OWL - Web Ontology Language

Hansi Keijonen

Seminaariraportti
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 3. maaliskuuta 2013

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Hansi Keijonen

Työn nimi — Arbetets titel — Title

OWL - Web Ontology Language

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaariraportti	3. maaliskuuta 2013	18

Tiivistelmä — Referat — Abstract

Tiivistelmä.

Avainsanat — Nyckelord — Keywords
avainsana 1, avainsana 2, avainsana 3

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Semanttinen web

Suurin osa tämän päivän webin sisällöstä on tarkoitettu ihmisten luettavaksi sekä tulkittavaksi. Kone pystyy tulkitsemaan esim. html-tiedoston ja esittämään dokumentin siinä määritellyllä tavalla. Ongelma on, että kone ei *ymmärrä* dokumentin sisällön merkitystä, semantiikkaa [2].Se, että kone ei ymmärrä dokumenttien semanttisia merkityksiä rajoittaa esimerkiksi haut internetissä olevista dokumenteista yksinkertaiseksi hakusanojen etsimiseksi. Sen sijaan jos hakukoneet ymmärtäisivät asioiden merkityksen ja niiden välillä vallitsevat yhteydet, olisi hakukoneiden hakutulokset tarkempia ja sisältäisivät mahdollisesti laajennettuja hakuja alkuperäisen asian ympäriltä [2]. On siis tarve olla menetelmä käsitteiden luomiseen, käsitteiden ominaisuuksien kuvaamiseen sekä käsitteiden välisten suhteiden kuvaamiseen [2]. Tim Berners-Lee, James Hendler ja Ora Lassila toteavat artikkelissaan "Semantic web", että "semanttinen web ei ole erillinen web vaan laajennos tämänhetkiseen webiin, jossa informaatiolle on annettu hyvin muotoiltu merkitys mahdollistaen koneiden ja ihmisten paremman yhteistyön."web of documents -> web of data dataa voi parsia manuaalisesti tai koneellisesti

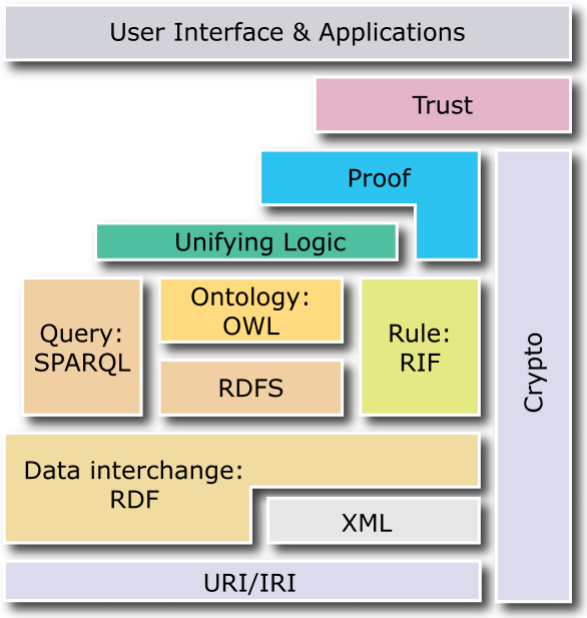
2 Teknologiat ja kielet jotka mahdollistavat OWL:n

W3c tarjoaa semanttisen webin toteuttamiseen standardit teknologioista ja kielistä. Kuvassa 1 on semanttisen webin teknologiapino sekä ajatuskonsepteja semanttisen webin toteuttamiseen. Osa teknologioista on todellisuutta ja käytössä, osa vasta ideatasolla. Jokainen kerros käyttää alemman kerroksen palveluita. Seuraavissa kappaleissa käyn läpi kaavion teknologioita ja kieliä alhaalta ylöspäin kohti OWL:ää. Jokaisesta seuraavissa kappaleissa esitetystä teknologiasta käsitellään tarkemmin ne konstruktiot, jotka ovat olennaisia ja käytössä myös OWL:ssä.

