LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

GRAFISKĀS ONTOLOĢIJU ATTĒLOŠANAS KONSTRUKCIJAS UN RĪKI

BAKALAURA DARBS

Autors: **Nauris Gruduls**
Studenta apliecības Nr.: ng13011
Darba vadītājs: profesors, Dr.dat. Kārlis Čerāns

RĪGA 2018

ANOTĀCIJA

Darba mērķis ir aplūkot un salīdzināt iespējas ontoloģiju vizualizētājiem un redaktoriem. Tiek apskatīts ontoloģiju jēdziens datorzinātņu,- semantiskā tīmekļa kontekstā, aplūkojot dažādas tīmekļa ontoloģiju valodas OWL 2.0 konstrukcijas un iespējas.

Darba ievadā tiek sniegts vispārīgs ieskats ontoloģiju modelēšanā un tās izteiksmes nozīmībā. Iztirzājuma daļā vispirms tiek apskatītas OWL 2.0 konstrukcijas no teorētiskā un praktiskā viedokļa. Tam seko rīku OWLGrEd, WebVOWL un Eddy (Graphol) apskats un konstrukciju attēlošanas iespējas, un tie attiecīgi salīdzināti, lai izvērtētu iespējas OWLGrEd redaktora grafiskās notācijas papildināšanai.

Izstrādājot darbu tika izmantotas publiski pieejamās rīku versijas, kādas tās bija uz darba rakstīšanas brīdi.

Atslēgvārdi: OWL, ontoloģijas, semantiskais tīmeklis, modelēšana, vizualizācija

ABSTRACT

CONSTRUCTIONS AND TOOLS FOR GRAPHICAL VISUALIZATION OF ONTOLOGIES

The aim of this work is to preview and compare various possibilities of graphical ontology representation and editing tools. In computer science, ontologies – precise descriptive statements about some domain, are often described using Web ontology languages OWL 2.0 possibilities.

The introduction part of this paper talks about modelling in the context of thesis, and the importance of good tools for this purpose. Following chapters describe the respective constructions from theoretical and practical point of view, previewing tools like OWLGrEd, WebVOWL and Eddy (in Graphol visual language), and respectively gives a good overall comparison and thus possible add-ons for OWLGrEd are described to be implemented.

Often the respective tools are undergoing development or changes, in comparing the possibilities of the tools, the public (stable) versions available at the time of writing were used.

Keywords: OWL, ontologies, semantic web, modelling, visualization

SATURS

APZĪMĒJUMU SARAKSTS

Apzīmējums	Skaidrojums
Aksioma	"zināms apgalvojums" [ontoloģijā], pamattēze
BNF forma	*Backus–Naur form* – bezkonteksta gramatikas uzdošanas veids
DL	*Description Logic* – formāla loģikas valoda zināšanu aprakstīšanai
ER	*Entity–relationship* – relāciju [diagrammas]
Izteiksme	*Expression, class expression* – izteiksmes, kas kādā veidā apraksta klases, propertijas un klašu instances
Klašu apgalvojums	*Class assertion* – apgalvojums par resursa piederību klasei
RDF	*Resource Description Framework* – semantiskā tīmekļa datu modelis
TDA	*Transformation-Driven Architecture* – transformāciju virzīta arhitektūra
UI	*User interface* - Lietotāja saskarne
UML	*Unified Modelling Language* – modelēšanas valoda
UNA	*Unique Name Assumption* – koncepcija, vārdam viennozīmīgi nosakot objektu
URI, IRI	*Uniform Resource Identifier* – simbolu virkne unikālai resursu identificēšanai. *Internationalized Resource Identifier* ir URI paplašinājums, atbalstot plašāku rakstzīmju atbalstu
VOWL	*Visual Notation for OWL Ontologies* – grafiskā valoda

IEVADS

Semantiskais tīmeklis ir aktuāla datorzinātņu tēma. Semantiskās tehnoloģijas mēdz raksturot ar frāzi – "padarīt tīmekli pieejamāku datoriem" [1]. Semantiskās tehnoloģijas piedāvā abstrakcijas slāni virs eksistējošajām tehnoloģijām, kas savieno datus, saturu un procesus [4]. Tās piedāvā ērtu datu integrācijas pieeju, glabājot tos RDF[1] veidā, kas līdzīgi NoSQL nebalstās uz informācijas glabāšanu statiskā, tabulārā formā, bet gan dinamiskā, 'grafveida' formā.

Dodot datoriem pieejamāku informāciju, autors uzskata, ka varam arī labāk izmantot jau esošās tehnoloģijas – kā veicot semantisko meklēšanu (piem. Google, kas sākotnēji neuzskatīja semantisko meklēšanu par nepieciešamu, bet to ir arvien vairāk ieviesusi savos risinājumos), izmantot attiecīgos datus mākslīgā intelekta apmācīšanai (jo nereti neironu tīklu apmācīšana dažkārt reducējas uz datu kvalitātes nodrošināšanu un izvēli, kā šo informāciju pasniegt datoriem), izstrādājot jaunas, dinamiskas lietotnes u.c.

Ontoloģijas, vispirms, ir svarīga komponente semantiskajā tīmeklī, izstrādājot "gudrākus" risinājumus. Ontoloģijas ļauj aprakstīt mūsu zināšanas konkrētā, formālā veidā, par kādu domēnu (*domain of interest*). Ontoloģijas apraksta konceptus šajā domēnā, un saites, kas ir spēkā starp tiem. Dažādas ontoloģiju valodas piedāvā dažādas iespējas. Piem. RDF un RDF Schema[2] ir diezgan ierobežota, ļaujot vien izteikt apakšklašu, propertiju hierarhijas, un to domēnu un vērtību apgabalu (*range*). Nereti zināšanas nepieciešams izteikt precīzāk, piem. lai pateiktu, ka katram cilvēkam ir tieši viens dzimšanas datums, vai cilvēks nevar vienlaikus būt vīrietis un sieviete. Šīs idejas ir lielā mērā balstītas uz matemātisko loģiku.

Jaunākais standarts no World Wide Web Consortium (W3C) ir OWL 2.0 [11], kas atrisina šīs izteiksmes iespējas. Tas ļauj gan precīzāk definēt, gan aprakstīt datus. Loģiskais modelis atļauj arī noteiktas spriešanas spējas (*reasoners*), lai pārbaudītu ontoloģiju konsistenci, atrastu kļūdas un sakarības, arī gadījumos, kad klasei ir piem. vairāki vecāki.

Šīs ontoloģijas var kļūt lielas un grūti uztveramas tekstuālā formā (skat 1.pielikumu). Industrijas standarta rīki kā Protégé [21] ļauj ļoti precīzi un hierarhiski aprakstīt datu struktūru, bet to attēlošanai jāizmanto papildus funkcionalitātes spraudņi, kā piem. OWLViz[3] [2, 5]. Bakalaura darba mērķis ir izpētīt atsevišķus rīkus, kas ir izstrādāti tieši grafiskās attēlošanas, vai rediģēšanas mērķiem.

[1] https://www.w3.org/RDF/
[2] https://www.w3.org/TR/rdf-schema/
[3] https://protegewiki.stanford.edu/wiki/OWLViz

Modelēšana arī grafiskā veidā ir nozīmīga, jo cilvēki uztver vizuālo informāciju daudz vieglāk, kā tekstuālu. Tā, gan piemēram, piesaistītām biznesa personām var atvieglot ontoloģijas izpratni, gan arī nozares speciālistiem piedāvā papildus perspektīvu, kas ļauj labāk iepazīt ontoloģiju, kā arī dažkārt atrast kļūdas specifikācijā.
Tomēr ne mazāk nozīmīgi ir, lai modelējot domēnus, to vizualizācijas nepazaudētu semantisko informāciju. Grafiskajai valodai ir pilnībā jāatbalsta OWL valodas iespējas, lai tas būtu iespējams. Tā, piem. Graphol grafiskā valoda ir iedvesmojusies no klasiskajām UML un ER diagrammām, attīstot tās pēc saviem ieskatiem un praktiskās pieredzes [6, 19]. Balstīšanās jau esošajās idejās ir loģiska, jo būtiski atvieglo darbu un uztveri, ja izmanto grafiskās konstrukcijas, kas pazīstamas industrijas speciālistiem. Tāpēc pat neapskatot grafiskās notācijas dokumentāciju, var diezgan viegli saprast OWLGrEd [8] piedāvātās OWL vizualizācijas, kas balstītas uz UML klašu diagrammām [3]. Tātad grafiskās notācijas ir diezgan maz ierobežotas. Galvenais ir izteiksmes spēja un lietojamība. Piem. Graphol valodas izstrādātāji veikuši lietojamības pētījumu, kas iekļauj arī OWLGrEd, bet pavisam nelielu skaitu respondentu, kas gan mazina tā faktisko nozīmību [19]. Šī darba mērķis nav veikt šādu pētījumu, vai viennozīmīgi novērtēt šos rīkus, bet vairāk salīdzināt to atšķirīgās iespējas, apskatot kā OWL 2.0 konstrukcijas attēlojamas šajos vizualizatoros un redaktoros, un sniegt piedāvājumus no gūtajām atziņām OWLGrEd grafiskās notācijas papildināšanai.

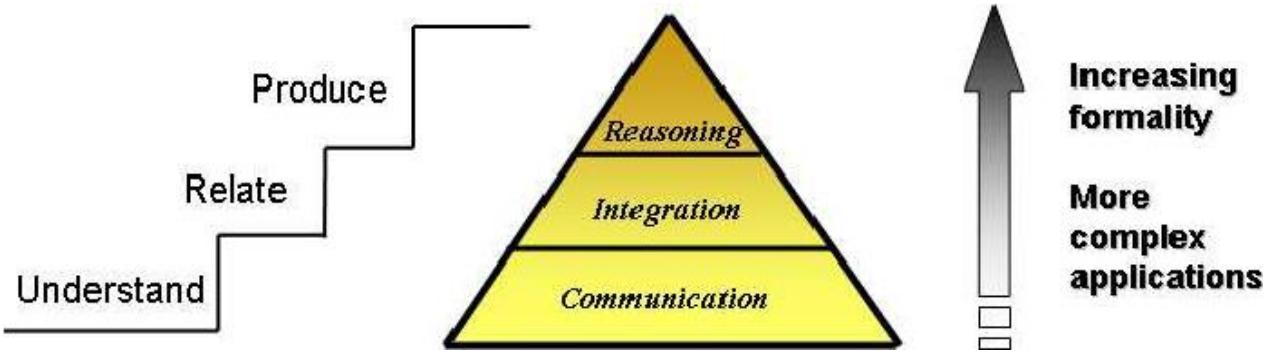
1. TĪMEKĻA ONTOLOĢIJU VALODA OWL 2.0

"OWL 2 Tīmekļa Ontoloģiju Valoda, OWL 2, ir ontoloģiju valoda Semantiskajam tīmeklim. Ontoloģija ir precīzu, aprakstošu izteikumu kopums par kādu pasaules daļu. Precīzi apraksti apmierina vairākus nosacījumus: galvenokārt, tie novērš kļūdas cilvēku komunikācijā un nodrošina, ka programmatūra strādā vienotā, paredzamā veidā un var sadarboties ar citu programmatūru. OWL 2 ontoloģijas nodrošina klases, propertijas, indivīdus, un datu vērtības, kas tiek saglabātas kā semantiskā tīmekļa dokumenti" [1, 13, 15]. Grafiskās notācijas izteiksmes brīvība ir samērā augsta, bet svarīga ir funkcionalitātes pārklāšanās ontoloģiju valodai, un grafiskajai izteiksmei. Šajā nodaļā vispirms tiks apskatītas nozīmīgākās vai biežāk izmantotās valodas konstrukcijas apkopotā veidā. Sekojošās nodaļās iekļautas arī citas atlikušās konstrukcijas.

1.1 OWL valodas īpašības

Vispirms jāapskata dažādās tīmekļa ontoloģiju valodas OWL 2.0 iespējas (mēdz apzīmēt arī kā OWL2). OWL ir daļa no W3C Semantiskā tīmekļa tehnoloģiju kopuma, kuru sastāda arī RDF un SPARQL vaicājumu valoda. RDF pieraksta datus trijnieku veidā (subjekts → predikāts → objekts), un SPARQL atļauj veikt vaicājumus pār šiem RDF datiem. Ontoloģijas ļauj specificēt jēdzienus šajos datos.
OWL 2.0 versija ieviesta kopš 2009.gada (OWL pirmā versija kopš 2001). Kā jau tika minēts, ontoloģiju valodas ļauj formāli aprakstīt domēna konceptus un saites starp tiem. Ontoloģiju valodu galvenās prasības ir:

- labi definēta sintakse,
- formāla semantika,
- pietiekami liela izteiksmes spēja,
- izteiksmes ērtums un
- efektīvas spriešanas nodrošinājums [1].



1.1. att. **Ontoloģiju funkcionalitātes izmantojums un sarežģītība [22]**

Jo bagātākas valodas iespējas, jo neefektīvākas kļūst spriešanas iespējas. Sliktākajā gadījumā valoda var būt neizrēķināma [7]. Tāpēc jāmeklē kompromiss, lai nodrošinātu pietiekami labas izteiksmes iespējas, un arī neizstrādātu pārāk grūti uztveramas konstrukcijas cilvēkiem, vai spriešanai datoriem. OWL ir virkne dažādu DL spriešanas un ar to saistīto programmatūru, kā Fact++, Pellet, RACER[4] u.c., un izteiksmes spēja ir daudz augstāka par RDF Schema, tāpēc lielākoties izstrādājot ontoloģijas vajadzīgs izmantot tikai daļu no OWL iespējām. Bet OWL nav programmēšanas valoda. OWL 2.0 ir deklaratīva valoda, kas apraksta mums interesējošo domēnu loģiskā veidā.
Turpmākajās nodaļās šīs valodas konstrukcijas tiks apskatītas detalizēti.

Atgriežoties pie ontoloģiju valodu prasībām:

- sintaktiski "labi" definēta, viennozīmīga valodas sintakse ir viena no svarīgākajām valodas iezīmēm. OWL2 tāda ir, balstoties uz un paplašinot RFD un RDFs
- formāli definēta semantika ļauj precīzi interpretēt sintaktiski izteiktos apgalvojumus.
  Tā ir pamatā arī loģiskai spriešanai par attiecīgajiem datiem. Piemēram – zinot, ka trijnieka predikāta vērtību apgabals ir Class1

  :prop1 rdfs:range :Class1 (1)
  :x :prop1 :y, varam secināt, ka y atbilst klasei Class1
- ekspresija, vai iepriekš minēta kā izteiksmes spēja, izteiksmīgums – ir viens no galvenajiem OWL valodas izmantošanas iemesliem. Piemēram, iepriekšējā piemērā netieši parādās RDF izteiksmīguma ierobežojumi, jo nevaram ierobežot vērtību apgabalu noteiktām klasēm, savā ziņā varam to tikai papildināt.

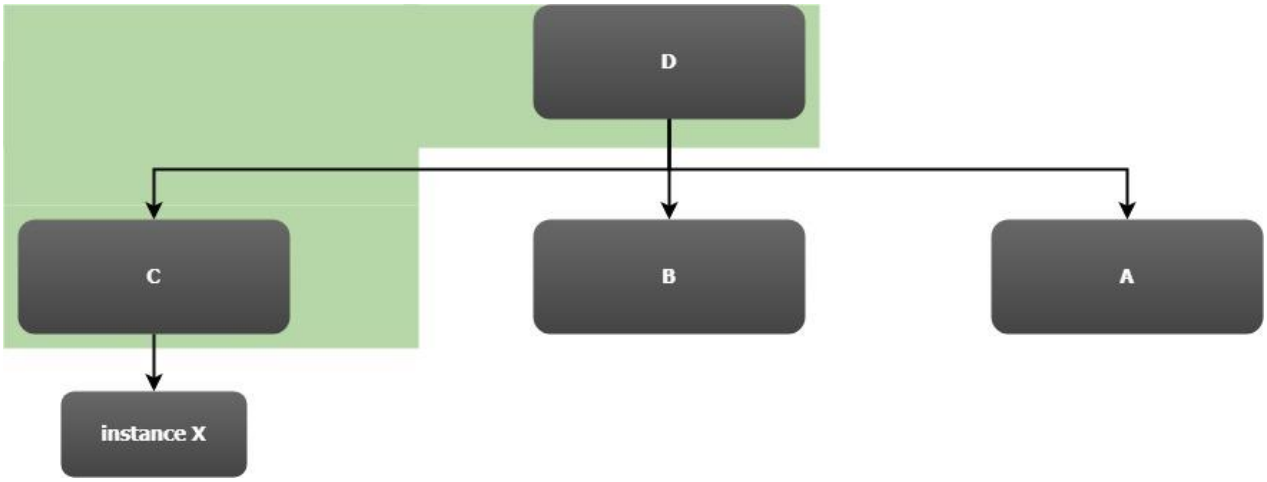
[4] http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/

- labs spriešanas atbalsts, kas OWL ir pamatā no DL.

Izteiksmīgums ir viennozīmīgi svarīgs un jāapskata sīkāk. Ja būvējam ontoloģijas, nepieciešama atbilstoša izteiksmes spēja.

Klašu piederību jāspēj izteikt tādā mērā, lai varētu secināt instanču piederību ontoloģijas klasēm, izmantojot klašu hierarhiju, domēnus un vērtību apgabalus attiecīgajā ontoloģijā. Piemēram, ja zinām nepieciešamos nosacījumus kādas klases definīcijai varam to secināt ar šīs pašas ontoloģiju valodas 'spēku'.

Klasifikācijai jāatbalsta nosacījumi klašu piederībai, lai varētu izvest attiecības starp dažādām klasēm,- "Ja x ir klases C instance, un C ir D apakšklase, tad mēs varam secināt, ka x ir arī D instance" (skat. 1.2. att.) [7].



1.2. att. **Netieša saite uz D klasi, autora ilustrācija**

Ekvivalence un vienādība – piemēram, ja visi klases :A pārstāvji ir arī klases :B pārstāvji, tad klases ir ekvivalentas. Vienādība nodrošina, ka līdzīgi kā vairāki URI var apzīmēt vienu resursu, vairākas instances var patiesībā apzīmēt tikai vienu (reālās pasaules) instanci. To OWL ir iespējams izteikt gan tieši, gan izsecināt ar loģikas rīkiem.

Klašu nepārklāšanās, diferences. Dažreiz mums ir zināms, ka divām klasēm nevar būt kopīgas instances – tās nepārklājas (*disjoint* – no angļu val.). OWL to ir iespējams norādīt. Piemēram :

Klašu Būla kombinācijas. Dažreiz ir nepieciešams kombinēt klases dažādākos veidos, kad nepietiek ar apakšklašu norādīšanu vai piem. šķelšanos/nešķelšanos. Piemēram, "persona ir klašu vīrietis un sieviete nešķeļošs apvienojums"[7].

Propertiju konteksta lokalitāte. Kā tika minēts jau iepriekš, piem. RDFs varam norādīt, ka visas propertijas :eats vērtību apgabala instances pieder kādai konkrētai klasei. Nevaram likt ierobežojumus vērtību apgabalam atkarībā no konteksta. Ar OWL varam pateikt, ka daļa

cilvēku spēlē basketbolu – profesionālu spēlētāju klase, bet daļa spēlē piem. futbolu, nemaz tieši nedefinējot tādas klases kā 'NBA spēlētājs' un 'futbola spēlētājs'.

Īpašas propertiju iezīmes. Dažkārt nepieciešams pateikt, ka kāda propertija ir transitīva (piem. "vecāks"), vai pretēja ("māca" un "mācās"), unikāla īpašība ("māte") u.c.

Kardinalitātes ierobežojumi. Dažreiz nepieciešams pateikt, tieši cik dažādas vērtības propertijai var būt (personai 2 vecāki, kursam vismaz 1 pasniedzējs).

Konsistences pārbaude – iespējams pārbaudīt ontoloģiju un atrast kļūdas,

- pateikt vai mūsu deklarētās instances atbilst definētajai ontoloģijai,
- vai nav ieviesušās kļūdas definējot saites starp klasēm.

Ne mazāk svarīgs ir arī ontoloģiju valodas lietošanas ērtums, lai nebūtu jāveic daudz lieka darba.

1.1.1 OWL apakšvalodas OWL2 Full un OWL2 DL

Lai atbalstītu atšķirīgās prasības un nodrošinātu efektivitāti, OWL2 tiek nošķirta divās apakšvalodās, katrai atbilstoši specifiskām prasībām – "OWL2 Full" un "OWL2 DL". OWL Full izmanto visus OWL 2.0 valodas primitīvus, un nodrošina pilnīgu sasaisti ar RDF un RDFs. Ļoti augsta brīvība un izteiksmīgums, varot pat, piemēram, definēt klašu skaita limitu visu klašu klasei. OWL2 Full dokumenti ir pilnībā saderīgi ar RDF. Bet OWL Full efektīvu spriešanas atbalstu praktiski nav iespējams nodrošināt.

Lai to panāktu, tiek izmantota OWL2 DL, kas tiek kartēta uz DL. Daudzas DL valodas ir spēcīgākas par Izteikumu loģiku, bet vājākas par Pirmās kārtas loģiku[5] (DL indivīds ir analogs konstantei pirmās kārtās loģikā). OWL DL nosaka, kā un kādi valodu OWL2, RDF un RDFs primitīvi var tikt izmantoti

- primitīvus nevar izmantot pašus uz sevi,
- visas klases ir owl:Class instances, nevis rdfs:Class klases instances,
- OWL DL stingri nošķir propertijas, kuru vērtību apgabals satur literāļus un ne-literāļus. Visas propertijas ir vai nu owl:ObjectProperty vai owl:DatatypeProperty,
- Resurss nevar vienlaikus būt pārstāvis dažādiem primitīviem (lai gan to nosaukumi var sakrist).

