS07]

11.5 Ontologijos tikrinimo priemonių palyginimas ir įvertinimas

Išnagrinėjus keturias technologijas galima jas palyginti įvertinus teigiamas bei neigiamas puses. Palyginimas padarytas sekančioje lentelėje:

[11] OWL-QL – tai formali kalba ir protokolas užklausų agentui bei atsakymų agentui apdorojant užklausas-atsakymus, naudojant žinių bazę, realizuotą OWL kalba.

Lent. 1 „Ontologijos tikrinimo priemonių palyginimas“

	FaCT+	**KAON2**	**Pellet**	**RacerPro**
Dokumentacija/ instrukcija	Nėra	Nėra	Yra	Yra
Prieinamumas	Laisvai platinamas produktas	Laisvai platinamas produktas	Laisvai platinamas produktas	Produkto licencija yra mokama
Suderinamumas su Protégé	Yra	Yra	Yra	Yra
Konfigūravimo paprastumas	Ne	Ne	Ne	Ne
Realizavimas su atviru Java kodu	Ne	Ne	Taip	Taip
DIG sąsaja	Yra	Yra	Yra	Nėra
Didelių ontologijų palaikymas	Ne	Ne	Taip	Taip
Ontologijos prieštaringumo tikrinimo įvertinimas	Geras	Pakankamas	Labai geras	Geras

Išanalizavus šią lentelę, galima išrinkti OWL DL išvadas tiekiantį mechanizmą (tikrinimo priemonę), atsižvelgiant į sekančius reikalavimus: priemonė turi būti nemokama, turi turėti laisvai prieinamus Java kodus, realizuoti DIG sąsajas, kuo didesnes ontologijos tikrinimo galimybes bei aiškiai suprantamą darbo aplinką sukonfigūravus su Protege įrankiu. Tokia priemonė gali sėkmingai ir efektyviai patikrinti visus nesuderinamumus ontologijoje, užbaigiant jos teisingumo įvertinimą**.** [KP06]

Tokia tinkamiausi priemonė, atsižvelgiant į duotus reikalavimus – yra Pellet technologija.

Reikia pabrėžti, kad bendras visų išnagrinėtų technologijų trūkumas yra konfigūravimo informacijos trūkumas. Nei viena priemonė neturi pakankamai aprašytos informacijos, kurios užtektų tikrinimo priemonės į Protege diegimui bei tolimesniam praktiniam testavimui. Todėl, analizė buvo padaryta panaudojant grynai teorinę bazę.

Išvados

Magistro darbe buvo išnagrinėtos literatūroje aprašytos skirtingų sričių ontologijos. Šio nagrinėjimo tikslas buvo - išsiaiškinti bendrą ontologijos kūrimo procesą keliuose etapuose ir pritaikyti šiuos etapus kuriant savo ontologiją. Toliau, buvo aprašyta ontologijos kūrimo metodologija bei sudaryta vaistų ontologija, panaudojant metodologijoje aprašytas taisykles. Buvo išvardinti visi ontologijos kūrimo etapai bei aprašyti hierarchijos klasių, savybių ir egzempliorių nustatymai. Ontologijos analizei buvo atliktas ontologijos tikrinimo priemonių tyrimas, palyginimas ir įvertinimas. Šios analizės rezultate buvo rastas tinkamiausias tikrinimo priemonių produktas, kuris gali būti panaudotas ontologijos teisingumo užtikrinimui. OWL ontologija yra jauna sritis, kuri yra vystymo stadijoje, todėl, šiam momentui praktiškai atlikti ontologijos analizę bei išbandyti ontologiją taikomosiose programose, panaudojant šiuolaikines tikrinimo priemones ir technologijas yra gana sunku dėl informacijos trūkumo literatūroje.