2.1 URI ja XML

Semanttisessa webissä luokkia, ilmentymiä, ominaisuuksia ja ominaisuuksien arvoja kutsutaan resursseiksi. Jotta sekaannusta jo määritettyjen resurssien sekä uusien määritysten kanssa ei syntyisi, identifioidaan kaikki resurssit (pl. ominaisuuksien literaaliarvot) yksilöllisesti URI(Unified Resource Locator):lla. URI:n avulla voidaan viitata mihin tahansa määritettyyn resurssiin. Useinmiten URIna toimii perinteinen URL(Unified Resourse Locator)-osoite [2]. IRI (Internationalized Resource Identifier) on ainoastaan merkistölaajennos URI:in.

Semanttisen webin datan kuvaukset toteutetaan useimmiten XML-tiedostoina. XML-kieltä voidaan käyttää monimutkaisen rakenteisen tiedon esittämiseen ja tarjoaa näin standardoidun mallin tiedon vaihtamiseen prosessoijien välillä.

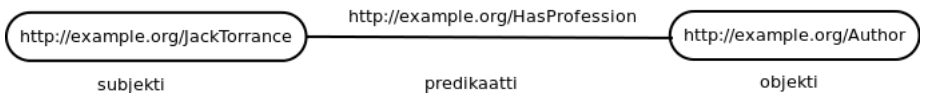


Kuva 1: Semanttisen webin toteutukseen tarvittavat teknologia, kielet ja konseptit.

Tärkeä XML:n ominaisuus on nimiavaruudet, jotka mahdollistavat resurssien identifioinnin URI:en avulla. Tästä kuitenkin enemmän seuraavissa kappaleissa.

2.2 Tiedon esittäminen RDF-triploilla

Resource Description Framework RDF on kieli webissä olevien resurssien kuvaamiseen. RDF perustuu asioiden identifiointiin URI:lla ja näiden asioiden kuvaamiseen ominaisuuksilla ja ominaisuuksien arvoilla. Tämä mahdollistaa yksinkertaisten lausumien esittämisen verkkoina, joissa resurssit ja ominaisuudet ovat soluja ja ominaisuudet verkon kaaria [4]. Kuvassa 2 on havainnollistettu asiaa.



Kuva 2: RDF-tripla joka kuvaa yksinkertaisen lausuman. Jokainen triplan solmu ja kaari on identifioitu URI:lla.

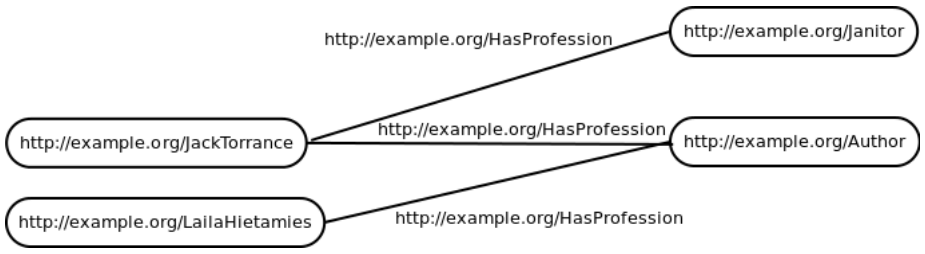
Kuvassa 2 on havainnollistettu yksinkertainen *RDF-tripla*. Triplan subjekti, predikaatti ja objekti kertovat, että "Jack Torrance on ammatiltaan kirjailija". Jokainen solu ja kaari on esimerkissä identifioitu URI:lla. Objekti

voi olla myös literaali, jolloin sitä ei identifioida erikseen, mutta formaatti määritellään ****NIIN VITTU MILLÄ?*****

Yleisin tapa esittää tripla on XML-notaatio. Myös muut tavat ovat mahdollisia, kuten esimerkiksi *JSON* ja *turtle*. Alla on esitetty kuvassa 2 triplan XML-notaatio:

```
<rdf:Description rdf:about="http://example.org/JackTorrance">
    <rel:HasProfession rdf:resource="http://example.org/Author"/>
</rdf:Description>
```

Esimerkistä näkee selvästi, kuinka subjekti, predikaatti ja objekti ovat identifioitu URI:lla. Jack Torrance on toiselta ammatiltaan talonmies ja myös Laila Hietamies on ammatiltaan kirjailija. Kun myös nämä triplat kuvataan samaan kaavioon, alkaa pieni mutta informatiivinen verkkkuva ja selitys triploista ja URI:sta? Kyllä, tähän se sopisi kokoavana elementtinä. o syntyä, kuva 3. Tästä verkosta voisimme tehdä hakuja, kuten "ketkä ovat kirjailijoita?".