[5] https://en.wikipedia.org/wiki/Description_logic

Tātad tiek saglabāta cieša saite ar izteikumu loģiku, un RDF/RDFs noteiktā veidā apvienots ar OWL2 – tiek definēta hierarhija starp šo valodu primitīviem. Tas, protams, ierobežo valodu, bet ir iespējama efektīva spriešana [1]. Katrs OWL DL dokuments ir arī derīgs RDF dokuments, bet ne otrādi.
Līdzīgi apakšvalodu iedalījumam tiek nošķirti arī OWL profili dažādām spriešanas efektivitātes un izteiksmīguma prasībām.

1.1.2 OWL 2 valodas profili

Dažus no profiliem, veido, piemēram, OWL2 DL specifikācijas atsevišķas apakškopas, vai nepilnas OWL2 Full iespējas. Nereti nav nepieciešamas visas DL piedāvātās konstrukcijas, tāpēc efektīvāk būtu izmantot tam atbilstošu profilu.

OWL 2 EL – ir apakškopa no OWL 2, kas

- Piemērota ontoloģijām ar ļoti lielu klašu un propertiju skaitu,
- Var izmantot visas šādu ontoloģiju konstrukcijas,
- Spēj pārbaudīt ontoloģijas konsistenci polinomiālā laikā.

OWL2 QL – spriešana OWL2 DL un OWL2 EL ir optimizēta klašu aksiomu līmenī, bet sliktāk apstrādā ontoloģijas, kurās ir daudz apgalvojumu par indivīdiem. QL profils ir

- Daudz efektīvāks atbildot uz vaicājumiem (logaritmisks laiks),
- Pietiekams UML un līdzīgu modeļu attēlošanai
- Ierobežo konstrukciju izmantošanu – nevar izmantot kardinalitātes nosacījumus, tranzitīvas propertijas u.c

OWL2 RL – piedāvā mērogojamu spriešanas atbalstu, un balstās uz efektīvu loģikas un likumu izmantošanu [11].
Precīzāks profilu pārskats ir ārpus šī darba tvēruma.

1.2 OWL 2.0 valodas sintakses un ontoloģiju dokumenti

OWL 2.0 (kā OWL 1.0) ir būvēta uz RDF un RDFS un attiecīgi var tikt izteikta arī ar RDF derīgiem sintakses veidiem. Bet OWL ir izveidotas specifiskas sintakses atšķirīgiem mērķiem, kas tikušas attīstītas līdz ar jaunāko OWL standartu.

Funkcionālā stila sintakse – ir viena no kompaktākajām un vieglāk lasāmajām sintaksēm, tiek izmantota dažādos rīkos, OWL semantikas dokumentos (t.sk. OWL 2.0 valodas specifikācijā), kartējot uz un no RDF sintaksēm, un sastopama dažādos OWL2 profilos. Tajā funkcija rakstāma pirmā, kam seko tās operandi. Funkcionālās sintakse ontoloģija vispārīgā formā pierakstāma diezgan īsi (skat. 1.3.att.), un var tikt kanoniski parsēta, lai no tās konstruētu šo OWL ontoloģiju.

```
ontologyDocument := { prefixDeclaration } Ontology
prefixDeclaration := 'Prefix' '(' prefixName '=' fullIRI ')'
Ontology :=
    'Ontology' '(' [ ontologyIRI [ versionIRI ] ]
       directlyImportsDocuments
       ontologyAnnotations
       axioms
    ')'
ontologyIRI := IRI
versionIRI := IRI
directlyImportsDocuments := { 'Import' '(' IRI ')' }
axioms := { Axiom }
```

1.3.att. **Ontoloģijas dokumenta vispārīgs pieraksts [11]**

Parsējot šo dokumentu tiek translēti visi prefiksi, savienojot zināmo informāciju ar dokumentu un tajā importēto ontoloģiju dokumentu definīcijas ar dotajām aksiomām.
Funkcionālās sintakses mērķis ir vieglāk redzēt ontoloģiju formālo struktūru [1, 11].

RDF/XML sintakse – ir pamata sintakse, kura var tikt lasīta un rakstītas jebkurā OWL 2.0 atbilstīgā programmatūrā – tā ir primārā sadarbībai starp dažādiem rīkiem. Ļoti svarīgi, ka jebkuru OWL ontoloģiju var kartēt uz RDF/XML, kas ir relatīvi sarežģīts process, bet ir vitāli nepieciešams ontoloģiju savienojamībai semantiskajā tīmeklī.

OWL/XML sintakse - ir atbilstoša XML serializācija dažādu esošo XML rīku izmantošanai, analoģiski RDF/XML sintaksei. Tā savā ziņā līdzīga arī Funkcionālajai sintaksei. Tās saknes elementam vienmēr jābūt *Ontology* elementam.

Mančesteras sintakse – Mančesteras universitātē izstrādāta, lai būtu pēc iespējas labāk uztverama cilvēkiem, un īsāk pierakstāma. Tā tiek izmantota arī teksta ontoloģiju redaktoru UI. Mančesteras sintakse OWL 2 ontoloģijām definēta vienkāršā BNF formā, kas ļauj veidot sarakstus ar neterminālļiem – klašu izteiksmēm.

Turtle sintakse – ir paredzēta ērtākam RDF trijnieku pierakstam un lasīšanai - nav jāatkārto informācija, kā tas ir XML sintaksēs, bet to var pierakstīt īsāk izmantojot nedrukājamās rakstzīmes. Rakstot pašrocīgi vai lasot tas ir viennozīmīgi ātrākais un ērtākais sintakses veids.

Tipiski ontoloģijas dokumentam jāsatur vismaz rdf, rdfs, owl un xsd nosaukumvietu prefiksus [1], kā arī apgalvojumus kā bāzes nosaukumvieta, iezīme, komentārs, un norādes uz citām ontoloģijām, kuras var tikt importētas (owl:imports) – tranzitīvi importējot visas attiecīgās aksiomas un saistītos datus (skat. 1.pielikumu).

1.3 OWL valodas primitīvi augstā līmenī

Lai gan OWL cenšas attēlot visas zināšanas par attiecīgo domēnu, tā protams nevar attēlot visas cilvēku zināšanas [1, 11]. Tāpēc OWL "var uzskatīt par spēcīgu, universālu (plašlietojuma) modelēšanas valodu atsevišķiem zināšanu apgabaliem" [13]. Attiecīgi OWL2 esošajai ciešajai saiknei ar formālo loģiku tiek izmantoti arī nedaudz atšķirīgi termini, kā piemēram:

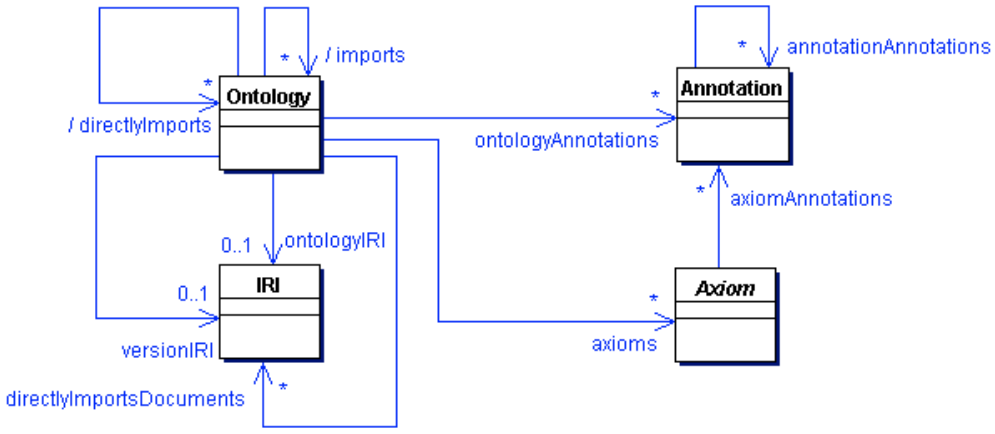
- Mainīgie/individuālie mainīgie, indivīdi, entītijas – *individuals* – apzīmē klašu instances,
- apgalvojums/klašu apgalvojums - *assertion* - apgalvojums par piederību konkrētai klasei, piemēram, apgalvojot, ka mainīgais pieder klasei *Person;* apgalvojumi var tikt iekļauti ontoloģijā, lai papildinātu to ar zināšanām par konkrētiem objektiem,
- izteiksmes – *expressions* - izsaka informāciju par klasēm, propertijām un instancēm; izteiksmes veido kompleksus mainīgo aprakstus, piemēram –

```
_:x rdf:type owl:Class ;
    Owl:unionOf (:Vīr :Siev) .                    (2)
```

nosaka anonīmo klasi, kas ir klašu Vīr un Siev apvienojums, un ir identisks ar

```
Owl:unionOf (:Siev :Vīr) .                        (3)
```

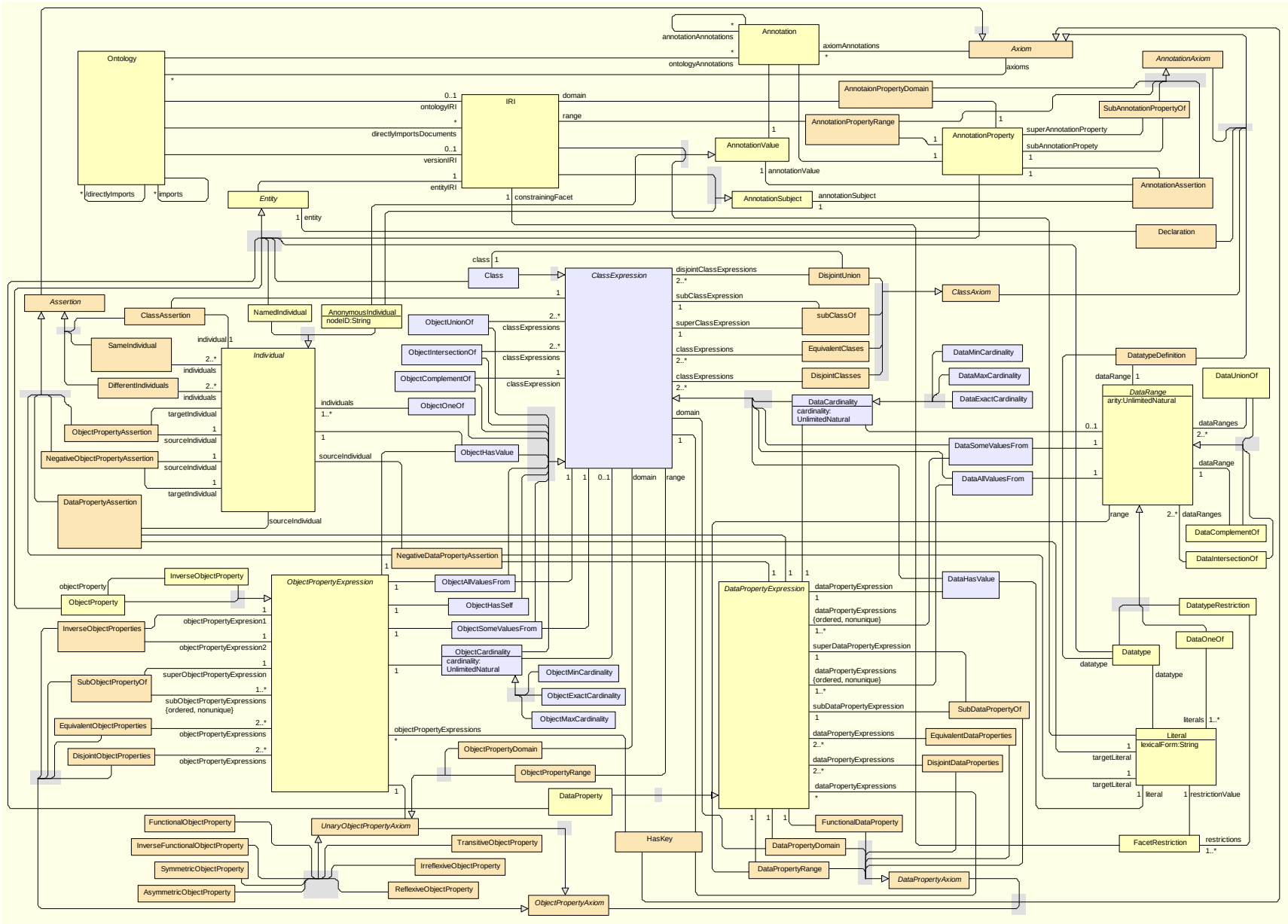
- aksiomas – *axioms, restrictions* – var tikt definētas pēc tam. Aksiomas ir izteikumi dotajā ontoloģijā. Aksiomas ir pati svarīgākā OWL ontoloģijas komponente, jo nosaka, kas ir patiess (*true*) dotajā ontoloģijā [15]. "*Every man is mortal*" ir viens no reālās pasaules aksiomu piemēriem. Aksiomas ir izteikumi, kas saistīti ar esošajām klasēm. Aksiomas ontoloģijās, līdzīgi kā matemātiskajā loģikā, attēlo mūsu "pamata/fona zināšanas" [14]. Piemēram, definējot klasi, kas ir ekvivalenta ar iepriekš izveidoto. Aksiomas ir ļoti svarīgas, jo OWL2 valoda lielā mērā nosaka lietu (*things, owl:Thing*) kopas. Par ontoloģijām var domāt kā par šādu zināšanu izteikumu (aksiomu) kopumu [1]. To labi attēlo OWL UML diagramma (skat. 1.4.att.). Dažkārt saka, ka aksiomas nosaka un ierobežo, kādi indivīdi var piederēt kādai klasei. Bet kā redzēsim, klašu aksioma arī ir aksioma, tāpēc uzskatu, ka aksiomu jēdziens ir jādefinē nedaudz plašāk.



1.4.att. **Ontoloģijas struktūras UML diagramma [7]**

No šīs diagrammas var secināt jau dažas lietas, kā piem., ka nevar būt divas aksiomas, kas ir semantiski ekvivalentas, jo veido aksiomu kopu, vai ka ontoloģija var saturēt arī vairākas citas ontoloģijas.

Savukārt ontoloģiju konstrukcijas ērti kopumā aplūkot metamodeļa veidā (skat 1.5. att., modelis nav autora izstrādāts); visas metamodelī redzamās konstrukcijas tiks apskatītas turpmākās nodaļās.



1.5. att. ***OWL ontoloģiju metamodelis***[6]

[6] K.Čerāns, privāta komunikācija

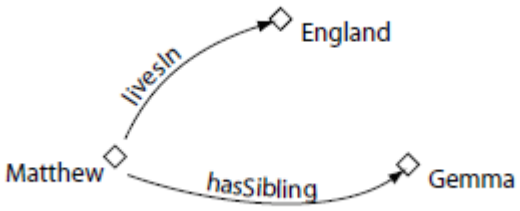
1.3.1 OWL valodas nozīmīgākie propertiju tipi

Kā tika minēts iepriekš, OWL2 tiek izdalīti divi galvenie propertiju tipi – objektu propertijas (*object properties*) un datu propertijas (*datatype properties*). Ir vairāki propertiju veidi, kas tiek atsevišķi izdalīti OWL, bet kā redzams metamodelī (1.5.att.) tieši *ObjectPropertyExpression* un *DataPropertyExpression* sastāda nozīmīgu daļu no modeļa klasēm.

Tātad propertija ir binārs savienojums, kas saista indivīdus. Propertija ir OWL analogs saišu relācijām UML, lomām aprakstošajā loģikā.

"Propertija var tikt ierobežota ar tieši vienu vērtību – tādā gadījumā tā būs funkcionāla" [9, 10].

Indivīdam var būt viena, vairākas propertijas, vai neviena (skat. 1.6.att.)



1.6. att. **propertiju grafiskā reprezentācija [9]**

Valodā OWL sekojoši divi indivīdi ar dažādiem vārdiem varētu apzīmēt vienu reālās pasaules instanci, un varētu arī to neapzīmēt (netiek izmantots UNA). Līdzīgi kā mums viennozīmīgi jāpasaka, vai propertija ir datu tipa, vai objekta, mums arī specifiski jāpasaka, vai attiecīgie indivīdi patiesībā apzīmē vienu indivīdu.

Objektu propertijas ir viens no biežāk izmantotajiem propertiju tipiem, jo tas savā starpā saista mainīgos. Tas nozīmē ka RDF trijniekā gan subjekts, gan objekts ir resursi (neliterāļi). Piemēram,

```
bio:married     rdf:type        owl:ObjectProperty ;
                rdfs:domain     :Person ;
                rdfs:range      :Person ;                    (4)
```

Apakšklases Objektu propertijām ir owl:topObjectProperty – propertija, kas saista visus mainīgos ontoloģijā, un owl:bottomObjectProperty – kas nesaista nevienu.

Datu propertijas saista indivīdus ar noteikta datu tipa literāļu vērtībām, kā piem. :Pers_kods. OWL2 atļauj izmantot Datu propertijas vērtību apgabala definēšanai XML Schema līdzekļus. Sekojoši var definēt arī savus datu tipus XML un izmantot OWL ontoloģijā. Datu propertijas var būt arī funkcionālās propertijas (tiks apskatīts tālāk).

Anotācijas propertijas ir propertijas, kas netiek izmantotas spriešanai izmantojot OWL DL, bet var tikt izmantotas RDF Schema un OWL Full spriešanas rīkos. Tās visbiežāk ir kā, piemēram, komentāri, iezīmes, sniedzot paskaidrojošu informāciju vai metadatus (datus par datiem) par klasēm, indivīdiem vai objektu un datu propertijām – tām nav nekādu 'loģisku' seku ontoloģijā. Savukārt, piemēram, iezīmi (rdf:label) vai komentāru (rdfs:comment) var pierakstīt vairākās valodās, grafiskā attēlojumā to neparādot pilnībā, bet metadatos iekļaut arī piemēram komentāra vai definīcijas avotu.

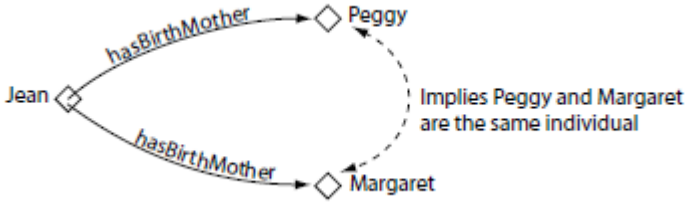
Objektu apakšpropertijas – analoģiski apakšklašu aksiomām (skat. Nodaļu 1.3.3) – ļauj definēt apakšpropertijas, nosakot vēl precīzākas to īpašības. Ja indivīdi ir saistīti ar apakšpropertiju, tie būs saistīti arī ar bāzes propertiju, bet ne otrādi.

Tranzitīvas propertijas – līdzīgi apakšklasēm, atbalsta šādu loģisko ķēžu veidošanu. relācijas, kuras pēc būtības arī ir tranzitīvas. Relācija R ir tranzitīva, ja no R(a,b) un R(b,c) seko R(a,c). Vienkāršākie piemēri ir "mazāks" un "lielāks", nedaudz sarežģītāks piemērs ir piem. īpašība 'priekštecis' ģimenes kokos. Tranzitīvas propertijas pieder pie kompleksajām (saliktajām) propertijām. Pie kompleksajām propertijām pieder jebkura propertija, kas pati ir tranzitīva, vai tās inversā propertija ir tāda, iepriekšminētās topObjectProperty un bottomObjectProperty, jebkura propertija, kurai ir tranzitīva apakšpropertija, vai apakšpropertija, kuras inversā propertija ir tranzitīva, augstākā līmeņa propertija propertiju ķēdē vai tās inversā propertija, un protams jebkura propertija, kas ir ekvivalenta ar kādu no šādām propertijām [1]. Kompleksās propertijas nevar parādīties piemēram, aksiomās, kas nosaka kardinalitātes ierobežojumu klasēm.
Piemērs no OWL W3C labās prakses –

```
Class(Minimal_Italian_Dinner,
   subClassOf( Restriction( has_course, cardinality(3) )))          (5)
```

nosaka, ka propertijai var būt tikai trīs dažādas vērtības [12].
Kā redzēsim tālāk, OWL valodai ir daudz saistītu attiecīgo iespēju, gan tieši nosakot kardinalitāti ar tam paredzētajām valodas konstrukcijām, gan tādām konstrukcijām, kas to ietekmē netieši, kā piem. owl:someValuesFrom.
Tranzitīvās (tātad arī kompleksās) propertijas var tikt izmantotas vairākos posmos, piemēram, veicot spriešanu, un no garām propertiju virknēm veicot noderīgus secinājumus, izmantojot tikai tranzitivitātes īpašību.
Jāpieblist, ka tranzitīvā propertija ir kā klase, tātad savai ontoloģijas propertijai varam piešķirt rdf:type owl:TransitiveProperty.