Detalus tyrimų ir darbų planas

Nr.	Veiksmas	Terminas
1.	Mokslinės literatūros ir egzistuojančių ontologijų nagrinėjimas	2006 02
2.	Specialaus ontologijos kalbos OWL (Ontology Web Language) nagrinėjimas	2006 03
3.	Ontologijos srities ir masto apibrėžimas	2006 03 – 2006 04
3.1	Kokią sritį apims ontologija	2006 03 – 2006 04
3.2	Kam bus naudojama pasirinktos srities ontologija	2006 03 – 2006 04
3.3	Į kokio tipo klausimus ontologijos informacija turi duoti atsakymus	2006 03 – 2006 04
3.4	Kas naudosis ir palaikys ontologija	2006 03 – 2006 04
4.	Pritaikyti bendrą ontologijų kūrimo procesą prie vaistų ontologijos, formuluojant pagrindinius žingsnius	2006 04
5.	Išvadų pateikimas vadovui. Išvadų įvertinimas	2006 05
6.	Magistro darbo rašymas	2006 05 – 2006 06
7.	Kartotinių egzistuojančių ontologijų panaudojimo variantų nagrinėjimas	2006 09
8.	Svarbių ontologijos terminų išvardinimas	2006 10
9.	Klasių ir klasių hierarchijos apibrėžimas	2006 10
10.	Klasių savybių – slotų apibrėžimas	2006 11
11.	Slotų facetų apibrėžimas	2006 11
12.	Egzempliorių kūrimas	2006 12
13.	Galutinės ontologijos surinkimas	2006 12
14.	Išvadų pateikimas vadovui. Išvadų įvertinimas	2006 12
15.	Magistro darbo rašymas	2007 01
16.	Ontologijos testavimas	2007 02
17.	Ontologijos tobulinimas	2007 03
18.	Padaryto darbo analizė	2007 04
19.	Programų naudojančių sukurtą ontologiją pavyzdžiai	2007 05
20.	Išvadų pateikimas vadovui. Išvadų įvertinimas	2007 05
21.	Magistro darbo rašymas	2007 06

Literatūros sąrašas

[FH97] Natalya Fridman Noy and Carole D. Hafner. The State of the Art in ontology design. 1997, 23 pusl.

[NB05] Naveen Balani. The future of the Web Semantic.
URL: http://www-128.ibm.com/developerworks/library/wa-semweb/index.html
76 KB, 2005.10.18.

[DL04] Dmitrij Lande. Семантический Веб: от идеи – к технологии.
URL: http://www.dwl.kiev.ua/art/sw/index1.html
47,7 KB, 2005.06

[KK04] L. Kaftanikov, S Korovin. Перспективы использования web-онтологий в учебном процессе.
URL: http://ifets.ieee.org/russian/depository/v6_i3/html/5.html
19,4 KB, 2003.03

[SWM04] Michael K. Smith, Chris Welty, Deborah L. McGuinness. OWL Web Ontology Language Guide.
URL: http://www.w3.org/TR/2004/REC-owl-guide-20040210/
166 KB, 2004.02.10

[SD01] Stin Deker. Semantic Web: XML and RDF roles.
URL: http://www.osp.ru/text/302/180411
52,824 KB, 2001.09

[AŽ00] A. Žukov: Руководство по проектированию онтологий.
URL: http://dev.ice.ru/default/onto_doc
32,0 KB, 2000.09.17

[CG05] „Calimera guidelines“:Underlying technologies and infrastructure
URL:http://www.calimera.org/Lists/Guidelines/Underlying_technologies_and_infrastructure.htm
147,511 KB, 2005

[DK04] Daniil Kalčenko: Интеллектуальные агенты семантического Web'a
URL: http://www.compress.ru/Archive/CP/2004/10/48
39 KB, 2004.10

[UM04] The University of Manchester: Protégé OWL tutorial
URL: http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf
3.43 MB, 2004.08.27

[AR05] Alan Rector: A Practical Introduction to Ontologies & OWL
URL: http://www.co-ode.org/resources/tutorials/intro/Introduction.ppt
1.7 MB, 2005

[DT06] Dmitry Tsarkov: OWL: FaCT++
URL: http://owl.man.ac.uk/factplusplus/
5 KB, 2006.12.19

[DI04] Dmitry Tsarkov, Ian Horrocks: Reasoner demonstrator, Implementing new reasoner with datatypes support
URL: http://wonderweb.semanticweb.org/deliverables/documents/D14.pdf
82,5 KB, 2004.06.30

[BM06] Boris Motik: KAON2 - Ontology Management for the Semantic Web
URL: http://kaon2.semanticweb.org/
18,3 KB, 2006

[CP06] Clark & Parsia: Pellet: An OWL DL Reasoner
URL: http://pellet.owldl.com/
25,5 KB, 2006

[RS07] Racer Systems GmbH & Co. KG: What is RacerPro?
URL: http://www.racer-systems.com/products/racerpro
24,3 KB, 2007

[KP06] A. Kozodojev, A.Privezencijev: Аннотирование информационных ресурсов в распределенной информационной системе "Молекулярная спектроскопия"
URL: http://www.elbib.ru/index.phtml?page=elbib/rus/journal/2006/part3/KPF
32,4 KB, 2006