Kuva 3: RDF-triplat muodostavat verkon.

RDF:n peruskonstruktioita ovat: description, resource, about ***selitä nää*****

RDF tarjoaa siis vain melko alkeellisen tavan esittää lausumia, jotka muodostavat haut mahdollistavan verkon. RDF toteuttaa myös joukon muita ominaisuuksia, kuten *säiliöitä* (container) tiedon säilömiseen sekä *kokoomatietorakenteita* (collections) asioiden listaamiseen [4]. Näitä käsitellään myöhemmin niiltä osin kuin ne ovat relevantteja OWL-ontologioiden muodostamisessa.

2.3 Alkeelliset ontologiat RDF Schemalla

Semantiikkaa voidaan webissä ilmaista ontologioilla. Tietojenkäsittelytieteessä ontologialla tarkoitetaan dokumenttia, jossa kerrrotaan asioiden välisistä yhteyksistä [2]. ***vajaa läppä!! kuuluuko edes tähän???!!*******

RDF:llä ilmaistut ominaisuudet voidaan ajatella resurssien attribuuteiksi samassa mielessä kuin perinteiset attribuutti-arvo -parit [3]. Vaikka RDF:llä

voidaan kuvata resursseja, se ei tarjoa keinoja kuvata ominaisuuksia tai ominaisuuksien välisiä suhteita. Tämä on mahdollista RDF:n *sanastonkuvauslaajennoksella* RDF Schemalla. RDF Schemalla on mahdollista määritellä luokkia ja ominaisuuksia joita voidaan käyttää luokkien, ominaisuuksien ja resurssien kuvaamiseen [3]. RDF Schema on RDF:n semanttinen laajennos.

Seuraavassa esitellään RDF Scheman niitä peruskonstruktioita, joita myös OWL käyttää määrittelyssään lähes sellaisenaan. *RDFS Schema määrittelee myös suurehkon joukon muita konstruktioita, mutta niiden esittely OWL:n selostamisen kannalta on tässä tarpeetonta.*

2.3.1 Luokat ja ilmentymät

Resurssit voidaan jaotella luokkiin. Luokkien jäseniä nimitetään luokan ilmentymiksi [3]. Luokat ovat itsessään resursseja. Useimmiten luokat identifioidaan RDF:n URI:lla ja niitä voidaan kuvata RDF:n ominaisuuksilla (property). Luokka ilmaistaan rdfs:Class tagilla, rdf:type ominaisuudella voidaan ilmaista, että resurssi kuuluu luokkaan [3]. Esimerkiksi kustannusosakeyhtiö "WSOY"on itse luokka ja luokan "Kustantamo"ilmentymä:

```
Esim:WSOY rdf:type rdfs:Class
Esim:WSOY rdf:type esim:Kustantamo
```

2.3.2 Luokan aliluokka

rdfs:subClassOf ominaisuudella voidaan ilmaista, että jonkin luokan ilmentymät ovat myös jonkin toisen luokan ilmentymiä [3]. Esimerkiksi luokka "Kustantamo"on luokan "Liikeyritys"aliluokka:

```
esim:Kustantamo rdfs:subClassOf esim:Liikeyritys
```

2.3.3 Ominaisuudet ja periytetyt ominaisuudet

Resursien välisiä suhteita kuvataan ominaisuuksilla. Esimerkiksi "Matilla on veli Teppo". Voidaan ajatella, että resurssien "Matti"ja "Teppo"välinen suhde on ominaisuus "Veljeys". Ominaisuus "Veljeys"voi olla ominaisuuden "Perhesuhde"aliluokka tai paremminkin aliominaisuus. Tällaisia konstruktioita kuvataan RDF Schemassa termillä subPropertyOf:

```
esim:Veljeys rdfs:subPropertyOf esim:Perhesuhde
```

subPropertyOf ilmaisee määritelmällisesti, että resurssit joiden välistä suhdetta kuvataan jollain suhteella, voidaan kuvata myös tämän suhteen ylisuhteella, josta ko. suhde on periytynyt [3].