Funkcionālās un inversi-funkcionālās propertijas. Par funkcionālu propertiju var domāt kā par funkcijas propertiju – kā funkcija (īpašība) f(x), kurai katram x var būt tikai viena f(x) vērtība. Loģikā izmantots piemērs ar funkcijas īpašību māte, jo cilvēkam var būt tikai viena bioloģiskā māte. Šīs propertijas izmanto šo pašu ideju, palīdzot noteikt, kad divi resursi apzīmē vienu un to pašu resursu (piem. ar atšķirīgiem URI). Uzskatāms piemērs no OWL Protégé pārskata par šo pašu ideju –



1.7. att. **Funkcionālās propertijas piemērs [9]**

Būs interesanti tālāk apskatīt, vai un kā dažādie grafiskie rīki parāda, šīs un attiecīgi arī owl:InverseFunctionalProperty attiecības.
Inversā funkcionālā propertija ir viena no svarīgākajām propertiju konstrukcijām, jo ļauj apvienot semantiskā tīmekļa datus no dažādiem datu avotiem. Tātad, ja vērtību apgabala vērtības ir ekvivalentas, varam secināt, ka atbilstošās domēna vērtības arī ir ekvivalentas. Piemēram no matemātikas, īpašība ‘kvadrātsakne’ ir inversi funkcionāla un īpašība ‘kvadrāts’ funkcionāla.

Simetriskās un asimetriskās propertijas – simetriskas propertijas ir tādas, kurām izpildās sekojošais –

$$a \ :\text{simetriskaIpasiba} \ \ b \rightarrow b \ \ :\text{simetriskaIpasiba} \ \ c \qquad (6)$$

Simetriska propertija ir idejiski ļoti līdzīga owl:inverseOf propertijai, kura saista divas propertijas, bet simetriskā/asimetriskā īpašība apraksta tikai vienas propertijas piederību šādai klasei, jo simetriskās propertijas inversā propertija ir dotā simetriskā propertija. Simetriska propertija ir piem. 'kaimiņš' grafā.
Asimetriskā propertija var noderīga tieši mērķim, lai norādītu, ka attiecīgā propertija nav simetriska (kas savukārt var palīdzēt, piemēram ontoloģijas konsistences pārbaudē). Piemēram,

```
:kaimiņš   rdf:type   owl:ObjectProperty ;
           rdf:type   owl:SymmetricProperty ;
:vecāks    rdf:type   owl:ObjectProperty ;
           rdf:type   owl:AsymmetricProperty ;          (7)
```

Refleksīvas un neatgriezeniskas propertijas. Refleksivitāte ir viens no jaunieviesumiem OWL 2 versijā. Refleksivitāte propertijai nozīmē, ka katrs indivīds ir saistīts ar sevi ar šo propertiju. Tādu piemēru ir maz, piem. "jebkura lieta ir daļa (:isPartOf) no sevis" [2]. Viennozīmīgi lielākā daļa propertiju būs neatgriezeniska, jo domēns un vērtību apgabals nepārklāsies – propertija kas saista indivīdus a → b, bet tā pati propertija nesaista indivīdus pretējā virzienā.

1.3.2 Populārākās aksiomas par propertijām

Aksiomas ir galvenā ontoloģiju sastāvdaļa, un ar tām iespējams precīzāk 'definēt propertiju iezīmes, un kā tās saistās ar klasēm un citām propertijām' [1].

Domēns un vērtību apgabals tika pieminēts jau iepriekš. Ja propertijai tiek noteikti vairāki vērtību apgabali, rezultējošais vērtību apgabals būs šo vērtību apgabalu klašu šķēlums. Tam esot nodefinētam, no atbilstošajiem trijniekiem varam secināt to klašu piederību. Piemēram –

```
:eats      rdfs:domain    :Carnivoire
:eats      rdfs:range     :Meat                        (8)
```

Interesanti, ka spriešanā domēns un vērtību apgabals tiek izmantoti kā aksiomas – piešķirot saldējuma klasei propertiju, kuras domēns ir picas, tā nerezultētos kļūdā, bet gan spriedumā, ka saldējums ir apakšklase picai [10].

OWL atļauj definēt arī **inversās propertijas**, kas atbilst matemātiskajai definīcijai – ja f(x) = y, tad $f^{-1}(y) = x$ – tātad propertiju pāris, kuri saista indivīdus 'pretējos virzienos'. Populārs piemērs ir īpašības 'vecāks' un 'bērns'. Pati par sevi šī aksioma šķiet ne īpaši vērtīga, bet var noderēt modelēšanā.

Ekvivalentas propertijas – owl:equivalentProperty – ja divi indivīdi ir saistīti ar kādu propertiju, tad tie būs saistīti arī ar jebkuru šai propertijai ekvivalentu propertiju. Tās var pielietot, lai sasaistītu datus no dažādiem avotiem.

Nepārklājošās (atdalītās) propertijas – ja zināms, ka divas propertijas ir atdalītas, tad to subjektu un objektu pāri šīm propertijām nepārklāsies. OWL 2 DL owl:objectProperty un owl:DatatypeProperty arī ir atdalītas, kas ļauj to izmantot dažādos rīkos, t.sk. arī attēlot dažādi grafiskajos rīkos.

Propertiju ķēdes. Vēl viens no jaunieviesumiem OWL 2.0 versijā salīdzinot ar 1.0 versiju ir kompleksā propertiju ķēžu definēšana. Ar propertiju ķēdes aksiomu varam definēt virkni propertiju, un šo virkni izmantot kā super-propertiju citas propertijas definīcijai, ko sekojoši var izmantot kā saīsni. Propertiju ķēdes piemērs varētu būt –

$$:\text{vecāks} \rightarrow :\text{vecāks} \equiv :\text{vecvecāks} \tag{9}$$

Propertiju ķēdes neatbalsta rekursiju, tāpēc propertiju ķēdes aksiomā nevar tikt iekļauta tā propertija, ar kuru saistām šo aksiomu.

1.3.3 Aksiomas par klasēm

OWL klases tiek definētas, sniedzot apgalvojumus par to piederību tipam owl:Class. Owl:Class ir apakšklase rdfs:Class (un katrs indivīds OWL ‘pasaulē’ pieder owl:Thing). Tas tiek darīts ar mērķi atbalstīt OWL DL ierobežojumus, atbilstoši kuriem ne visas RDFs klases ir saderīgas ar OWL DL. Spriešanā svarīgas ir owl:Thing un owl:Nothing, kur tātad owl:Thing pieder visas klases owl:Class instances, savukārt owl:Nothing ir tukša kopa. Tomēr klašu, kas nav konsistentas, biedri piederēs šai klasei. Tātad praktiski owl:Thing ietver visas atbilstošās klases un indivīdus ontoloģijā. Klašu aksiomas veido relācijas starp klasēm [1].

Apakšklašu relācija. Apakšklašu relācija tiek aizņemta no RDFs, un izmantojama kopā ar OWL līdzekļiem:

```
:Carnivoire    rdf:type          owl:Class ;
               rdfs:subClassOf   :Animal .        (10)
```

Klašu ekvivalence. Klašu ekvivalence nozīmē, ka katram klases elementam jābūt arī tai ekvivalentas klases elementam, jeb abu klašu elementu kopām pilnībā jāsakrīt. Arī šo klasi var izmantot ontoloģiju semantiskai savienošanai – kā

```
:Animal    owl:equivalentClass    dbo:Animal        (11)
```

Pašizveidotajai klasei piesaistīta DBpedia klasei, kura varētu tikt piesaistīta, importējot attiecīgo DBpedia ontoloģiju. Šis apgalvojums semantiskā ziņā dod identisku rezultātu, definējot apakšklašu saites abos virzienos starp :Animal un dbo:Animal.
Klašu ekvivalences kontekstā tiek aplūkots arī OWL metamodelēšanas īpašība – *punning* (no angļu val.). *Punning* atļauj izmantot vienu un to pašu termu gan klasēm, gan klašu instancēm atbilstošā kontekstā. Funkcionāla pieraksta piemērs [16]:

```
Declaration( Class( :Person ) )                        (12)
ClassAssertion( :Service :s1 )                         (13)
ObjectPropertyAssertion( :hasInput :s1 :Person )       (14)
```

Piemērs (13) apzīmē, ka :s1 ir :Service klases instance (indivīds), un (12) definē klasi :Person, savukārt (14) saista divus indivīdus - :Person un :s1.

Uzskaitījums – *enumeration* – ir veids kā valodā OWL iepriekšdefinēt visus klases biedrus, izmantojot pierakstu[7]

$$:\text{myNewClass} \quad \text{owl:oneOf} \quad ([x_1; x_2; x_3; \dots ; x_n]) \qquad (15)$$

Tātad tas ir viens no veidiem, kā OWL2 definēt klases. Tā ir algoritmiski ļoti neefektīva pieeja, bet nereti var tikt izmantota atsevišķu robežgadījumu modelēšanā.

Nepārklājošās (atdalītās) klases – ir viens no aksiomu veidiem kas arī nebija ierobežojams ar RDF. Tātad nepārklājošo klašu indivīdu kopām nav neviena kopīga elementa. Arī šī konstrukcija izmantojama klašu modelēšanā.

Klašu papildinājums – *complement* – Klases C papildinājums A ir visas pārējās lietas (owl:Thing), kas nepieder A. Tātad A un C apvienojums arī ir owl:Thing. Sekojoši modelēšanā tam pielietojumu redzu samērā reti, jo parasti nav nepieciešams aptvert visu domēnu vienlaicīgi.

Klašu apvienojums – *union* – ir kā papildinājums klašu definēšanai ar klašu ekvivalenci. Tas ļauj definēt vienādību ar vairāku klašu apvienojumu. Turtle sintaksē vispārīgā veidā to var pierakstīt kā

$$:\text{myNewClass} \quad \text{owl:unionOf} \quad ([c_1; c_2; c_3; \dots ; c_n]), \text{kur c – klase} \qquad (16)$$

Tam pretējais ir owl:disjointUnionOf, gadījumā, ja klases nešķeļas.

Klašu šķēlums – *intersection* – ir vēl viens no OWL valodas jēdzieniem, kas aizņemts no matemātikas pamatjēdzieniem. Tātad klasi varam aprakstīt arī šādā veidā, izmantojot citas, iepriekš definētas klases.

[7] *Turtle* sintaksē katram no saraksta elementiem būtu jābūt jaunā rindā, piemērs pierakstīts ar adaptētu sintaksi uzskatāmības nolūkos

1.3.4 Klašu aksiomas par propertijām, jeb ierobežojumi

OWL 2.0 versijā ieviestas jaunas konstrukcijas, lai definētu klases ar aksiomām, kas satur arī klašu propertijas. Šīs aksiomas 'skatās' uz minētajām klasēm un ierobežo to piederību atbilstoši propertijām. Klašu ierobežojuma aksiomas tiek definētas izmantojot ierobežojumus – *restrictions* – kas ir owl:Class apakšklase owl:Restriction. Tātad šī klase tiek definēta ar iepriekš definētajām klasēm un propertijām.
Tas ir ļoti spēcīgs OWL valodas un modelēšanas līdzeklis. Tas atrisina RDF spēka ierobežojumu, kas atsevišķos gadījumos neļāva pietiekami specifiski definēt domēnu vai vērtību apgabalu. Piemēram, varam pateikt, ka klasei 'NBAplayer' pieder tādi spēlētāji no klases 'Player', kam ir propertija, kas apzīmē dalību NBA.

1) Owl:allValuesFrom ir universāls ierobežojums klasei C un propertijai P formā "tie indivīdi, kuriem visas propertijas P vērtības nāk no klases C". Rezultātā iegūstam klasi, kas apmierina šo nosacījumu. Tātad šo konstrukciju var izmantot, lai iegūtu jaunu klasi, kurai varam ierobežot konkrētas propertijas vērtības. Propertijas vērtība var būt arī tukša.
   OWL tas sākas ar Restriction klases izmantošanu, piem. –

```
:AdvancedTest      a              owl:Class ;
                   rdfs:subClassOf      [a          owl:Restriction ;
                                        owl:onProperty    :hasQuestion
                                        owl:allValuesFrom   :HardQuestion ] . (17)
```

   Piemērā papildus izmantotā apakšklases konstrukcija palīdz izveidot jaunu klasi :AdvancedTest, kurā visi jautājumi ir grūti - atbilst nepieciešamajiem nosacījumiem (visi klases AdvancedTest elementi ir arī atbilstošās Restriction klases elementi).

2) Owl:someValuesFrom, saukts arī par eksistences ierobežojumu, ir ļoti līdzīgs. Ja (17) piemērā aizstājam owl:allValuesFrom ar owl:someValuesFrom, iegūstam jaunu klasi, kurā vismaz 1 jautājums ir grūts jautājums:

```
:NormalTest        a              owl:Class ;
                   rdfs:subClassOf      [a          owl:Restriction ;
```

owl:onProperty :hasQuestion
owl:someValuesFrom :HardQuestion] . (18)

SubClassOf izmantošana nozīmē, ka ar ierobežojumu nosakām nepieciešamos, bet ne pietiekamos nosacījumus ierobežojuma klases atbilstībai, jo rezultējošā klase būs tikai apakšklase ierobežojuma rezultātam.

3) Vērtību apgabalu jauni veidojamās klases propertijām varam ierobežot arī ar owl:hasValue. Rezultējošajā klasē propertijai P jānosaka konkrēta vērtība, nevis vērtību kopa no klases, kā tas bija piemēros 1) un 2).

4) Kardinalitātes ierobežojumi. Ar kardinalitātes ierobežojumu var noteikt, tieši cik dažādu vērtību propertija var ieņemt. Kardinalitātes ierobežojums funkcionālā stila sintaksē tika parādīts (5) piemērā. Kardinalitāte ir kvalificēta (owl:qualifiedCardinality), ja papildus skaitam tiek norādīta arī klase/klases, kuras vērtības šai propertijai var būt. qualifiedCardinality nosaka precīzu vērtību skaitu; papildus pieejamas arī konstrukcijas minimālajam (minCardinality, qualifiedMinCardinality) un maksimālajam (maxCardinality, qualifiedMaxCardinality) skaitam.

5) Datu vērtību apgabala ierobežojumi un Datu tipi. Tie ļauj vēl precīzāk ierobežot vērtības, kā tas bija piemēros 1), 2) un 3). Propertijas vērtību apgabalu varam tālāk ierobežot (ne tikai klašu līmenī), paplašinot, piemēram, owl:someValuesFrom ierobežojumu ar papildus ierobežojumu – owl:withRestrictions . Tam protams nepieciešams arī datu tips (rdfs:Datatype), kā piemēram, xsd:integer (OWL atļauj izmantot XML Schema datu tipu definēšanai); iespējams izveidot un nosaukt arī savus datu tipus, ko var izmantot turpmākos piemēros ontoloģijā [1].

6) Atslēgas – OWL kādu lauku (datu tipa propertiju) vai to salikumu (kā piem. unikalitātes ierobežojums datubāzēs) var izmantot kā unikālu identifikatoru.

owl:hasKey (:aDatatypeProperty1 :aDatatypeProperty2) . (19)

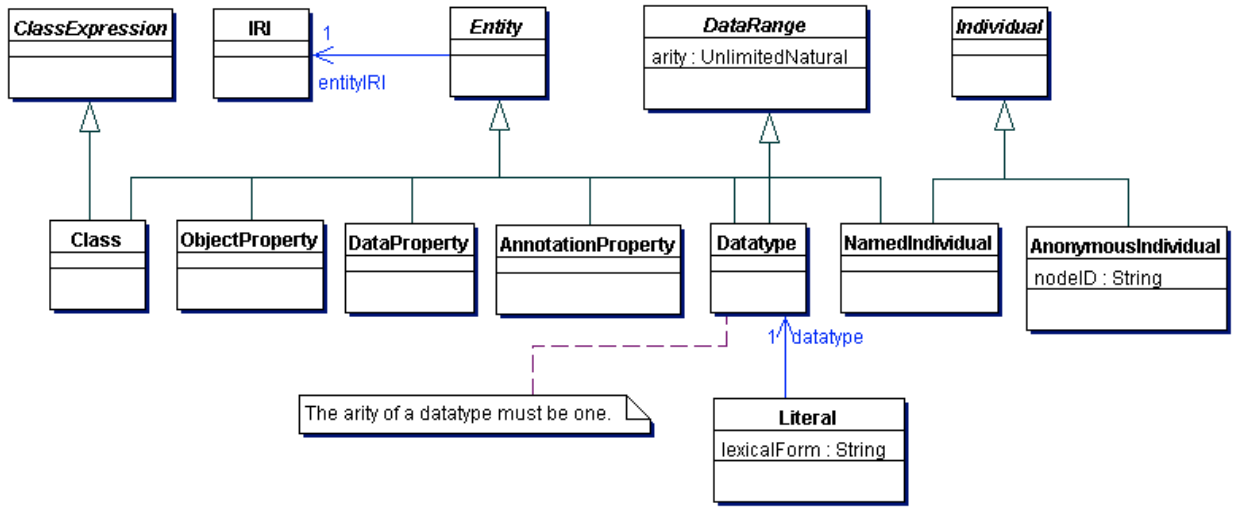
Piemērā :aDatatypeProperty1 :aDatatypeProperty2 varētu veidot salikto (unikālo) atslēgu datu bāzē.

Tika apskatītas valodas OWL konstrukciju grupas – propertijas un to tipi, dažādas aksiomas – aksiomas par propertijām, aksiomas par klasēm, aksiomas par propertijām un klasēm, kā arī iekļauti dažādi apgalvojumi par indivīdiem (klašu un propertiju apgalvojumi).

1.4 OWL konstrukciju strukturāls pārskats[8]

Nodaļā 1.3 netika aplūkotas visas attiecīgās konstrukcijas no OWL (1.5 att.), tās apskatot vairākās loģiskās grupās. Šī nodaļa to papildina.

Valodas specifikācija [15] papildus izmanto jēdzienus, kā nonNegativeInteger (ciparu virkne), quotedString, languageTag, nodeID (galīga rakstzīmju virkne), owl:real, owl:rational, un xsd datu tipus, un izvirza entītiju kā centrālo ontoloģijas sastāvdaļu (skat. 1.8 att. – W3C UML diagr.)



1.8 att. **Entītiju vieta OWL ontoloģijās [15]**

Sekojoši klases definētas kā indivīdu (tātad arī entītiju) kopas, un datu tipi kā datu vērtību kopas.

[8] Satur adaptētus piemērus no OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax [15]

Iepriekš apskatītie indivīdi/instances iedalās 2 grupās - NamedIndividual – nosauktie indivīdi, kuriem ir piešķirts IRI, un AnonymousIndividual, kurus apzīmē NodeID [15].

Literāļi – katru literāli sastāda simbolu virkne un datu tips, vai arī tikai simbolu virkne. Literāļi ir indivīdu datu vērtības [15].

Objektu propertiju izteiksmes var veidot gan iepriekš apskatītās ObjectProperty, gan InverseObjectProperty – ja divi indivīdi a un b ir saistīti ar ObjectProperty x, un b ir saistīts ar a ar objekta propertiju – ObjectInverseOf .

Tehniski valodā ir definētas arī datu tipu izteiksmes (DataPropertyExpression), bet tās pēc būtības ir tas pats, kas DataProperty.

1.4.1 Datu vērtību apgabali (*DataRange*)

Datu vērtību apgabalus veido literāļu pāru kopas. Datu vērtību apgabali var tikt izmantoti datu propertiju ierobežošanā (DataSomeValuesFrom, DataAllValuesFrom).

1.1. tabula

DataRange klases apakšklases OWL metamodelī

Klase	*Apakšklase*	*Apraksts*
Datarange	DataComplementOf	Piedāvā kopu teorijas operācijas datu vērtību apgabalu noteikšanai. Piemēram: `DataIntersectionOf( xsd:nonNegativeInteger xsd:nonPositiveInteger )`, kas apzīmē 0 [15]. Vai arī DataComplementOf, lai iegūtu visas atlikušās vērtības no kāda cita *DataRange*
	DataUnionOf	
	DataIntersectionOf	
	DataOneOf	Tiek izmantots literāļu uzskaitīšanai. *DataOneOf* var saturēt dažādu datu tipu vērtības, jo veidot to kopu.

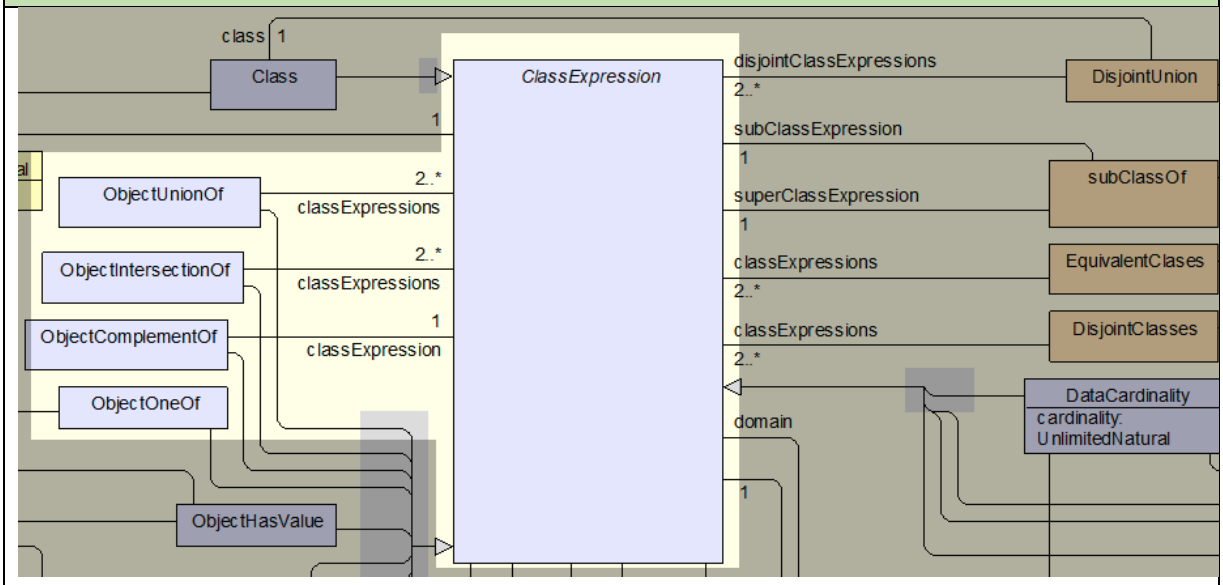
	Datatype	Datu tips ir vienkāršākais *DataRange* piemērs, jo tas ir iepriekš definēts. Datu vērtību apgabali var tikt izmantoti Datu tipu definēšanai.
	DataTypeRestriction	Datu tipa vērtību apgabalu var tālāk ierobežot ar faseti (IRI formā) un atbilstoši ierobežojošo vērtību

1.4.2 Klašu - Indivīdu kopu - izveidošana

Šim nolūkam var tikt izmantotas dažādas klašu izteiksmes. Iepriekš galvenokārt tika apskatītas aksiomas par klasēm. Piemēri papildināti ar iekļautiem sintakses piemēriem.

1.2. tabula

Klases kā objektu kopas OWL 2.0

Novietojums metamodelī		
		
Nosaukums	*Sintakse*	*Piemērs*
Klašu šķēlums	Turtle	Owl:intersectionOf(a:Animal a:CanFly)
	Funkcionālā	ObjectIntersectionOf(a:Animal a:CanFly)
Klašu apvienojums	Turtle	Owl:unionOf(a:Man a:Woman)
	Funkcionālā	ObjectUnionOf(a:Man a:Woman)
Klašu papildinājums	Turtle	Owl:complementOf (a:Human)
	Funkcionālā	ObjectComplementOf(a:Human)

Klases indivīdu uzskaitījums	Turtle	Owl:oneOf(a:Lisa a:Stephen)
	Funkcionālā	ObjectOneOf(a:Lisa a:Stephen)

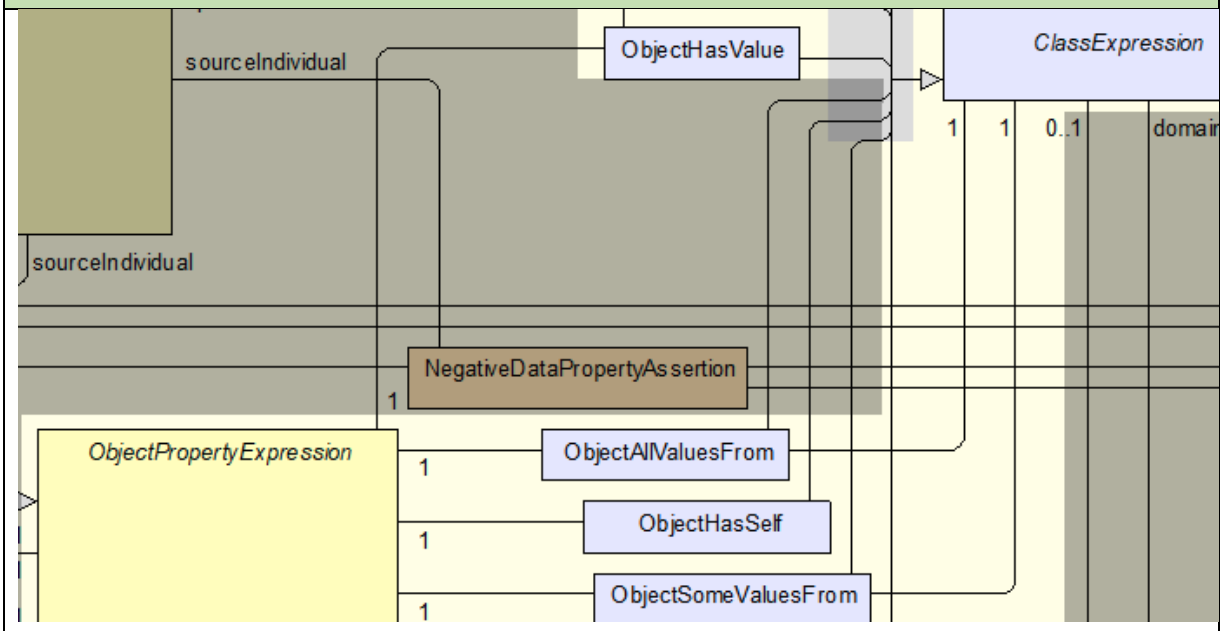
1.4.3 Klašu izteiksmju izveidošana ar propertiju ierobežošanu

Tabulā 1.3 uzskaitīts papildināts saraksts ar izteiksmēm, kas iepriekš tika apskatītas kā klašu aksiomas par propertijām.

1.3. tabula

Klases kā aksiomas par propertijām OWL 2.0

Novietojums metamodelī[9]



Tips	*Sintakse*	*Piemērs*
Universāls ierobežojums	Turtle	Owl:onProperty a:hasPet ; Owl:allValuesFrom a:Dog ;
	Funkcionālā	ObjectAllValuesFrom(a:hasPet a:Dog)
Eksistences ierobežojums	Turtle	Owl:onProperty a:fatherOf ; Owl:someValuesFrom a:Man ;
	Funkcionālā	ObjectSomeValuesFrom(a:fatherOf a:Man)

[9] grafiski apstrādāts 1.5 attēls

Vērtības ierobežojums	Turtle	Owl:onProperty a:fatherOf ; Owl:hasValue a:Stewie ;
	Funkcionālā	ObjectHasValue(a:fatherOf a:Stewie)
Ierobežojums pašam uz sevi[1]	Turtle	Owl:onProperty a:likes ; Owl:hasSelf “true”^^xsd:boolean;
	Funkcionālā	ObjectHasSelf(a:likes)

1. Iegūstam tikai tos indivīdus kas saistīti paši ar sevi ar propertiju a:likes.

1.4.4 Datu Propertiju aksiomas

Gan datu propertijas, gan objektu propertijas manto visas galvenās propertiju īpašības. Datu propertijām gan, saprotams, nepiemīt visas tās pašas aksiomas, kas piemīt objektu propertijām.

1.4. tabula

Propertiju apakšklašu īpašības OWL 2.0

Īpašība	*Objektu propertijas*	*Datu propertijas*
Apakšklase	X	X
Ekvivalence	X	X
Nepārklājošas	X	X
Inverss	X	
Domēns	X	X
Vērtību apgabals	X	X
Funkcionāla	X	X
Inversi-funkcionāla	X	
Refleksīva	X	
Neatgriezeniska	X	
Simetriska	X	
Asimetriska	X	
Tranzitivitāte	X	

1.4.5 Apgalvojumi ontoloģijā

Atsevišķi jāapskata dažādās iespējas izteikt apgalvojumus (faktus, nevis aksiomas) ontoloģijā.

1.5. tabula

Dažādi apgalvojumi OWL 2.0

Apgalvojums		
Ekvivalence	SameIndividual	Var norādīt sarakstu ar IRI, kas apzīmē vienu un to pašu indivīdu
Nevienādība	DifferentIndividuals	Pretējs ekvivalencei
Klašu apgalvojums	ClassAssertion	Apskatīts iepriekš
Propertijas apgalvojums	ObjectPropertyAssertion	Apskatīts iepriekš
Propertijas noliegums	NegativeObjectPropertyAssertion	Pretējs propertijas apgalvojumam
Datu propertijas apgalvojums	DataPropertyAssertion	Apskatīts iepriekš
Datu propertijas noliegums	NegativeDataPropertyAssertion	Pretējs datu propertijai

1.4.6 Anotācijas

Anotācijas ļauj izteikt papildus informāciju par ontoloģijām, entītijām un aksiomām.
Anotācijas ir “pirmās klases pilsoņi”[15] valodā OWL. Anotācijas vērtība var būt literālis, IRI, vai nenosauktie indivīdi.
Anotācijas apgalvojumi pierakstāmi formā

$$\text{AnnotationAssertion}(\ \text{AnotationProperty aSubject aValue}\ ) \tag{20}$$

, kur AnnotationProperty ir attiecīgais kāda anotācijas veida URI.

2. GRAPHOL REDAKTORS EDDY

Kā pirmā no relatīvi sarežģītākajām grafiskajām valodām tiks apskatīta Graphol un tās redaktors Eddy. Eddy rīks strādā ar Graphol (.GraphML) failiem. Graphol valoda ir radīta, konceptuāli balstoties uz ER diagrammām, papildinot ER sintaksi ar grafiskajām konstrukcijām, kas attēlotu visas attiecīgās OWL konstrukcijas, t.sk. arī sarežģītas izteiksmes [17, 19].
Ontoloģijas tiek attēlotas grafa veidā, izteiksmju mezglus savienojot ar operatoriem ar vērstām, raustītām šķautnēm, kas noslēgtas ar dimanta simbolu, un apgalvojumus ar vērstu bultu.

2.1 Graphol sintakses primitīvi

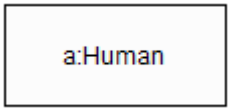
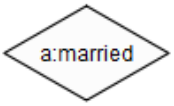
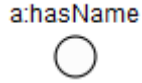
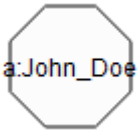
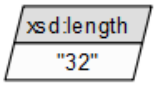
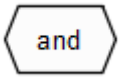
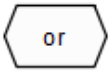
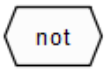
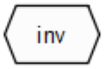
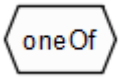
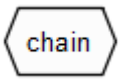
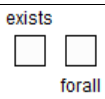
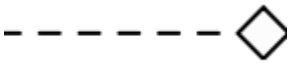
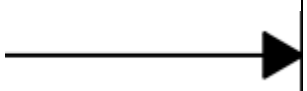
Graphol ontoloģija tiek definēta kā Graphol apgalvojumu kopums. Graphol grafā mezgli tiek iedalīti divos veidos – ‘predikātu mezglos’, un ‘konstruktoru mezglos’.
Predikātu mezgli attēlo dažādos ontoloģijas jēdzienus. Konstruktoru mezgli ir paredzēti kompleksu izteiksmju attēlošanai, un savukārt iedalās predikātu mezglos un operatoru mezglos (2.1 tabula), no kā pirmās respektīvi attēlo konceptus, kas veido ontoloģijas vārdnīcu (kā atsevišķās klases, propertijas, indivīdi), bet otrie loģiskās operācijas, kādas pieejamas OWL, kā apvienojums un šķēlums, kā arī piemēram, aksiomas kā propertiju ķēdes [5, 18].
Galvenās vizualizācijas iezīmes konstrukcijām ir:

- Netiek izmantotas krāsas (atšķiras tikai aizpildījums ierobežotājiem, predikātiem)
- Konstrukcijas tiek diferencētas pēc to izmēra (piem. lomu mezgli grafā un simboli pie savienotājiem)
- Dažādas ģeometriskās formas apzīmē dažāda tipa konstrukcijas.

Graphol valodas primitīvi attēloti tabulā 2.1, par pamatu ņemot Graphol valodas specifikācijas dokumentu [18], izmantojot autora ekrānuzņēmumus visām konstrukcijām, jo specifikācija nav atjaunota līdz ar jaunākām programmas versijām.

2.1. tabula

Graphol primitīvi – predikātu un konstruktoru mezgli

Graphol notācija	*OWL konstrukcija*	*Graphol notācija*	*OWL konstrukcija*
Predikātu mezgli			
	Klase		Objekta propertija
	Datu propertija		Datu tips
	Indivīds vai Literālis		Datu tipa ierobežojums
Graphol notācija	*OWL konstrukcija*	*Graphol notācija*	*OWL konstrukcija*
Konstruktoru mezgli			
	Šķēlums		Datu tipa ierobežojums
	Apvienojums		Papildinājums
	Inversā propertija		Uzskaitījums
	Propertiju ķēde		Domēns
	Vērtību apgabals		
Graphol notācija	*OWL konstrukcija*	*Graphol notācija*	*OWL konstrukcija*
Savienotāji			
	Savieno predikāta vai konstruktora mezglu ar konstruktora mezglu (no kreisās uz labo)		Savieno izteiksmes

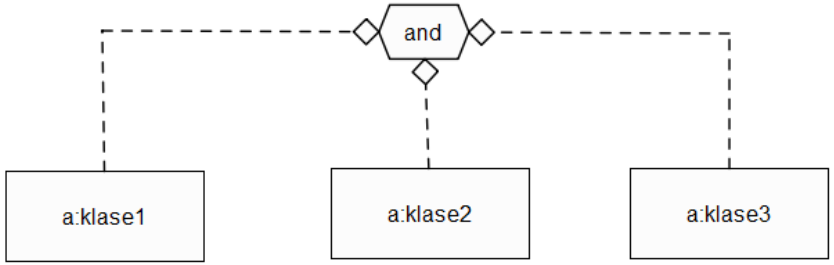
Atšķirīgo savienotāju ieviešana padara notāciju sarežģītāku, bet modulārāku.

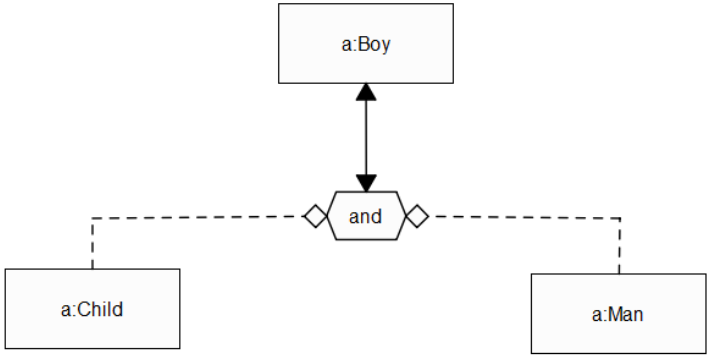
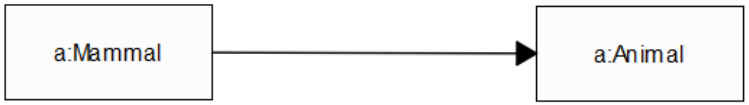
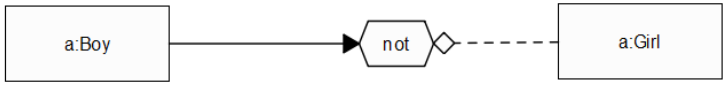
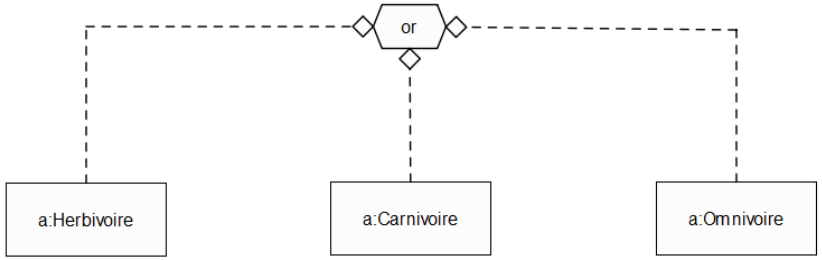
2.2 OWL2 konstrukcijas Graphol (Eddy)

Vispirms tiks apskatītas dažādas aksiomas (t.sk. klašu) un izteiksmes un to Graphol attēlojums Eddy rīkā, kā arī atsevišķi piemēri šādu konstrukciju apvienošanai. Visi piemēri ir autora modelēti (tabulā attēloti ekrānuzņēmumu fragmenti no Eddy redaktora).

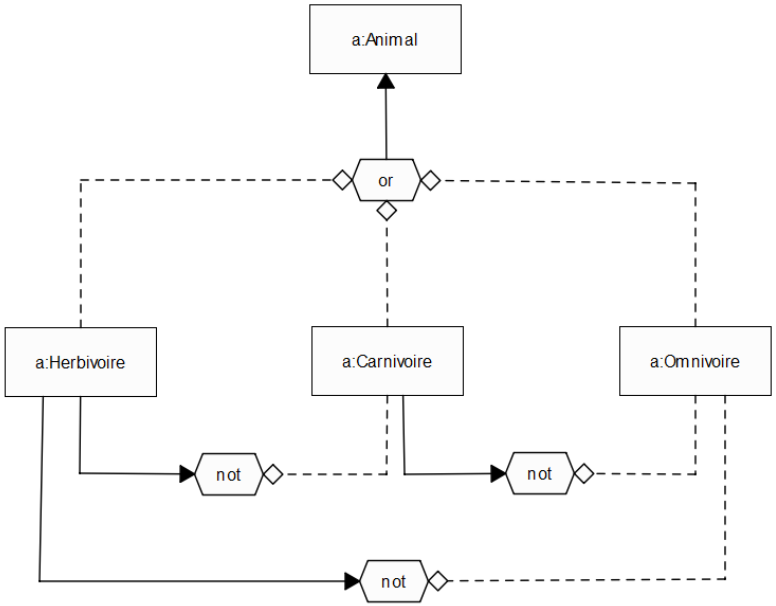
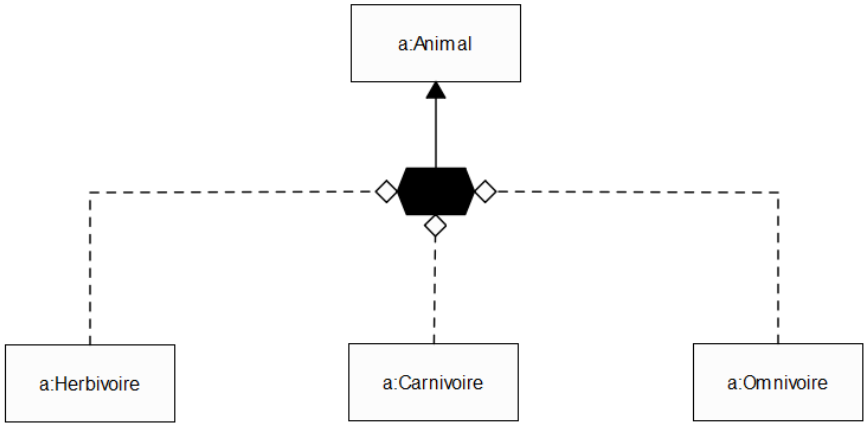
2.2. tabula

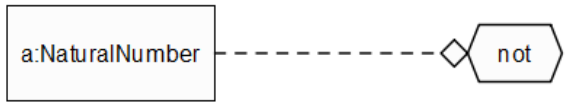
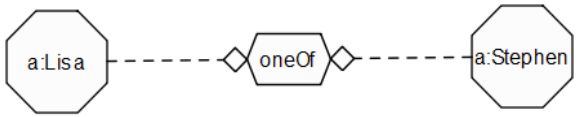
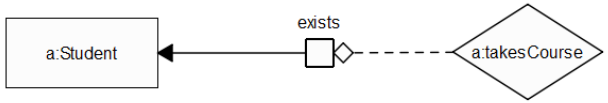
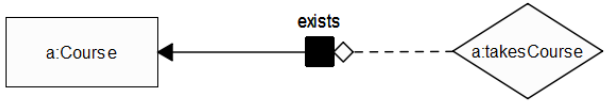
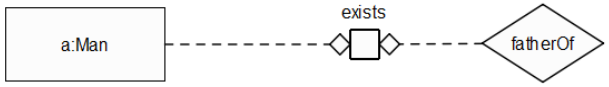
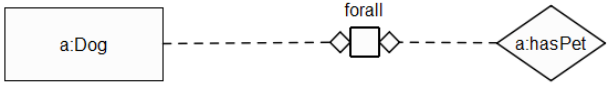
Dažādas izvēlētas OWL 2.0 konstrukcijas un to grafiskais attēlojums rīkā Eddy

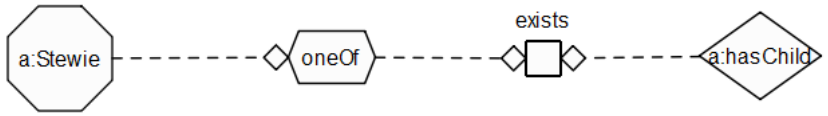
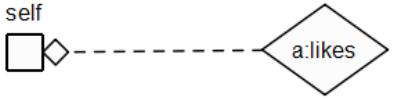
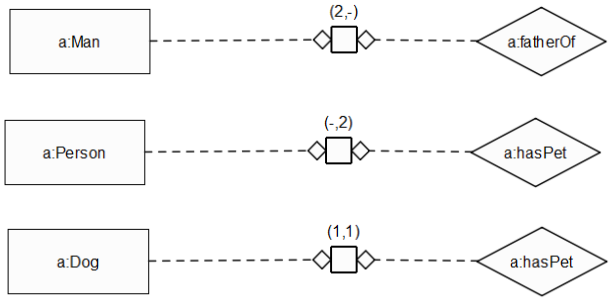
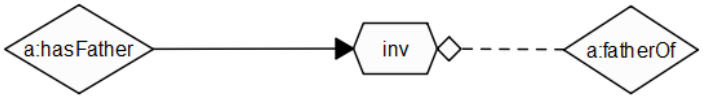
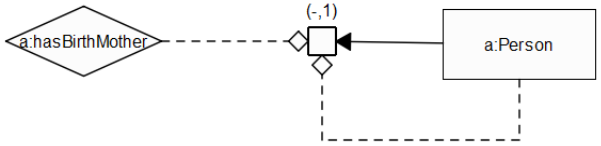
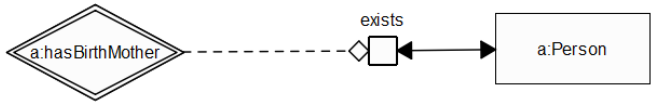
Nos.	*Konstrukcija Eddy rīkā*	*Atbilstošie OWL2 izteikumi*
Ekvivalentas klases (aks.)		EquivalentClasses(a:klase1 a:klase2)
Klašu šķēlums I (izt.)		ObjectIntersectionOf(a:klase1 a:klase2 a:klase3))