2.3.4 Rajoitukset ominaisuuksien sovellusalueissa ja arvoissa

RDF Schemassa on mahdollista antaa ominaisuuksille rajoituksia sen suhteen, että minkä luokkien välistä suhdetta ominaisuus kuvaa. Ensinnäkin voidaan rajoittaa suhteen sovellusaluetta (domain). Suhteen "Työskentelee"sovellusalue voidaan rajata koskemaan ainoastaan luokkaa "Työntekijä". Samaten suhteen saamat arvot (range) voidaan rajata olemaan ainoastaan luokan "Yritys"tai sen aliluokkien ilmentymiä.

```
esim:Työskentelee rdfs:domain esim:Työntekijä
esim:Työskentelee rdfs:range esim:Yritys
```

2.3.5 RDF Schema esimerkki

Myös RDF Scheman varsinainen notaatio on toteutettu XML:llä. Alla on lyhyt esimerkki RDF Schemalla toteutetusta ontologiasta.:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

<rdfs:Class rdf:ID="liikeyritys" />

<rdfs:Class rdf:ID="kustantamo">
  <rdfs:subClassOf rdf:resource="#liikeyritys"/>
</rdfs:Class>

</rdf:RDF>
```

Esimerkissä on yksinkertainen ontologia, joka kertoo, että "kustantamo"ja "liikeyritys"ovat luokkia ja että "kustantamo"on myös luokan "liikeyritys"aliluokka. RDF Schema -ontologiat ovat RDF-tiedostoja samoin kuin OWL-ontologiatkin. Tiedoston alussa olevat nimiavaruusmäärittelyt varmistavat, että rdf- ja rdfs-alkuisilla tageilla ympäröidyt elementit todellakin ovat rdf- ja rdfs- elementtejä. Nimiavaruudet käsitellään takemmin OWL:n esittelyn yhteydessä.

3 Kehittyneitä ontologioita OWL:llä

Tälä hetkellä kehittynein tapa ilmaista ontologioita on OWL-ontologiat. Lyhenne OWL tulee sanoista Web Ontology Language. Vaikka määritelmässä on sana language, kieli, on OWL-ontologiat ymmärettävä ennemminkin sanastoina, joita on kuvattu RDF-kielellä. Eräs tapa hahmottaa RDF-triplojen ja OWL-ontologioiden välinen ero on verrata niitä perinteiseen relaatiotietokantaan. RDF-triplat on tapa tallettaa tietoa olioiden ominaisuuksista samalla

tavalla kuin relaatiotietokannan taulun riveillä tallennetaan rakenteista tietoa tietokantaolioista. Jokaista riviä relaatiotietokannassa yksilöi yksilöivä avain kun taas RDF-triploissa avaimen virkaa hoitaa URI. Relaatiotietokannoissa tietokantaolion attribuuttien suhteita ilmaistaan taulurakenteilla ja tietokantaolioiden suhteita toisiin tietokantaolioihin ilmaistaan viitteillä taulujen välillä. Vastaavasti voidaan ajatella, että OWL-ontologiat kuvaavat ja jäsentävät samalla tavalla RDF-triploilla ilmaistua tietoa: ontologialla voidaan määritellä monipuolisesti luokkien, ilmentymien ja suhteiden ominaisuuksia. RDF on kuin kieli jolla ilmaistaan lausumia asioista kun taas OWL on sanasto, jonka avulla lausumien merkitykset ymmärretään.

OWL tarjoaa ontologioiden määrittelyyn [1]

- *hyvin määritellyn syntaksin* , jotta ontologiat olisivat koneluettavissa
- *hyvin määritellyn semantiikan*, jolla voi ilmaista merkityksiä tarkasti ja konsistentisti
- *tuen koneelliselle päättelylle*, jotta esim. ontologioiden eheys voidaan tarkastaa automaattisesti
- *riittävästi ilmaisuvoimaa* ilmaisemaan kaikki tarvittavat merkitykset
- *miellyttävän ilmaisutavan*, jotta työskentely olisi sujuvaa