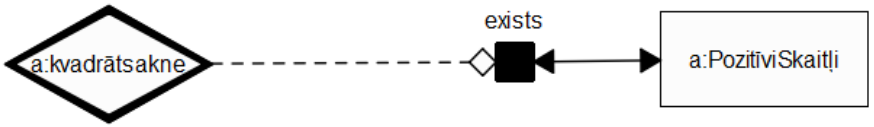
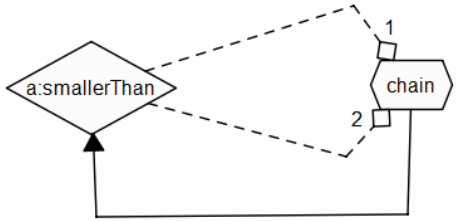
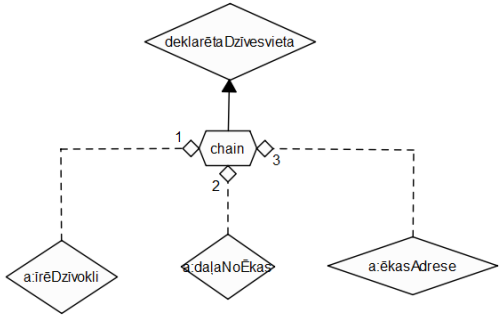
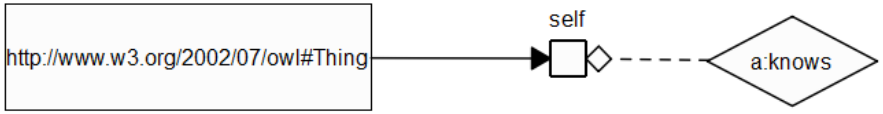
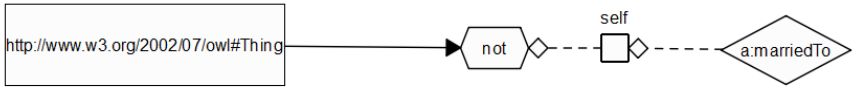
Klašu šķēlums II (izt.)		Iepriekšējās konstrukcijas varam apvienot vienā izteikumā [Declaration(Class(a:Boy)) Declaration(Class(a:Child)) Declaration(Class(a:Man))][10] EquivalentClasses(a:Boy ObjectIntersectionOf(a:Child a:Man))
Apakšklase		SubClassOf(a:Mammal a:Animal)
Nepārklājošas klases		DisjointClasses(a:Boy a:Girl)
Klašu apvienojums (izt.)		ObjectUnionOf(a:Carnivoire a:Herbivoire a:Omnivoire)

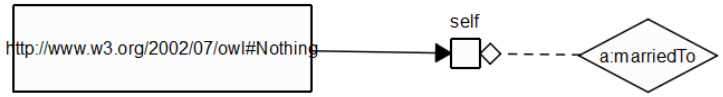
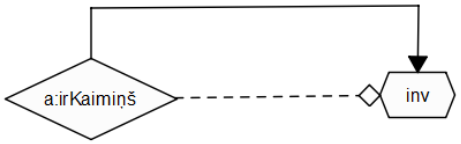
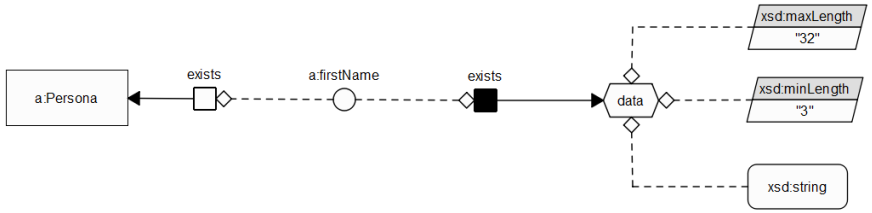
[10] Turpmākos piemēros elementārie apgalvojumi tiek izlaisti

Nepārklājošu klašu apvienojums (I)		DisjointClasses(a:Carnivoire a:Herbivoire) DisjointClasses(a:Carnivoire a:Omnivoire) DisjointClasses(a:Herbivoire a:Omnivoire) EquivalentClasses(a:Animal ObjectUnionOf(a:Carnivoire a:Herbivoire a:Omnivoire))
Nepārklājošu klašu apvienojums (II) Graphol saīsnes konstrukcija		SubClassOf(ObjectUnionOf(a:Carnivoire a:Herbivoire a:Omnivoire) a:Animal) DisjointClasses(a:Carnivoire a:Herbivoire a:Omnivoire)

Klašu papildinājums		ObjectComplementOf(a:NaturalNumber)
Klases indivīdu uzskaitījums		ObjectOneOf(a:Lisa a:Stephen)
Domēns		ObjectPropertyDomain(a:takesCourse a:Student)
Vērtību apgabals		ObjectPropertyRange(a:takesCourse a:Course)
Domēns un vērtību apgabals		ObjectPropertyDomain(a:takesCourse a:Student) ObjectPropertyRange(a:takesCourse a:Course)
Eksistences ierobežojums		ObjectSomeValuesFrom(a:fatherOf a:Man)
Universālais ierobežojums		ObjectAllValuesFrom(a:hasPet a:Dog)

Konkrētas vērtības ierobežojums		ObjectHasValue(a:hasChild a:Stewie)
Ierobežojums pašam uz sevi[1]		ObjectHasSelf(a:likes)
Kardinalitātes ierobežojums (minimālā, maksimālā, tiešā)		ObjectMinCardinality(2 a:fatherOf a:Man) ObjectMaxCardinality(2 a:hasPet a:Person) ObjectExactCardinality(1 a:hasPet a:Dog)
Inversas propertijas		InverseObjectProperties(a:hasFather a:fatherOf)
Funkcionāla propertija (I)		SubClassOf(a:Person ObjectMaxCardinality(1 a:hasBirthMother a:Person))
Funkcionāla propertija (II) Graphol saīsne		FunctionalObjectProperty(a:hasBirthMother) ObjectPropertyDomain(a:hasBirthMother a:Person)

Inversi funkcionāla propertija		InverseFunctionalObjectProperty(a:kvadrātsakne)
Tranzitīva propertija		SubObjectPropertyOf(ObjectPropertyChain (a:smallerThan a:smallerThan) a:smallerThan)
Propertiju ķēde		SubObjectPropertyOf(ObjectPropertyChain (a:īrēDzīvokli a:daļaNoĒkas a:ēkasAdrese) a:deklarētaDzīvesvieta))
Refleksīva propertija		Netiek izmantota OWL 2.0 valodas specifikācijā [13] definētā saīsne ReflexiveObjectProperty(a:knows), bet SubClassOf(a:Person ObjectHasSelf(a:knows))
Neatgriezeniska propertija (I)		DisjointClasses(owl:Thing ObjectHasSelf(a:marriedTo))

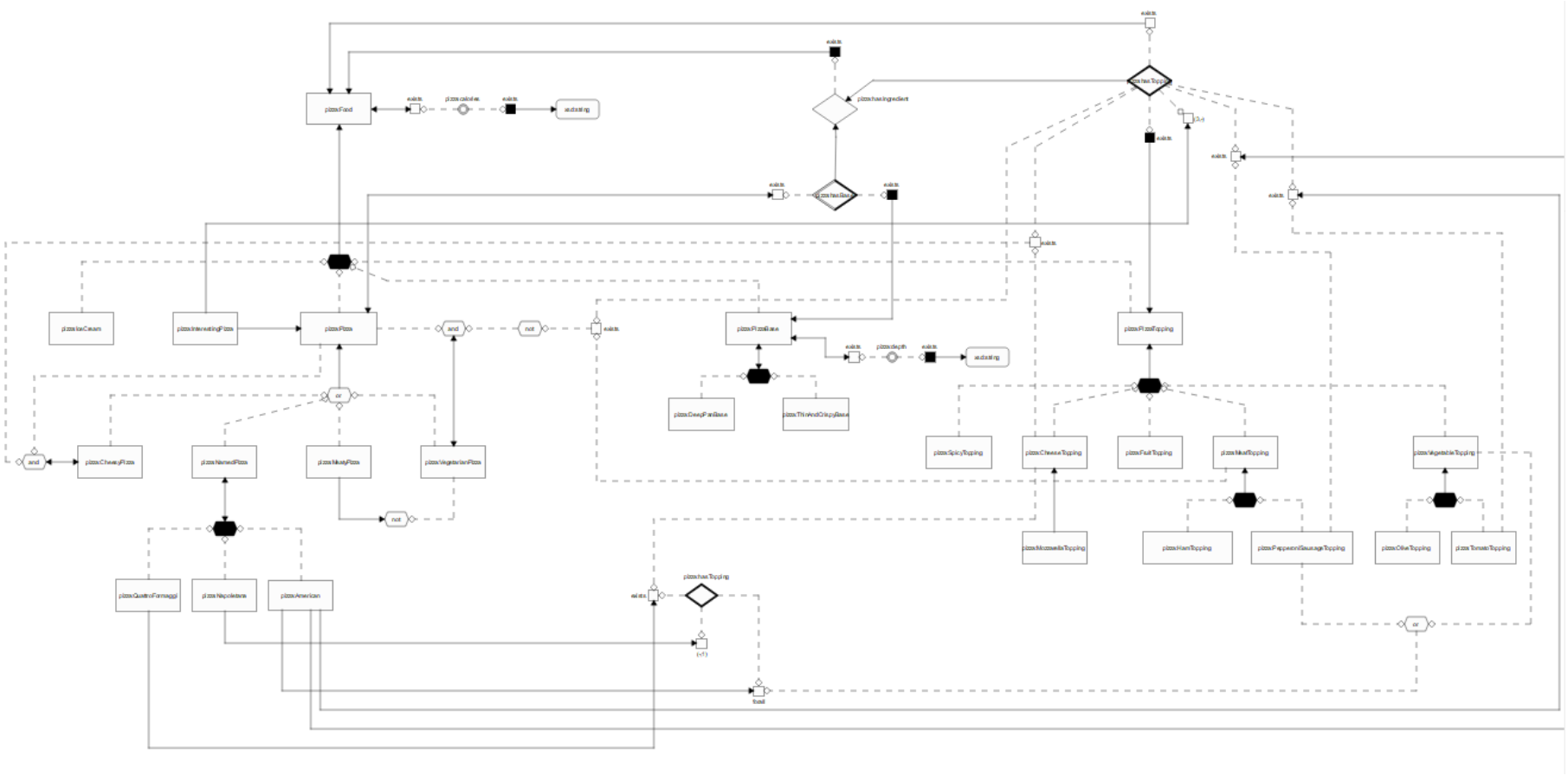
Neatgriezeniska propertija (II)		SubClassOf(owl:Nothing ObjectHasSelf(a:marriedTo))
Simetriska propertija		SubObjectPropertyOf(a:irKaimiņš ObjectInverseOf(a:irKaimiņš)) nevis SymmetricObjectProperty(a:irKaimiņš)
Datu propertijas ierobežojums		DataPropertyRange(a:firstName DatatypeRestriction(xsd:string xsd:minLength "3"^^xsd:string xsd:minLength "32"^^xsd:string))

2.3 Ontoloģiju vizualizācijas rīkā Eddy

Katram no rīkiem tiek apskatītas konkrētu ontoloģiju vizualizācijas. Tās palīdz kopumā novērtēt ontoloģiju grafisko notāciju sarežģītību un lietošanas ērtumu.

2.3.1 Picu ontoloģija

Viens no populārākajiem piemēriem savas vienkāršības dēļ ir ‘Picu ontoloģija’ [10]. Ontoloģija ņemta no Eddy rīka publiskajiem parauga piemēriem, kas iekļauti rīka pakotnē.



2.1 att. **Picu ontoloģija rīkā Eddy, rīka parauga piemērs**

2.4 Secinājumi

Eddy rīks ir diezgan ērts. Attiecīgās Graphol grafiskās konstrukcijas ļauj lielā mērā grafiski attēlot arī sarežģītus OWL izteikumus. Tomēr tikai atsevišķām OWL konstrukcijām ir definētas tam konkrēti paredzētas grafiskās konstrukcijas, tāpēc pat samērā vienkārši izteikumi ir jābūvē no pamata konstrukcijām, neizmantojot OWL2 piedāvātās saīsnes, kas būtiski palielina grafisko elementu skaitu (un jāievieš jaunas grafiskās konstrukcijas, lai biežāk izmantotajos gadījumos kompleksās konstrukcijas saīsinātu). Sekojoši arī ierobežojumi, kā piemēram, nav tiešā veidā iespējams importēt RDF (vai OWL) Ontoloģijas, lai tās attēlotu.

3. OWLGRED ONTOLOĢIJU REDAKTORS

OWLGrEd rīks izmanto paplašinātu populāro UML grafisko notāciju ontoloģiju attēlošanai, efektīvi izmantojot UML konstrukcijas maksimāli kompaktai OWL ontoloģiju vizualizācijai, kas tiek sasniegta, papildinot to ar Mančesteras sintakses elementiem klašu izteiksmēm [3]. Tātad UML klašu diagrammu notācija papildināta ar papildus notācijas elementiem un vietām izteiksmju izvietošanu.

OWLGrEd ir iekļauti papildus servisi kā piem. izkārtojuma un meklēšanas iespējas, un sadarbspēja ar vienu no 'industrijas standarta' rīkiem – Protégé (līdz 5.0.0 versijai)[11]. Rīkā var importēt OWL, RDF un XML sintakses ontoloģijas. Redaktors būvēts izmantojot TDA un atbalsta papildinājumu izveidošanu.

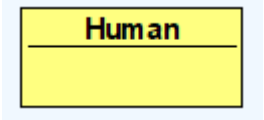
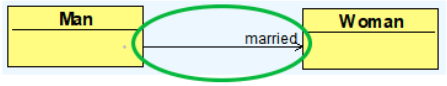
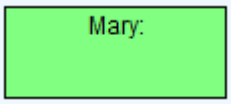
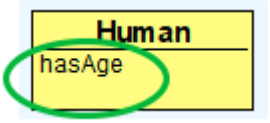
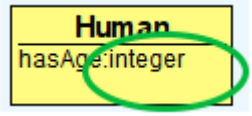
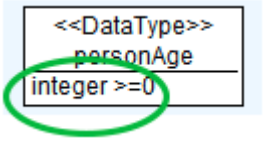
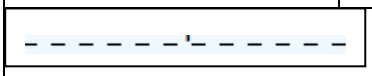
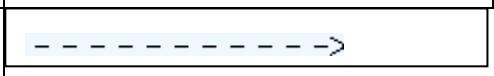
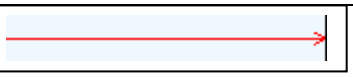
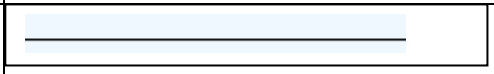
[11] http://owlgred.lumii.lv/get_started

3.1 OWLGrEd paplašinātās UML sintakses primitīvi

Galvenie UML elementi tiek izmantoti tiešā veidā – kā UML klases, un propertijas kā UML asociācijas, kas būtiski samazina grafisko elementu skaitu. Tabulā 3.1 attēlotie galvenie grafiskās notācijas primitīvi (autora ekrānuzņēmumu izgriezumi, veidoti līdzīgi rīka izstrādātāju notācijas[12] piemēriem, papildinot ar elementiem, kas izmantoti turpmākos piemēros).

3.1. tabula

OWLGrEd grafiskās notācijas primitīvi

grafiskā notācija	*pielietojums*	*grafiskā notācija*	*pielietojums*
	Klase		Propertijas apgalvojums
	Indivīds		Datu propertija
	Datu tips		Datu tipa ierobežojums
	Atkarībā no konteksta var tikt izmantota klašu vai indivīdu savienošanai		Savienotājs klases instancei ar klasi
	Ierobežotājs		Objekta/datu propertija (lietota tikai kopā ar iezīmi)
	Anotācijas propertija		

Sekojoši, piemēram, dažādas propertijas tiek diferencētas pēc to iezīmēm.

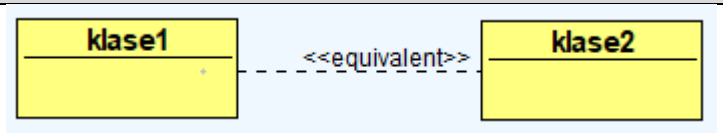
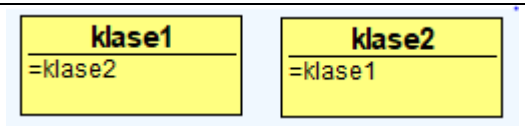
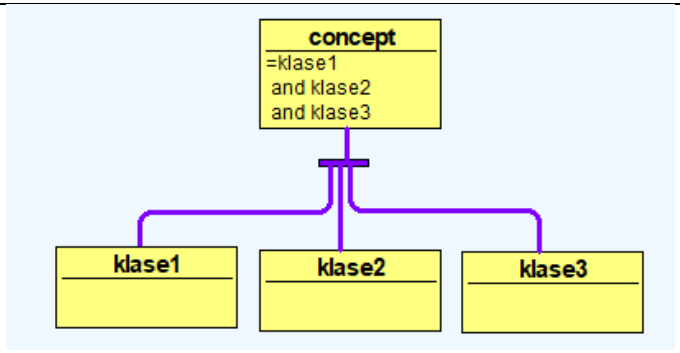
[12] http://owlgred.lumii.lv/notation

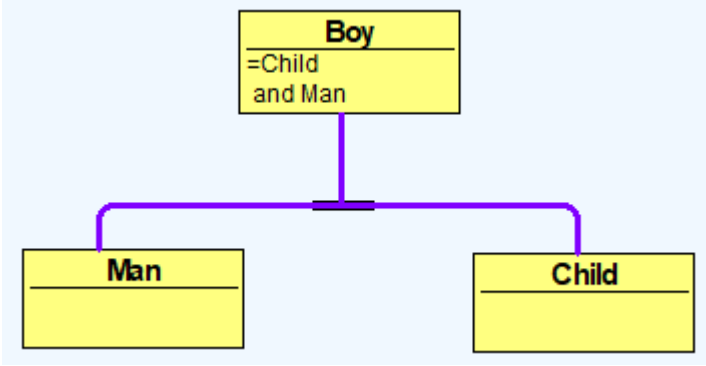
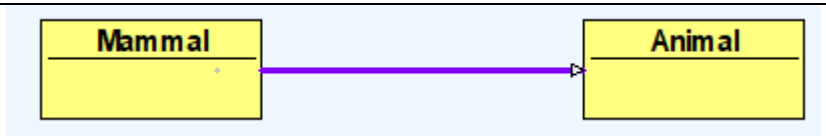
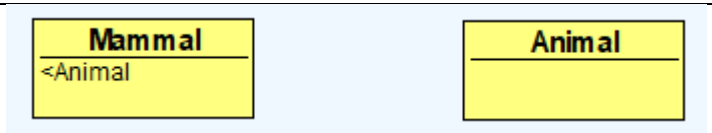
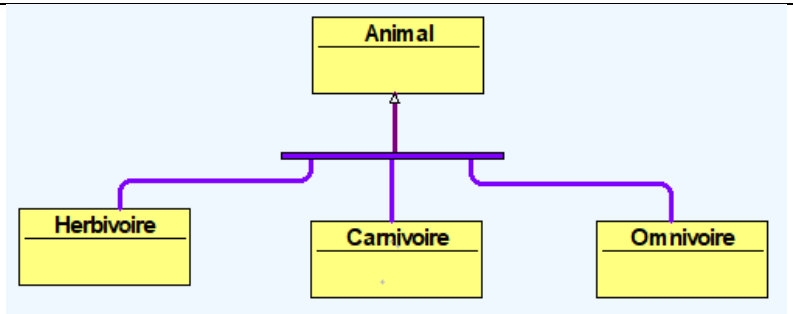
3.2 OWL2 konstrukcijas OWLGrEd

Tiek apskatīts dažāda veida izteiksmju attēlojums OWLGrEd rīkā. Sākumā apskatīti piemēri, kas atbilst 2.2 nodaļā apskatītajiem. Visi piemēri ir autora modelēti (tabulā attēloti ekrānuzņēmumu fragmenti no redaktora).

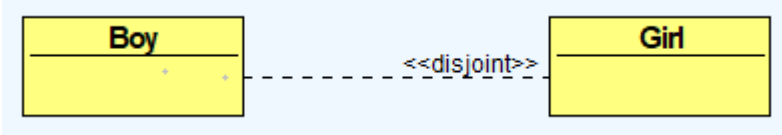
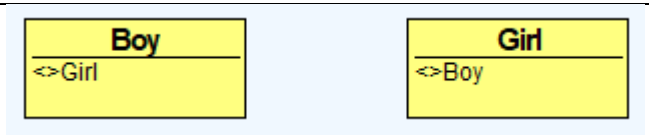
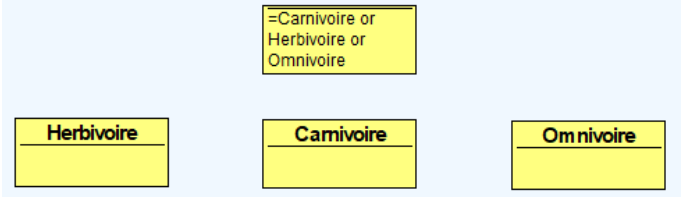
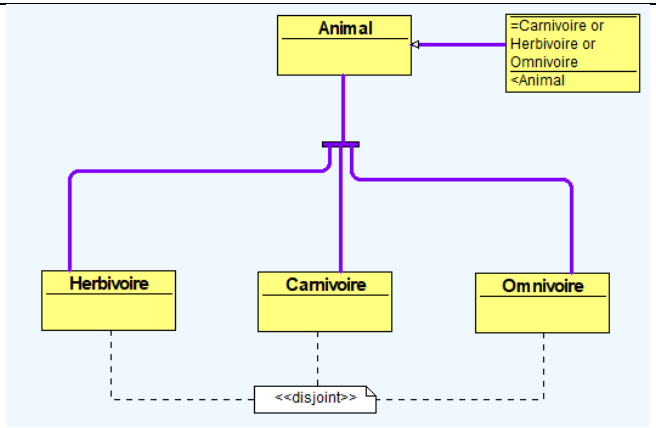
3.2. tabula

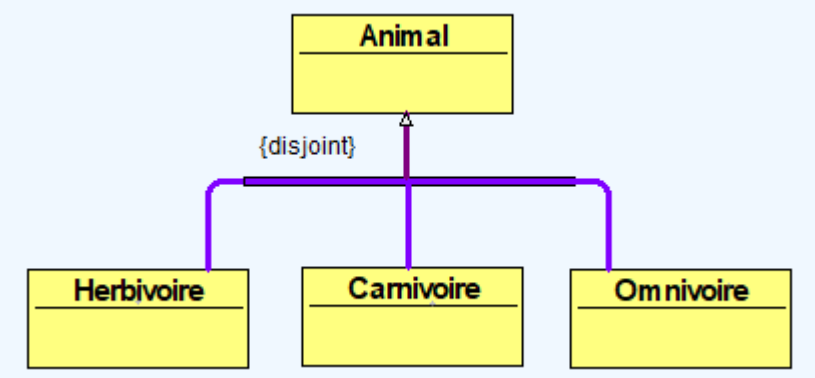
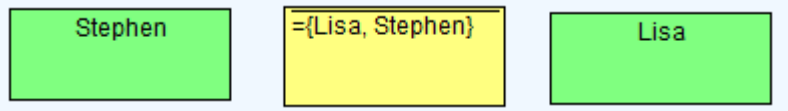
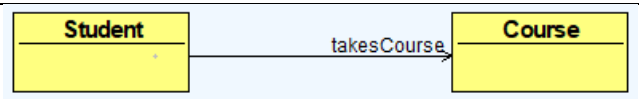
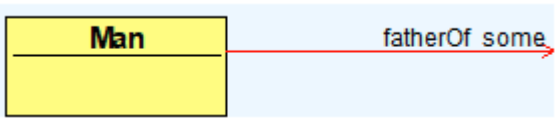
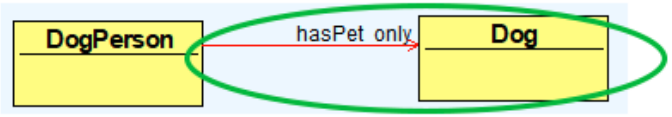
Izvēlētas OWL 2.0 konstrukcijas un to grafiskais attēlojums OWLGrEd redaktorā

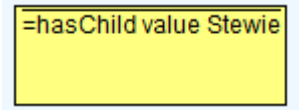
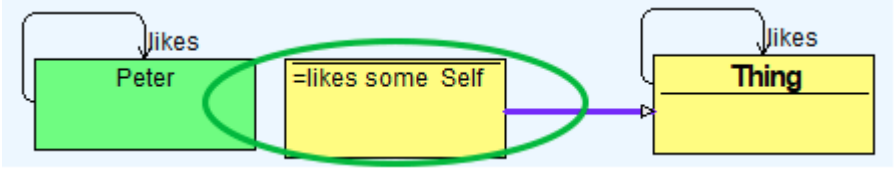
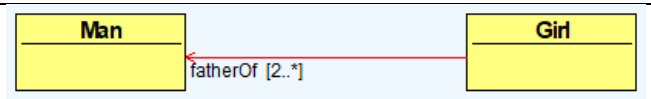
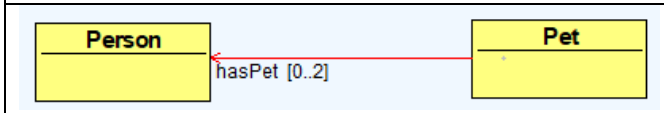
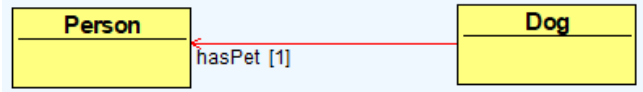
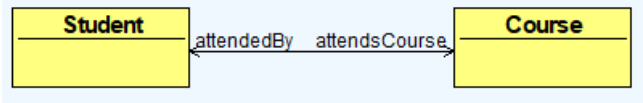
Nos.	*Konstrukcija OWLGrEd rīkā*	*OWL sintakse, paskaidrojumi*
	Eddy rīkā apskatītās konstrukcijas	
Ekvivalentas klases (I)(aks.)		EquivalentClasses(a:klase1 a:klase2)
Ekvivalentas klases (II)(aks.)		—”—
Klašu šķēlums I (izt.)		ObjectIntersectionOf(a:klase1 a:klase2 a:klase3))

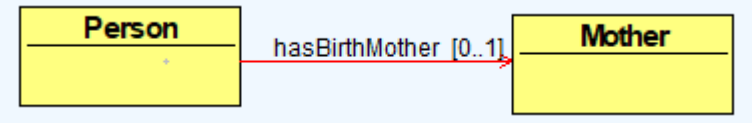
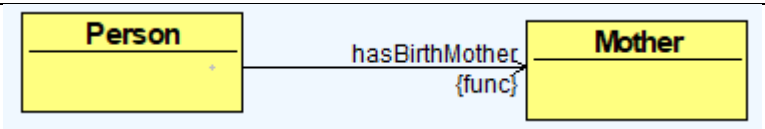
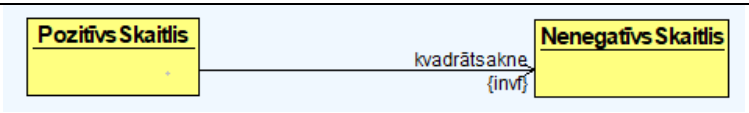
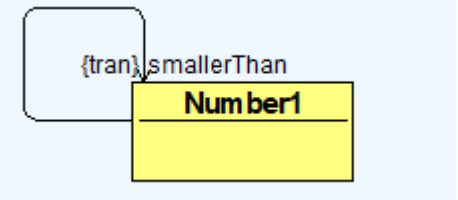
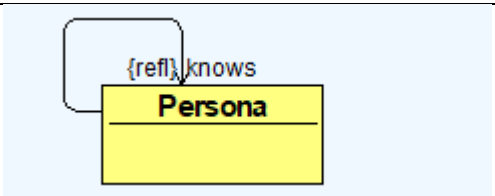
Klašu šķēlums II (izt.)		Iepriekšējās konstrukcijas varam apvienot vienā izteikumā [Declaration(Class(a:Boy)) Declaration(Class(a:Child)) Declaration(Class(a:Man))][13] EquivalentClasses(a:Boy ObjectIntersectionOf(a:Child a:Man))
Apakšklase (I)		SubClassOf(a:Mammal a:Animal) Declaration(Class (Mammal)) Declaration(Class(Animal)) SubClassOf(Mammal Animal)
Apakšklase (II)		Declaration(Class (Mammal)) SubClassOf(Mammal Animal) Declaration(Class(Animal))
Apakšklase (III)		SubClassOf(Herbivoire Animal) SubClassOf(Carnivoire Animal) SubClassOf(Omnivoire Animal)

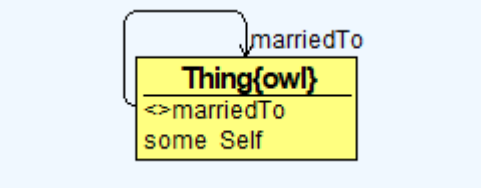
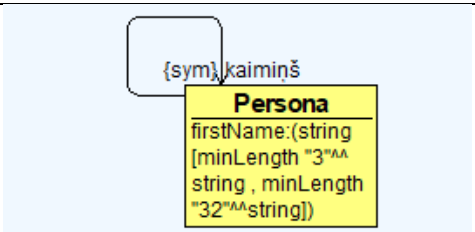
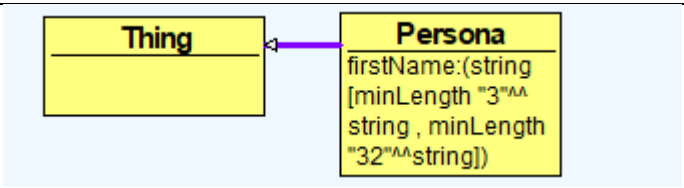
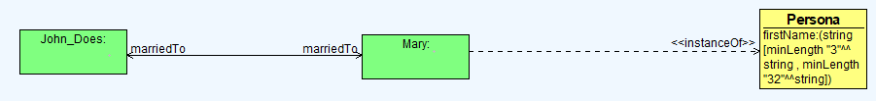
[13] Turpmākos piemēros elementārie apgalvojumi lielākoties tiek izlaisti

Nepārklājošas klases (I)		DisjointClasses(a:Boy a:Girl) *OWLGrEd tīmekļa versijā <'disjoint> propertijas caurspīdīgas*
Nepārklājošas klases (II)		DisjointClasses(a:Boy a:Girl)
Klašu apvienojums (izt.)		ObjectUnionOf(a:Carnivoire a:Herbivoire a:Omnivoire)
Nepārklājošu klašu apvienojums (I)		DisjointClasses(a:Carnivoire a:Herbivoire) DisjointClasses(a:Carnivoire a:Omnivoire) DisjointClasses(a:Herbivoire a:Omnivoire) EquivalentClasses(a:Animal ObjectUnionOf(a:Carnivoire a:Herbivoire a:Omnivoire))

Nepārklājošu klašu apvienojums (II)		DisjointClasses(< http://lumii.lv/ontologies/ontology.owl#Herbivoire >< http://lumii.lv/ontologies/ontology.owl#Carnivoire >< http://lumii.lv/ontologies/ontology.owl#Omnivoire >)
Klašu papildinājums		ObjectComplementOf(a:NaturalNumber)
Klases indivīdu uzskaitījums		ObjectOneOf(a:Lisa a:Stephen)
Domēns un vērtību apgabals		ObjectPropertyDomain(a:takesCourse a:Student) ObjectPropertyRange(a:takesCourse a:Course)
Eksistences ierobežojums		ObjectSomeValuesFrom(a:fatherOf a:Man)
Universālais ierobežojums		ObjectAllValuesFrom(a:hasPet a:Dog)

Konkrētas vērtības ierobežojums		ObjectHasValue(a:hasChild a:Stewie)
Ierobežojums pašam uz sevi[1]		ObjectHasSelf(a:likes) *Anonīmā klase iekļaus tādus indivīdus, kuri paši sev patīk, tādad arī indivīdu :Peter*
Kardinalitātes ierobežojums (minimālā, maksimālā, tiešā)		SubClassOf(< http://lumii.lv/ontologies/ontology.owl#Girl > ObjectMinCardinality(2 < http://lumii.lv/ontologies/ontology.owl#fatherOf > <http://lumii.lv/ontologies/ontology.owl#Man >))
		ObjectMaxCardinality(2 a:hasPet a:Person)
		ObjectExactCardinality(1 a:hasPet a:Dog)
Inversas propertijas		InverseObjectProperties(a:hasFather a:fatherOf) *Tādas propertijas, kas atrodas uz viena savienotāja*

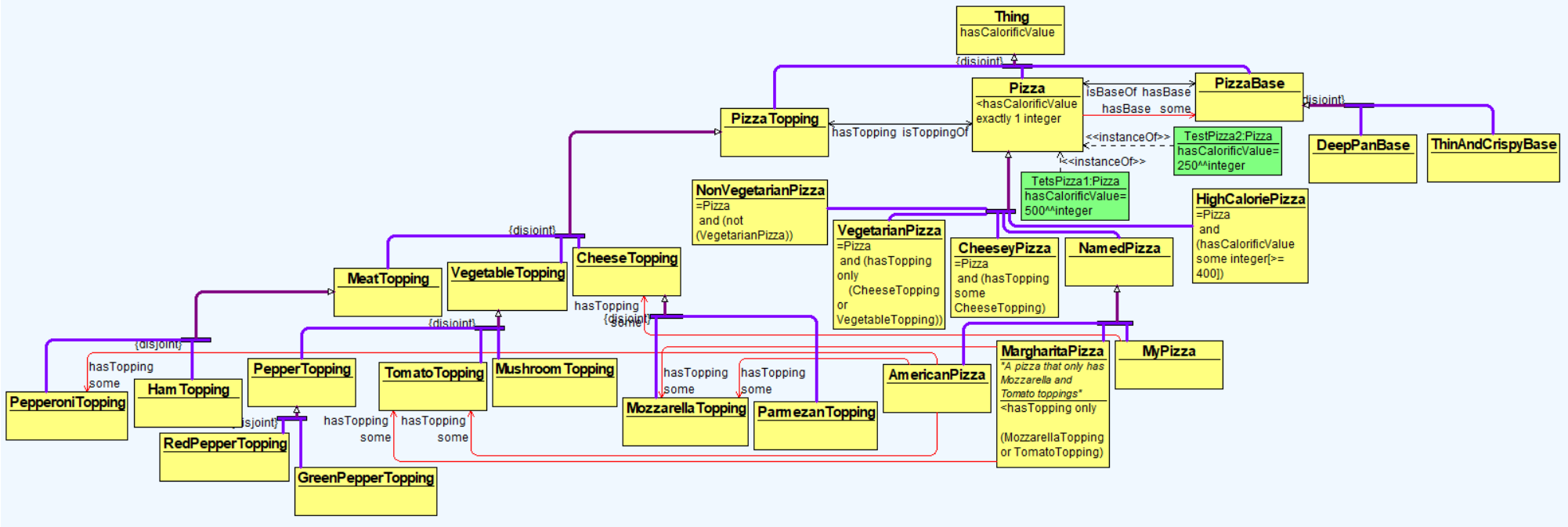
Funkcionāla propertija (I)		SubClassOf(a:Person ObjectMaxCardinality(1 a:hasBirthMother a:Person)) *Iezīme {func} pie predikāta*
Funkcionāla propertija (II) OWLGrEd saīsne		FunctionalObjectProperty(a:hasBirthMother) ObjectPropertyDomain(a:hasBirthMother a:Person) EquivalentClasses(a:Person ObjectSomeValuesFrom(a:hasBirthMother owl:Thing))
Inversi funkcionāla propertija		InverseFunctionalObjectProperty(a:kvadrātsakne) *Iezīme {invf} pie predikāta (savienotāja)*
Tranzitīva propertija		TransitiveObjectProperty(< http://lumii.lv/ontologies/ontology.owl#smallerThan >) *Iezīme {tran} pie predikāta (savienotāja)*
Propertiju ķēde	*N/A*	*Konstrukcija netiek attēlota*
Refleksīva propertija		ReflexiveObjectProperty(knows) , *Iezīme {refl} pie predikāta (savienotāja)*

Neatgriezeniska propertija (I)		DisjointClasses(owl:Thing ObjectHasSelf(a:marriedTo))
Simetriska propertija		SymmetricObjectProperty(a:irKaimiņš)
Datu propertijas ierobežojums		DataPropertyRange(a:firstName DatatypeRestriction(xsd:string xsd:minLength "3"^^xsd:string xsd:minLength "32"^^xsd:string))
Papildus apskatītās konstrukcijas		
Objekta propertijas apgalvojums		Individual: <[..]/#John_Does> Facts: <[..]/#marriedTo> <[..]#Mary> [*Mančesteras sintakse*]
Klases instance		Individual: <[..]/#Mary> Types: <[..]/#Persona> [*Mančesteras sintakse*]

3.3 Ontoloģiju vizualizācijas rīkā OWLGrEd

3.3.1 Picu ontoloģija

Ontoloģija attēlota izmantojot attēlošanas servisu vertikālam apakšklašu izvietojumam. Attēlā 3.1. redzama rīka pakotnē iekļautā ontoloģija.



3.1 att. **Picu ontoloģija rīkā OWLGrEd, rīka parauga piemērs**

Līdzīgi kā Protégé spraudnī, labi redzama atšķirība dažāda veida savienotājiem, pateicoties krāsu izmantojumam.

3.4 Secinājumi

OWLGrEd rīkā izmantotā paplašinātā UML grafiskā notācija ļauj salīdzinoši ļoti kompaktā veidā attēlot ontoloģijas. Papildus izteiksmju informācija tiek attēlota uz savienotājiem vai pie klasēm. Sekojoši, piemēram, objektu propertijām un datu propertijām nav atsevišķa grafiskā elementa, tāpēc vienādas propertijas var atšķirt pēc attiecīgām iezīmēm uz savienotājiem. Nodrošināta RDF/OWL importēšana un eksportēšana vienotam darbam ar semantiskā tīmekļa dokumentiem.

4. VOWL VIZUALIZATORS WEBVOWL

WebVOWL, līdzīgi kā Eddy, attēlo ontoloģijas orientēta grafa veidā, bet objektu propertijas apzīmē ‘kastes’ virs savienotājiem. WebVOWL ir realizēts reaģējošas tīmekļa vietnes veidā [23], un paredzēts vizualizācijai VOWL grafiskajā notācijā [20].
Rīkā iespējams dinamiski aplūkot attiecīgo ontoloģiju, pārvietot un iezīmēt tās elementus. Attēlošanai var importēt standarta formātu ontoloģijas failus.

4.1 VOWL valodas primitīvi

VOWL valoda ir izstrādāta “uz lietotājiem orientētai ontoloģiju reprezentācijai” [20]. Valodā ir minimāls primitīvu skaits (skat. 4.1 att., publikācija par VOWL).

Primitive	Application	Primitive	Application
	classes		datatypes, property labels
	properties		special classes/properties
	property directions	text number symbol	labels, cardinalities

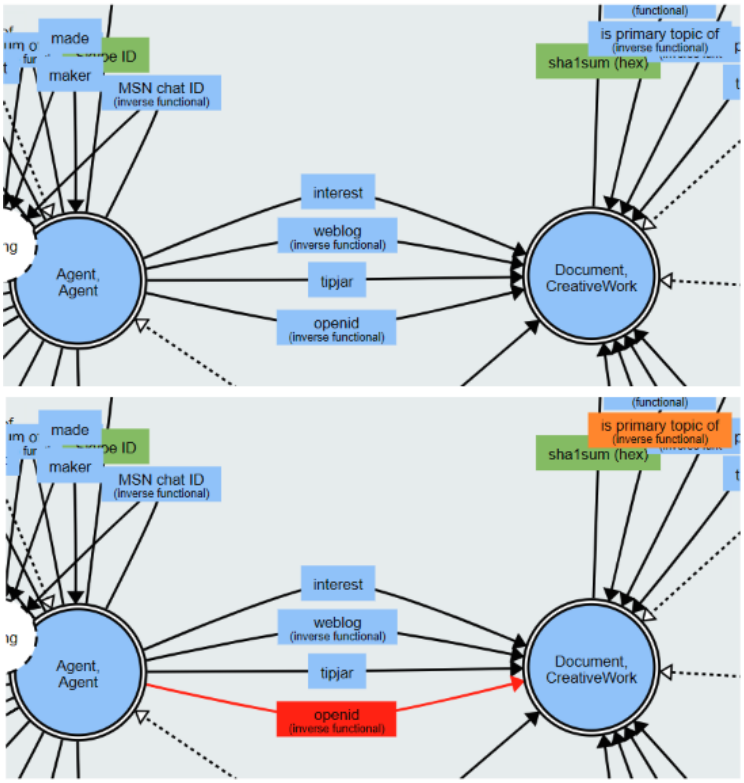
4.1 att. **VOWL grafiskās valodas primitīvi apkopotā veidā [20]**

Vienkāršā grafiskā notācija tiek kompensēta ar ekspansīvu krāsu pielietojumu (skat. 4.2. att.).

Name	Color	Application
General		classes, object properties, disjointness
External		external classes and properties
Deprecated		deprecated classes and properties
Datatype		datatypes, literals
Datatype property		datatype properties
Highlighting		circles, rectangles, lines, borders, arrows

4.2 att. **VOWL grafiskās valodas krāsu izmantojums konstrukciju diferencēšanai [20]**

Piemēram, iezīmēšana (*highlighting*) ļauj lietotājam dinamiski iezīmēt elementus un atkarībā no konteksta izgaismo arī citas konstrukcijas – piemēram, iezīmējot apakšpropertiju, tiek izgaismota tās virsklases propertija, vai iezīmējot vienu propertiju no funkcionālas un inversi-funkcionālas propertijas pāra, attiecīgi tiek izgaismota otra propertija (skat. 4.3. att).