Ideaalisesti OWL on RDF:n ja RDF Scheman laajennos [1]. OWL käyttää RDF:n luokkia ja suhteita lisäten niihin omia laajennoksiaan. RDF Schemassa on joitain hyvin vahvoja konstruktioita, kuten rfd:Class (kaikkien luokkien yliluokka) sekä rdf:Property (kaikkien suhteiden yliluokka). Näiden primitiivien ilmaisuvoima yhdistettynä OWL:n tarjoamaan laajennoksiin on ristiriidassa sen tavoitteen kanssa, että ontologiat olisivat koneellisesti pääteltävissä. Tämä tasapainotila mielessäpitäen on määritelty kolme OWL:n alikieltä sen perusteella, painotetaanko ilmaisuvoimaa vai koneellista päättelyä [1].

3.1 OWL:n kolme alikieltä

W3C:n Web Ontology Working Group on määritellyt OWL:lle kolme alikieltä, joiden on takoitus toteuttaa eri aspektit (ilmaisuvoima, koneellinen päättely), joita ontologioiden kuvaamiskieleltä vaaditaan [5]. Alikielet ovat ilmaisuvoiman mukaisesti kasvavassa järjestyksessä:

- *OWL Lite* tarjoaa ainoastaan yksinkertaisen luokitteluhierarkian ja yksinkertaiset rajoitteet[5]. Kardinaalisuusrajoitteet ovat ainoastaan muotoa 0 ta 1. Työkalutuen tarjoaminen on OWL Litelle helpompaa kuin ilmaisuvoimaisemmille versioille [5].

- *OWL DL* on tarkoitettu niille käyttäjille, jotka haluavat mahdollisimman hyvän ilmaisukyvyn siten, että ontologia on koneellisesti pääteltävissä (kaikki päätelmät tehtävissä järjellisessä ajassa) [5]. OWL DL tarjoaa kaikki kielen konstruktiot, mutta niitä voi käyttää tietyin rajoituksin, esimerkiksi luokka voi olla monen luokan aliluokka mutta ei voi olla samalla luokan ilmentymä. DL tulee sanoista Description Logics, deskriptiivinen logiikka, joka on tieteenala, joka tutkii logiikkaa ja on OWL:n perusta [5].
- *OWL Full* on tarkoitettu niille käyttäjille, jotka haluavat maksimaalisen ilmaisukyvyn välittämättä siitä, onko ontologiat enää koneellisesti pääteltävissä [5]. Toisin kuin OWL DL:ssä, luokka voi olla kokoelma yksilöitä (instansseja) samalla kuin luokka itsessään on jonkin luokan yksilö. OWL Full mahdollistaa jo olemassa olevien ontologioiden laajentamisen. On epätodennäköistä, että mikään ohjelmisto pystyy täydellisesti päättelemään OWL Full ontologioita [5].

3.2 OWL-ontologian rakenne

3.2.1 Nimiavaruudet

XML-dokumentissa tulee määritellä nimiavaruudet (namespace). Nimiavaruuksien avulla voidaan ratkaista mm. samannimisten elementtien aiheuttamia tulkintaongelmia sekä kertoa lukijalle (koneelle tai ihmiselle) konteksti, jonka mukaan elementtien tageja tulee tulkita. OWL-ontologiassa nimiavaruudet määritellään rdf:RDF -kahvojen sisään. *Kaikki OWL-ontologiat ovat RDF-dokumentteja* [6]. Alla olevassa esimerkissä on eräs mahdollinen nimiavaruusmäärittely .

```
<rdf:RDF
    xmlns:esim ="http://www.esimerkki.com/esim#"
    xmlns:owl ="http://www.w3.org/2002/07/owl#"
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

Esimerkin nimiavaruusmäärittelyissä on määritelty nimiavaruus niille tageille, jotka käyttävät etuliitettä esim:. Nimiavaruudet on määritelty myös owl:-, rdf:- ja rdfs:-etuliiteille kertomaan, että näillä etuliitteillä varustetut tagit edustavat OWL:n, RDF:n ja RDF Scemann termistöä. OWL-ontologia on riippuvainen myös XMLSchema-datatyypeistä (xsd:), joten myös niiden nimiavaruus tulee määrittää.