4.3. att. **WebVOWL dinamiskās iezīmēšanas piemērs, autora ekrānuzņēmums rīkā iekļautajam ontoloģijas piemēram**

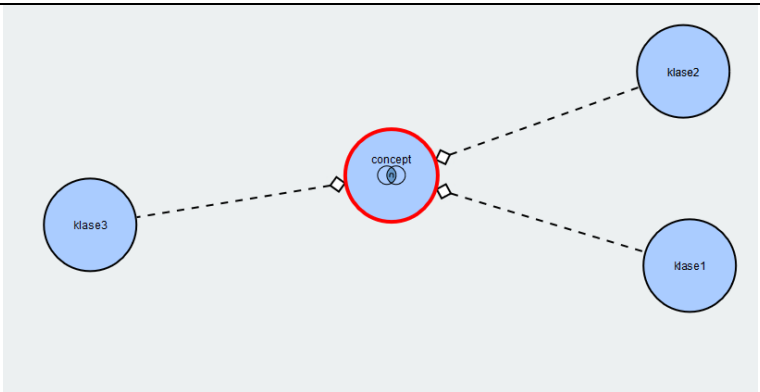
Piemērā iezīmēta ‘openid’ propertija no rīkā iekļautās piemēra FOAF ontoloģijas fragmenta.

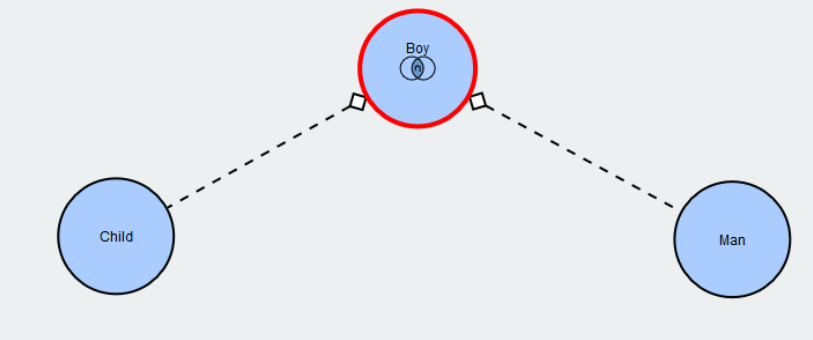
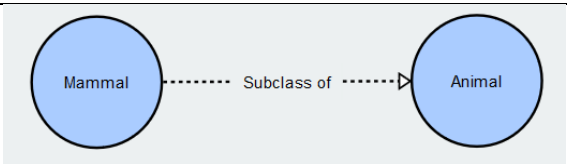
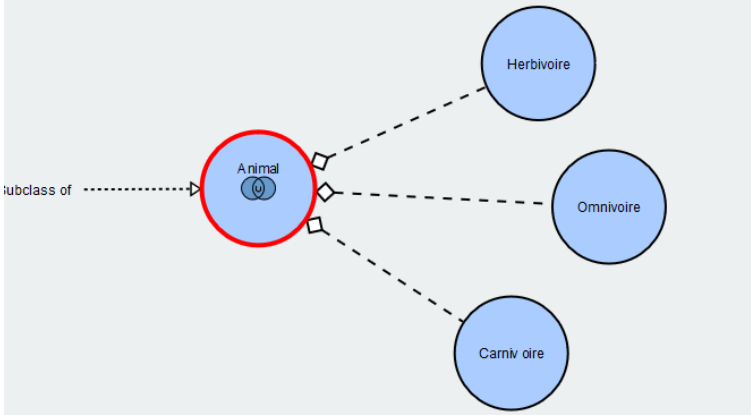
4.2 OWL2 konstrukcijas WebVOWL rīkā

Tiek apskatīts dažāda veida izteiksmju attēlojums WebVOWL tīmekļa rīkā. Vispirms apskatīti piemēri, kas apskatīti nodaļās 2.2 un 3.2. Visi piemēri ir autora izveidoti (piemēram, importējot, iepriekš rīkā Eddy izveidotās konstrukcijas, vai uzrakstot jaunus owl izteikumus); tabulā attēloti ekrānuzņēmumu fragmenti no WebVOWL redaktora UI.

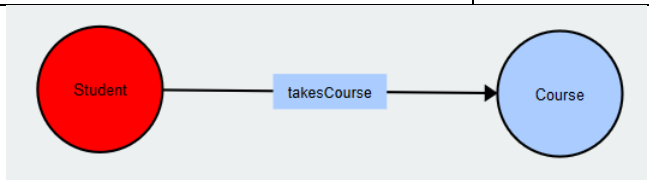
4.2. tabula

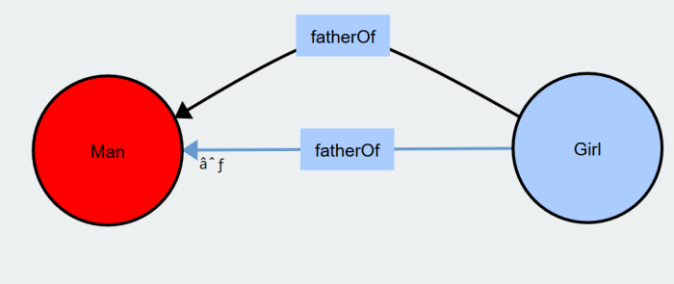
Izvēlētas OWL 2.0 konstrukcijas un to grafiskais attēlojums WebVOWL

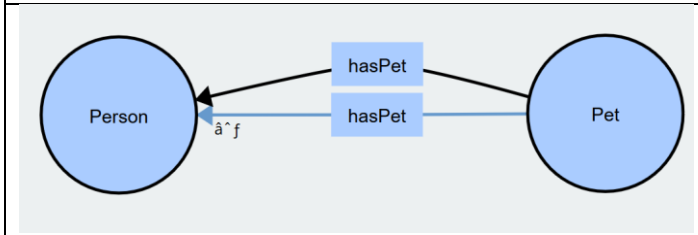
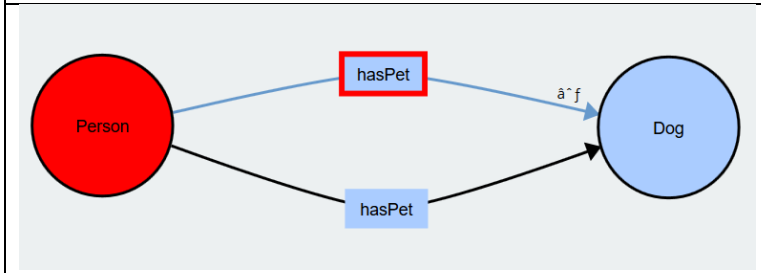
Nos.	*Konstrukcija WebVOWL rīkā*	*OWL sintakse, paskaidrojumi*
OWLGrEd un Eddy rīkos apskatītās konstrukcijas		
Ekvivalentas klases (I)(aks.)		EquivalentClasses(a:klase1 a:klase2)
Klašu škēlums I (izt.)		ObjectIntersectionOf(a:klase1 a:klase2 a:klase3))

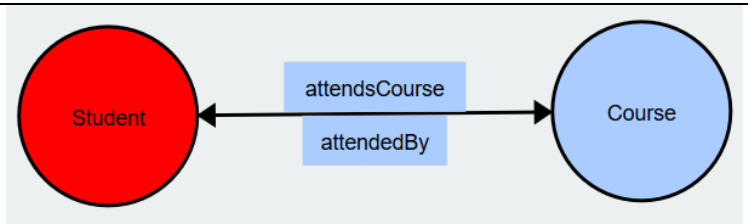
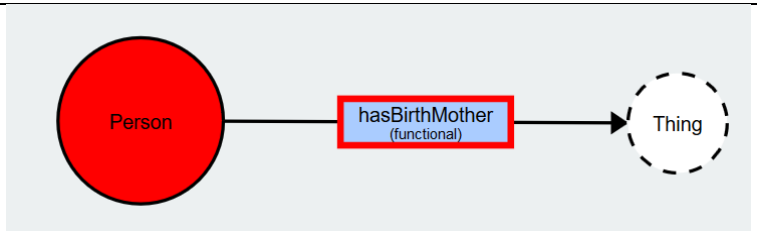
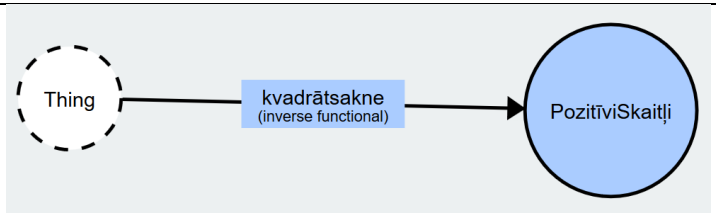
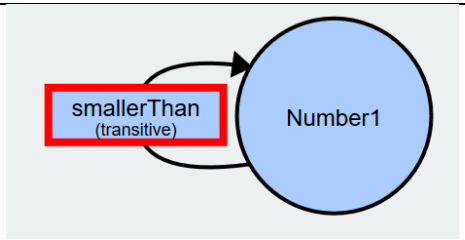
Klašu škēlums II (izt.)		Declaration(Class(a:Boy)) Declaration(Class(a:Child)) Declaration(Class(a:Man))[14] EquivalentClasses(a:Boy ObjectIntersectionOf(a:Child a:Man))
Apakšklase (I)		SubClassOf(a:Mammal a:Animal) Declaration(Class (Mammal)) Declaration(Class(Animal)) SubClassOf(Mammal Animal)
Apakšklase (III)		SubClassOf(Herbivoire Animal) SubClassOf(Carnivoire Animal) SubClassOf(Omnivoire Animal)

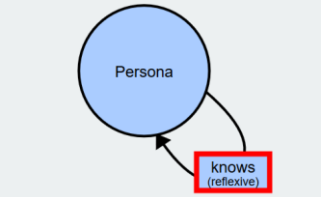
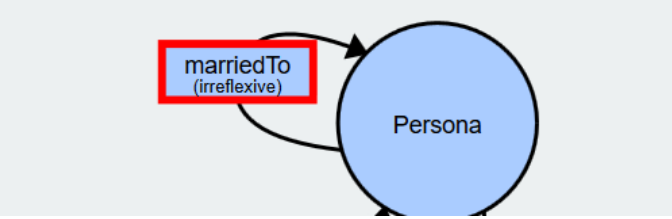
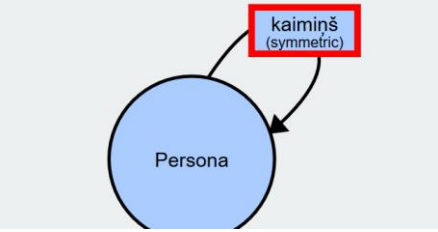
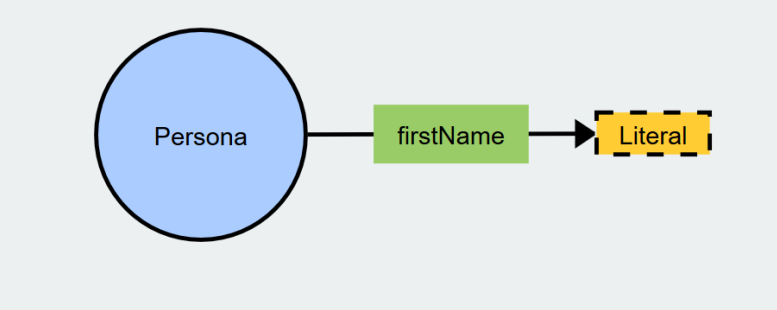
[14] Turpmākos piemēros elementārie apgalvojumi lielākoties tiek izlaisti

Nepārklājošas klases (I)		DisjointClasses(a:Boy a:Girl)
Klašu apvienojums (izt.) *N/A*	*N/A*	ObjectUnionOf(a:Carnivoire a:Herbivoire a:Omnivoire) *Anonīmā klase netiek attēlota*
Nepārklājošu klašu apvienojums (I) *N/A*		*Klases netiek attēlotas*
Klašu papildinājums *N/A*		ObjectComplementOf(a:NaturalNumber) *Klase netiek attēlota*
Klases indivīdu uzskaitījums *N/A*		ObjectOneOf(a:Lisa a:Stephen) *Klase netiek attēlota*
Domēns un vērtību apgabals		ObjectPropertyDomain(a:takesCourse a:Student) ObjectPropertyRange(a:takesCourse a:Course)
Eksistences ierobežojums *N/A*	*N/A*	ObjectSomeValuesFrom(a:fatherOf a:Man) *Klase netiek attēlota*
Universālais ierobežojums *N/A*		ObjectAllValuesFrom(a:hasPet a:Dog) *Klase netiek attēlota*
Konkrētas vērtības ierobežojums *N/A*		ObjectHasValue(a:hasChild a:Stewie) *Klase netiek attēlota*

<table>
<tr><td colspan="2">Ierobežojums pašam uz sevi[1]
N/A</td><td>N/A</td><td>ObjectHasSelf(a:likes)
Anonīmā klase netiek attēlota</td></tr>
<tr><td rowspan="3">Kardinalitātes ierobežojums (minimālā, maksimālā, tiešā)</td><td> 
</td><td></td><td>SubClassOf(< http://lumii.lv/ontologies/ontology.owl#Girl >
ObjectMinCardinality(2 <
http://lumii.lv/ontologies/ontology.owl#fatherOf > <h
ttp://lumii.lv/ontologies/ontology.owl#Man >))

Kļūda kardinalitātes attēlojumā</td></tr>
<tr><td> 
</td><td></td><td>ObjectMaxCardinality(2 a:hasPet a:Person)
Kļūda kardinalitātes attēlojumā</td></tr>
<tr><td> 
</td><td></td><td>ObjectExactCardinality(1 a:hasPet a:Dog)
Kļūda kardinalitātes attēlojumā</td></tr>
</table>

Inversas propertijas		InverseObjectProperties(a:hasFather a:fatherOf) *Tādas propertijas, kas atrodas uz viena savienotāja*
Funkcionāla propertija (I)		SubClassOf(a:Person ObjectMaxCardinality(1 a:hasBirthMother a:Person)) *Iezīme {func} pie predikāta*
Inversi funkcionāla propertija		InverseFunctionalObjectProperty(a:kvadrātsakne) *Iezīme {invf} pie predikāta (savienotāja)*
Tranzitīva propertija		TransitiveObjectProperty(< http://lumii.lv/ontologies/ontology.owl#smallerThan >) *Iezīme {tran} pie predikāta (savienotāja)*
Propertiju ķēde	*N/A*	*Rīks neatbalsta konstrukciju –nekas netiek attēlots*

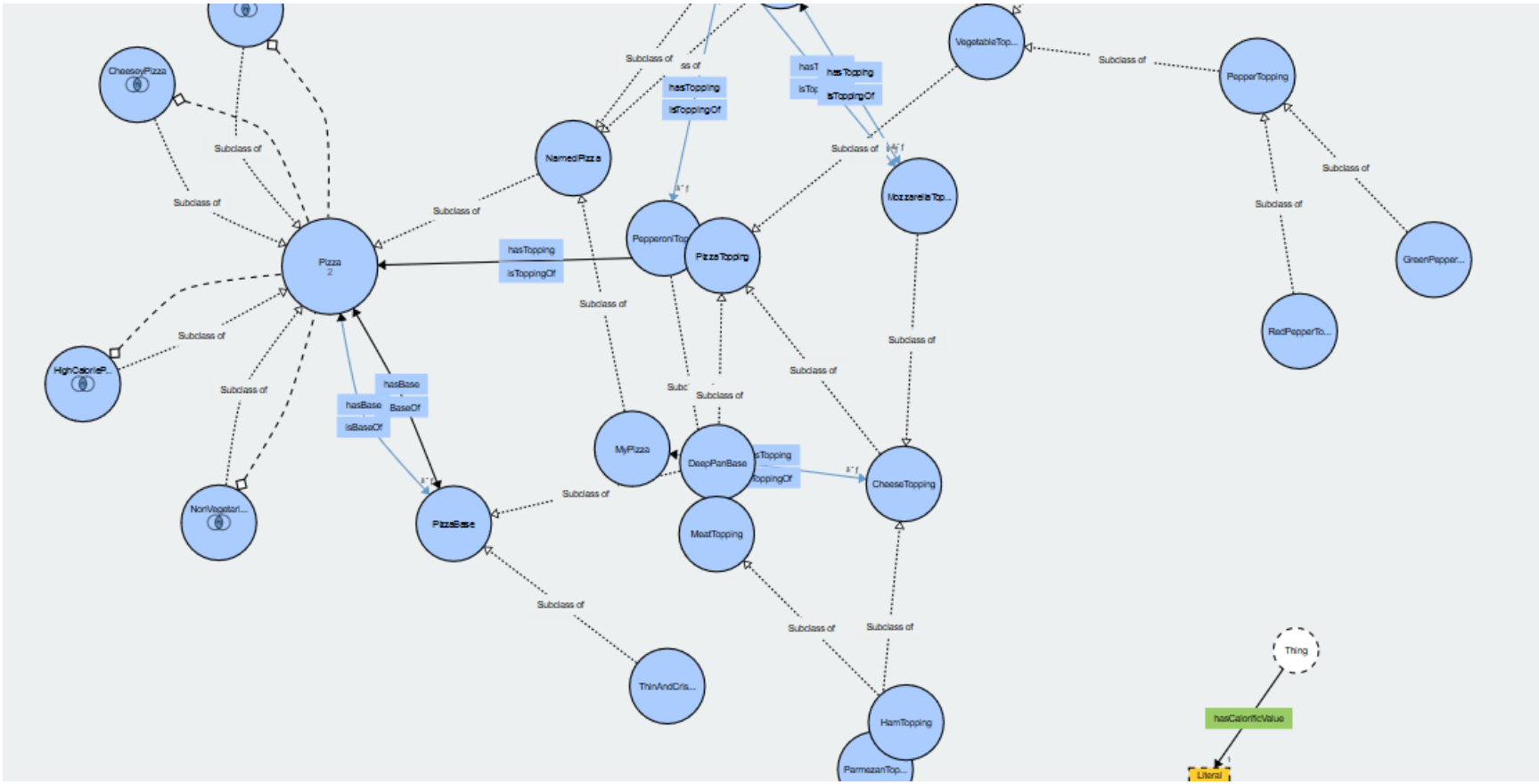
Refleksīva propertija		ReflexiveObjectProperty(knows), *Iezīme {refl} pie predikāta (savienotāja)*
Neatgriezeniska propertija (I)		IrreflexiveObjectProperty (a:marriedTo)
Simetriska propertija		SymmetricObjectProperty(a:irKaimiņš)
Datu propertijas ierobežojums		DataPropertyRange(a:firstName DatatypeRestriction(xsd:string xsd:minLength "3"^^xsd:string xsd:minLength "32"^^xsd:string)) *(!!)Trūkums attēlojumā – dažādie datu tipa ierobežojumi nekā netiek parādīti*

Nepareizi attēlotie piemēri tika pārbaudīti ar dažādiem formātiem (t.sk. OWL 1.0 sintaksi un uz vecākām versijām) un piemēriem, bet piemēru veikšanas brīdī rezultātus nedeva.

4.3 Ontoloģiju vizualizācijas rīkā WebVOWL

4.3.1 Picu ontoloģija

Attēlā 4.4 vizualizēts tas pats ontoloģijas dokuments, kas vizualizēts nodaļā 3.3.1.



4.4 att. **Picu ontoloģija rīkā WebVOWL (fragments)**

4.4 Secinājumi

Izmantotā VOWL notācija ir relatīvi vienkārša un viegli uztverama. WebVOWL paredzēts tikai ontoloģiju vizualizācijai. WebVOWL ļauj importēt OWL ontoloģiju, lai to dinamiski attēlotu, kas var būt noderīgi, piemēram, kad ir ļoti daudz līdzīgu līmeņu klašu, kas slikti pakļaujas hierarhiskas izkārtošanas modeļiem. Tomēr WebVOWL praktiski neatbalsta visas OWL 2.0 konstrukcijas kā vairākus komplekso klašu aksiomu veidus un datu tipu ierobežotāju pārklājumu.

5. KOPSAVILKUMS PAR EDDY, OWLGRED UN WEBVOWL RĪKIEM

Visiem apskatītajiem rīkiem tika nodrošināta ontoloģiju vizualizācija kādā līmenī, tomēr tie dažādos aspektos vērtējami atšķirīgi.

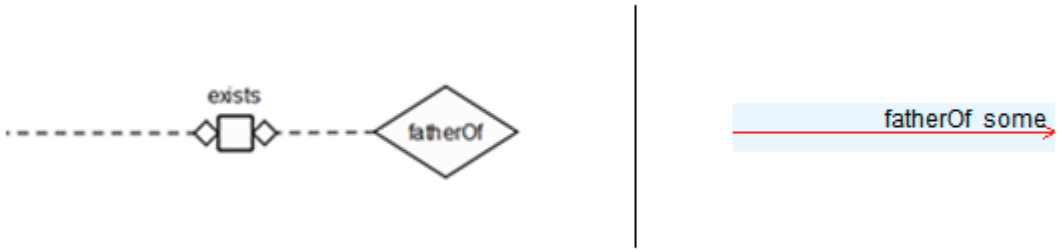
1. **Ontoloģiju grafisko elementu daudzums un vizualizācijas kompaktums**

No apskatītajiem konstrukciju piemēriem var novērot, kā izteikumu grafiskā forma salīdzināma starp dotajiem rīkiem. OWLGrEd rīkā iekļautā ontoloģija, kas tika apskatīta darbā, ir nedaudz vienkāršāka, nekā Eddy attēlotā (attiecīgi 26 un 33 klases), tāpēc salīdzināšanai Eddy iekļautā ontoloģija tika importēta attēlošanai OWLGrEd. Jāpiebilst, ka arī izprotot visu konstrukciju attēlošanu, autors uzskata, ka dažādu savienotāju līniju izmantošana mēdz būt mulsinoša, un sarežģīja modelēšanu, cenšoties pašam modelēt lielākus ontoloģiju piemērus.

Autors uzskata, ka dažādu līniju tipi – raustīta līnija, punktu līnija, atšķirīgs līnijas biezums, un/vai dažādas to krāsas ir vislabāk uztveramais notācijas veids. OWLGrEd tīmekļa versijā redzami jaunieviesumi, kas neparādās darbā apskatītajā versijā, un šādu pieeju turpina (papildus iekļaujot dinamiskus elementus – līdzīgi WebVOWL – kā savienotāju iezīmēšana un caurspīdīgums).

Ģeometriskie ķermeņi – neskaitot līnijas (kā arī Eddy vairāk savienotāju elementu, jo izteikumi grafiski sadalīti ‘smalkāk’) OWLGrEd – 44, Eddy (predikātu un konstruktoru mezgli)– 62.

Tas izskaidrojams ar Mančesteras sintakses izmantojumu izteiksmju pierakstīšanai. No OWL konstrukciju apskata nodaļām sekojoši var atrast semantiski ekvivalentus, bet grafiski atšķirīgus (skat. 5.1 att.) izteikumus.



5.1 att. **Grafiskās notācijas atšķirības propertijas un tās ierobežojuma norādīšanai**

Attēlā redzami fragmenti no iepriekšējās nodaļās apskatītajiem piemēriem (autora ilustrācija). Tomēr jāpiebilst, ka OWLGrEd nepieciešams arī atsevišķs savienotājs ‘fatherOf’ predikāta definēšanai, kas nav uzrādīts.

Novērotie piemēri papildus apkopoti 5.1 tabulā.

5.1. tabula

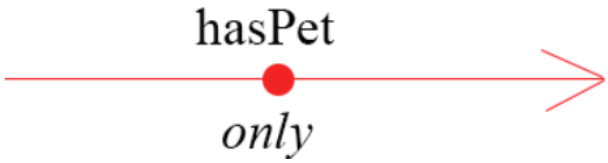
OWL 2.0 konstrukciju grafiskās un Mančesteras sintakses izmantojuma atšķirības OWLGrEd un Eddy rīkos

Izteikums		*Eddy*	*OWLGrEd*	*Mančesteras sintakses atslēgvārds*
Ierobežotāji	Eksistences ierobežojums	Konstruktora mezgls "exists"	Iezīme "some" piesaistīta ierobežotājam	*some*
	Universālais ierobežojums	Konstruktora mezgls "forall"	Iezīme "only" piesaistīta ierobežotājam	*only*
	Konkrētas vērtības ierobežojums	Konstruktora mezgls "oneOf" + konstruktora mezgls "exists"	Iezīme "value" piesaistīta ierobežotājam	*value*
Loģiskās operācijas	Klašu papildinājums	Konstruktora mezgls "not"	Iezīme <<complementOf>> piesaistīta savienotājam	*not*
	Klašu šķēlums	Konstruktora mezgls "and"	Izteiksme kā atribūts klasei formā = klase$_1$ and klase$_2$ ... and klase$_n$	*and*
	Klašu apvienojums	Konstruktora mezgls "or"	Izteiksme kā atribūts klasei formā = klase$_1$ or klase$_2$ ... or klase$_n$	*or*

Autors uzskata, ka 'liela' (atsevišķa ģeometriska ķermeņa, kā '*first-class citizen*'), Eddy predikāta vai konstruktora mezgla līmeņa konstrukcijas, kurai vajadzētu 'sacensties' ar galvenajām UML konstrukcijām, ieviešana būtu lieka.

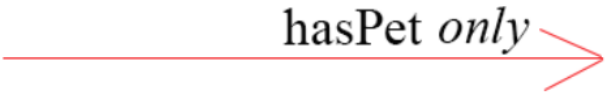
Tomēr iespējams varētu apsvērt neliela ķermeņa ieviešanu – piem. 'līnijas viduspunkts' (vai piem., opciju tā attēlošanai visā ontoloģijā), kuru attēlot 'pa virsu' citām konstrukcijām, un uz

kura attēlot šo Mančesteras sintakses atslēgvārdu – piemērs ierobežotājam (5.2 att., autora piemēra ilustrācija):



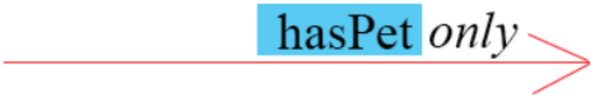
5.2 att. **Grafiskās notācijas izmaiņas, dizaina prototips**

Vai, pat tikai nomainot fontu Mančesteras/loģiskajiem atslēgvārdiem, var iegūt augstāku grafisko izteiksmību (5.3. att., autora ilustrācija) :



5.3 att. **Atslēgvārdu fonta izmaiņas, dizaina prototips**

Iezīmes fona iekrāsošana vizuāli to varētu padarīt kā atsevišķu konstrukciju (kā 'kaste' VOWL, skat. 5.4 att).



5.4 att. **Predikāta dizaina izmaiņas, dizaina prototips**

2. OWL2 (pilns) atbalsts

VOWL ir daudz lielisku iezīmju, kas diemžēl pilnībā neparādās apskatītajā dinamiskajā tīmekļa implementācijā, jo importējot ontoloģijas, kas pilnībā attēlojās OWLGrEd un Eddy rīkos, ontoloģijas dažkārt netika attēlotas nemaz. Darbā apskatīto konstrukciju kopsavilkums:
Graphol – attēlo praktiski visas populārākās aksiomas par propertijām (nodaļa 1.3.2, izņemot anotācijas propertijas), visas aksiomas par klasēm (nodaļa 1.3.3), praktiski visas klašu aksiomas par propertijām (nodaļa 1.3.4, izņemot atslēgas propertijas).
OWLGrEd – attēlo pilnīgi visas populārākās aksiomas par propertijām (nodaļa 1.3.2), visas aksiomas par klasēm (nodaļa 1.3.3), praktiski visas klašu aksiomas par propertijām (nodaļa 1.3.4, izņemot propertiju ķēde).
WebVOWL – attēlo gandrīz visas populārākās aksiomas par propertijām (nodaļa 1.3.2, izņemot anotācijas), korekti attēlo tikai 4 no 8 aksiomām par klasēm (nodaļa 1.3.3), neattēlo nevienu no klašu aksiomām par propertijām (nodaļa 1.3.4).

3. Rediģēšana, importēšana un eksportēšana

Lai gan acīmredzami, ir vērts vēlreiz piebilst, ka WebVOWL paredzēts tikai vizualizācijai un rediģēšana nav iespējama (ir pieejama tīmekļa redaktora testa versija, kura tika izmēģināta, bet šobrīd atbalsta daudz mazāk konstrukciju, kā WebVOWL, kas aprakstīts iepriekšējā punktā).
Eddy – var importēt tikai Graphol ontoloģijas, eksportēt OWL ontoloģijas noklusētā formātā
OWLGrEd – var importēt visas korektas OWL ontoloģijas, eksportēt visos OWL atbalstītajos formātos, kā arī kā SVG attēlu
WebVOWL – var importēt korektas OWL ontoloģijas no faila vai pēc IRI, var eksportēt kā SVG attēlu vai JSON failu, uz kādu ontoloģija tiek konvertēta attēlošanai.

4. Citi vispārīgi secinājumi

WebVOWL priekšrocība ir ontoloģijas dinamiska attēlošana bez programmatūras uz klienta datora. Dinamiskā klašu izvietošana labi piemērota ‘tīklveida’ ontoloģijām, kurām nav raksturīga klasiska, hierarhiska struktūra, vai ļoti lielam skaitam neatkarīgu klašu (skat. 2. pielikumu), kas gan nav īpaši aktuāli, jo šādās ontoloģijās vieglāk ‘orientēties’ būs ar rīkiem kā Protégé [21].

REZULTĀTI

Darbā izvirzītie mērķi tika sasniegti. Vispirms apskatīta tīmekļa ontoloģiju valoda OWL 2.0 un attiecīgi konstrukcijas no teorētiskā viedokļa, sasaistot tās ar matemātikas un loģikas terminiem, un apskatot dažādas situācijas un piemērus, kur šīs konstrukcijas varētu tikt izmantotas.

OWL konstrukcijas sekojoši tika apskatītas no praktiskā viedokļa – modelējot tās ar redaktora rīkiem OWLGrEd vai Eddy, vai rakstot atbilstošos OWL izteikumus, un importējot piemēra ontoloģijas grafiskajos rīkos.

Veikts OWLGrEd, Eddy un WebVOWL un attiecīgi paplašinātas UML sintakses, Graphol sintakses, un vizuālas ontoloģiju sintakses VOWL apskats, un secinājumi par tiem.

SECINĀJUMI

Bakalaura darba izstrādes gaitā tika secināts, ka OWL 2.0 ontoloģiju valodai nav noteikta stingra standarta tās grafiskā attēlojuma notācija. Tātad grafiskās ontoloģiju attēlošanas rīkiem galvenais ir atbalstīt ontoloģiju valodas īpašības. Sekojoši dažādu valstu (t.sk. bieži vien universitāšu) speciālisti izstrādājuši savus piedāvājumus grafiskajai ontoloģiju notācijai, un atbalsta tos attiecīgi tam izstrādātajos rīkos.

Darba rezultātos nav jaunu konstrukciju ieviesumu rīkā OWLGrEd, bet sniegti secinājumi un vērtējumi, kas varētu būt vērtīgi rīku iespēju salīdzināšanā un pilnveidošanā.
Darbā apskatītās konstrukcijas varētu arī palīdzēt apgūt svarīgākās OWL konstrukcijas, vai palīdzēt apgūt šo ontoloģiju redaktoru un vizualizatoru notāciju.

Var secināt, ka ontoloģiju rīkus nevar viennozīmīgi novērtēt, lai gan tie visi ir radīti ar vienotu mērķi. Izmantojamais rīks var būt ērtāks vai neērtāks atkarībā no izmantojamās ontoloģijas, tāpēc darbs varētu palīdzēt arī rīka izvēlē.

IZMANTOTIE AVOTI UN LITERATŪRA

[1] Grigoris Antoniou, Paul Groth, Frank van Harmelen, Rinke Hoekstra, *A Semantic Web Primer (Third edition).* Cambridge, Massachusetts: The MIT Press, 2012

[2] Dean Allemang, Jim Hendler, *Semantic Web for the Working Ontologist (Second edition).* Waltham, USA: Morgan Kauffman, 2011

[3] Bārzdiņš, J., Bārzdiņš, G., Čerāns, K., Liepiņš, R., & Sproģis, A. (2010, June). OWLGrEd: a UML style graphical notation and editor for OWL 2. In *Proceedings of 7th International Workshop "OWL: Experience and Directions* (Vol. 34).

[4] *IBM Community,* IBM Redbooks [Tiešsaiste]. Pieejams: https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_the_semantic_technologies?lang=en

[5] Console, M., Lembo, D., Santarelli, V. and Savo, D.F., 2012. *The Graphol language for ontology specification* [Tiešsaiste]. Pieejams: http://www.dis.uniroma1.it/~graphol/documentation/GrapholLVPrel.pdf

[6] Lembo, D., Pantaleone, D., Santarelli, V. and Savo, D.F., 2016, April. Easy OWL Drawing with the Graphol Visual Ontology Language. In *KR* (pp. 573-576).

[7] Bojārs, U. Kursa „Semantiskais tīmeklis" lekciju materiāls *OWL valoda, ontoloģijas un spriešana (reasoning)* (lekciju slaidi)

[8] OWLGrEd http://owlgred.lumii.lv/

[9] *A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3* [Tiešsaiste]. Pieejams: http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf

[10] Matthew Horridge *Protégé OWL Tutorial* (2011, 24.mar.) [Tiešsaiste]. Pieejams: http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/

[11] *OWL 2 Web Ontology Language Document Overview (Second Edition)* [Tiešsaiste]. Pieejams: https://www.w3.org/TR/owl2-overview/

[12] Alan Rector, Guus Schreiber, *Qualified cardinality restrictions (QCRs): Constraining the number of values of a particular type for a property* [Tiešsaiste]. Pieejams: https://www.w3.org/2001/sw/BestPractices/OEP/QCR/

[13] *OWL 2 Web Ontology Language Primer (Second Edition)* (2012, 11.dec) [Tiešsaiste]. Pieejams: https://www.w3.org/TR/2012/REC-owl2-primer-20121211/

[14] V.Detlovs, K.Podnieks *Introduction to Mathematical Logic* [Tiešsaiste]. Pieejams: https://dspace.lu.lv/dspace/bitstream/handle/7/2777/Detlovs_Podnieks_Math_Logic.pdf?sequence=1&isAllowed=y
[15] *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)* (2012, dec) [Tiešsaiste]. Pieejams: https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/
[16] *OWL 2 Web Ontology Language New Features and Rationale (Second Edition)* [Tiešsaiste]. Pieejams: https://www.w3.org/TR/owl2-new-features/#F12:_Punning
[17] Lembo, D., Pantaleone, D., Santarelli, V. and Savo, D.F., 2016, July. *Eddy: A Graphical Editor for OWL 2 Ontologies*. In *IJCAI* (pp. 4252-4253).
[18] Marco Console, Domenico Lembo, Valerio Santarelli, Domenico Fabio Savo *Graphol: a graphical ontology language* June 7, 2014 http://www.dis.uniroma1.it/~graphol/documentation/syntax.pdf
[19] Console, M., Lembo, D., Santarelli, V. and Savo, D.F., 2012. *The Graphol language for ontology specification*.
[20] Krzysztof Janowicz, Stefan Schlobach *Visualizing Ontologies with VOWL* [Tiešsaiste]. Pieejams: http://www.semantic-web-journal.net/system/files/swj1114.pdf
[21] Protégé http://protege.stanford.edu
[22] Peter Mika, Hans Akkermans *Towards a new synthesis of ontology technology and knowledge management* [Tiešsaiste]. Pieejams: https://www.researchgate.net/figure/Ontology-functionality-in-applications-ranges-from-ontology-usage-for-Communication-to_fig5_231890373
[23] *WebVOWL: Web-based Visualization of Ontologies* [Tiešsaiste]. Pieejams: http://vowl.visualdataweb.org/

PIELIKUMI

1. Pielikums. OWL ontoloģijas pieraksta piemērs no W3C

Ontoloģijas valodā OWL 2.0 piemērs no W3C [13] (fragments).

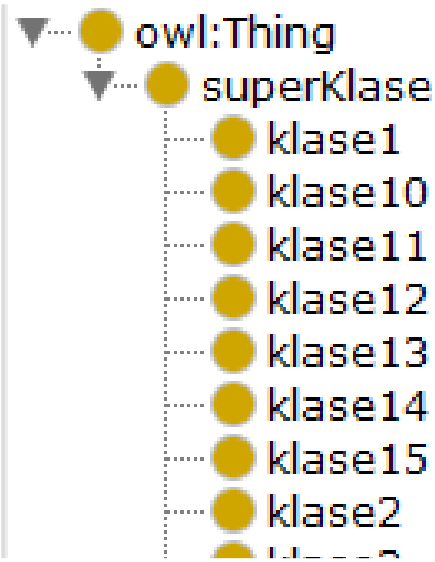
```
Prefix(:=<http://example.com/owl/families/>)
 Prefix(otherOnt:=<http://example.org/otherOntologies/families/>)
 Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
 Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
 Ontology(<http://example.com/owl/families>
   Import( <http://example.org/otherOntologies/families.owl> )

   Declaration( NamedIndividual( :John ) )
   Declaration( NamedIndividual( :Mary ) )
   Declaration( NamedIndividual( :Jim ) )
   Declaration( NamedIndividual( :James ) )
   Declaration( NamedIndividual( :Jack ) )
   Declaration( NamedIndividual( :Bill ) )
   Declaration( NamedIndividual( :Susan ) )
   Declaration( Class( :Person ) )
   AnnotationAssertion( rdfs:comment :Person "Represents the set of all people." )
   Declaration( Class( :Woman ) )
   Declaration( Class( :Parent ) )
   Declaration( Class( :Father ) )
   Declaration( Class( :Mother ) )
   Declaration( Class( :SocialRole ) )
   Declaration( Class( :Man ) )
   Declaration( Class( :Teenager ) )
   Declaration( Class( :ChildlessPerson ) )
   Declaration( Class( :Human ) )
   Declaration( Class( :Female ) )
   Declaration( Class( :HappyPerson ) )
   Declaration( Class( :JohnsChildren ) )
   Declaration( Class( :NarcisticPerson ) )
   Declaration( Class( :MyBirthdayGuests ) )
   Declaration( Class( :Dead ) )
   Declaration( Class( :Orphan ) )
   Declaration( Class( :Adult ) )
   Declaration( Class( :YoungChild ) )
   Declaration( ObjectProperty( :hasWife ) )
   Declaration( ObjectProperty( :hasChild ) )
   Declaration( ObjectProperty( :hasDaughter ) )
   Declaration( ObjectProperty( :loves ) )
   Declaration( ObjectProperty( :hasSpouse ) )
   Declaration( ObjectProperty( :hasGrandparent ) )
   Declaration( ObjectProperty( :hasParent ) )
   Declaration( ObjectProperty( :hasBrother ) )
   Declaration( ObjectProperty( :hasUncle ) )
   Declaration( ObjectProperty( :hasSon ) )
   Declaration( ObjectProperty( :hasAncestor ) )
   Declaration( ObjectProperty( :hasHusband ) )
   Declaration( DataProperty( :hasAge ) )
   Declaration( DataProperty( :hasSSN ) )
   Declaration( Datatype( :personAge ) )
   Declaration( Datatype( :minorAge ) )
```

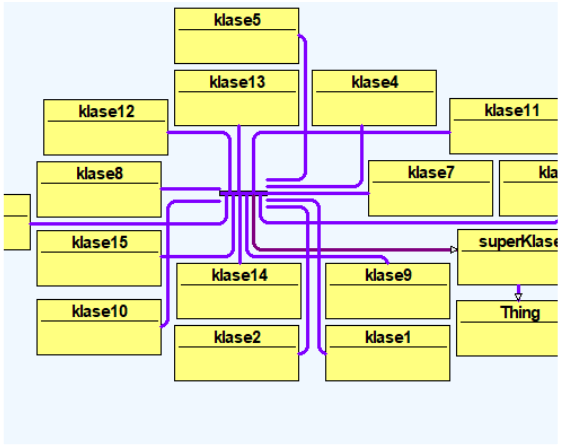
2. Pielikums. Ontoloģijas vizualizācijas salīdzinājums OWLGrEd, WebVOWL un Eddy

Pirmkārt redzama vienkāršas ontoloģijas hierarhija Protégé UI, un tālāk atbilstoši attiecīgajām sintaksēm automātisks OWLGrEd attēlojums, automātisks WebVOWL attēlojums un manuāls Eddy attēlojums (vizualizācijas attēlotas samazinātā mērogā).

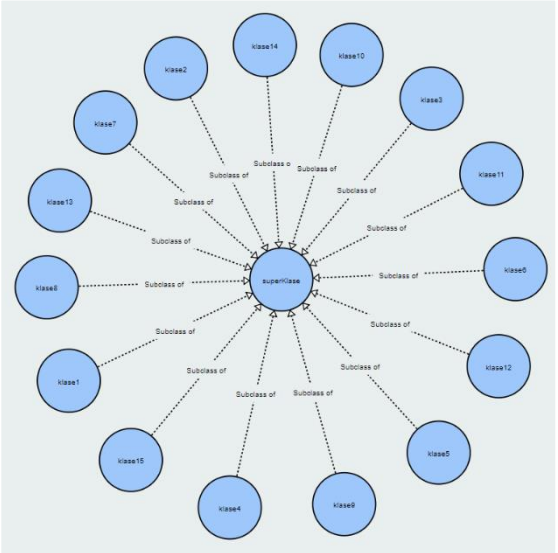
Protégé



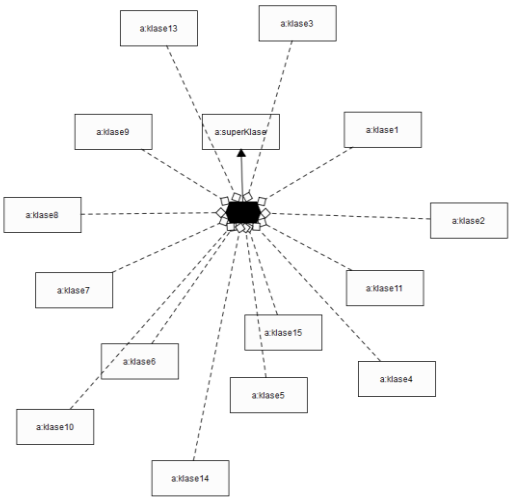
OWLGrEd



WebVOWL



Eddy



Bakalaura darbs „Grafiskās ontoloģiju attēlošanas konstrukcijas un rīki” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ Nauris Gruduls

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītājs: profesors Dr. dat. Kārlis Čerāns _____ 28.05.2018.

Recenzents: docents Maksims Kravcevs

Darbs iesniegts Datorikas fakultātē 28.05.2018.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.06.2018. prot. Nr. ___.

Komisijas sekretārs: _____