3.2.2 Otsikkotiedot

Owl-ontologian otsikkotiedoissa voidaan kertoa yleisiä asioita kuten versiotietoa, kommentteja, ontologian nimi sekä tärkeänä ominaisuutena importata

muita ontologioita. Otsikkotiedot määritellään owl:Ontology-elementiksi:

```
<owl:Ontology>
   <rdfs:label>Esimerkkiontologia</rdfs:label>
   <rdfs:comment>Esimerkin voimaa</rdfs:comment>
   <owl:priorVersion>
       rdf:resource="http://www.esimerkki.com/vanhempi#"
   </owl:priorVersion>
   <owl:imports>rdf:resource="http://purl.org/dc/elements/1.1"</owl:imports>
</owl:Ontology>
```

Esimerkkiotsikossa perustietojen lisäksi importataan Dublin Core - sanasto [1] ontologiamme käyttöön.

3.2.3 Yksinkertaiset luokat ja aliluokat

OWL:ssä kaikki yksilöt ovat luokan owl:Thing jäseniä ja kaikki käyttäjän määrittämät luokat owl:Thingin aliluokkia [6]. Myös owl:Nothing on määritelty(MIX VITUSCH?). Luokkamäärittelyt tapahtuvat hyvin samaan tapaan kuin RDF Schemassa, ainoastaan tagin nimi on owl:Class. Aliluokkanotaatio vastaa suoraan RDF Scheman konstruktiota:

```
<owl:Class  rdf:ID="Auto">
<owl:Class  rdf:ID="Ajoneuvo">

<owl:Class rdf:ID="Auto">
    <rdfs:subClassOf rdf:resource="Ajoneuvo"/>
</owl:Class>
```

Esimerkissä on määritetty kaksi luokkaa "Auto"ja "Ajoneuvo"sekä määritelty edellinen jälkimmäisen aliluokaksi.

OWL:ssä on myös mahdollista määritellä luokka siten, että luokan jäsenet voivat kuulua ko. luokkaan mutta eivät missään tapauksessa implisiittisesti määritettyyn toiseen luokkaan:

```
<owl:Class ref:about="Ajoneuco">
    <owl:disjointWith rdf:resource="Elain"
</owl>
```

Luokan "Ajoneuvo"ja sen mahdollisten aliluokkien jäsenet eivät voi siis kuulua samaan aikaan luokkaan "Kissa"tai sen mahdollisiin aliluokkiin. Voidaan siis päätellä, että ilmentymä "Lada"ei voi olla luokan "Kissa"jäsen jos "Kissa"on määritetty luokan "Eläin"aliluokaksi.

[1] Dublin Core on nnnnn

3.2.4 Luokan yksilöt

Luokan jäseniä kutsutaan yksilöiksi, eli luokan ilmentymiksi [6]. Jonkun ilmentymän kuuluminen luokkaan ilmaistaan samalla tavalla kuin RDF Schemassa rdf:type konstruktiolla. Ensiksi määritellään kuitenkin instanssi owl:Thing-tagilla:

```
<owl:Thing rdf:ID="Lada"/>

<owl:Thing ref:about="Lada">
    <rdf:type= rdf:resource="Autonvalmistaja"/>
</owl:Thing>
```

Esimerkissä määritettiin "Autonvalmistaja-luokan ilmentymä "Lada". Esimerkissä viitataan oletetusti jo aiemmin määritettyyn "Autonvalmistaja-luokkaan [6].

3.2.5 Luokkaominaisuus ja datatyyppiominaisuus

Ominaisuudet jaetaan OWL:ssä luokkaominaisuuksiksi (Object Property) ja datatyyppiominaisuuksiksi (Data Type Property) sen perusteella liittääkö ominaisuus yhteen kaksi luokan ilmentymää vai luokan ilmentymän ja RDF-literaalin tai XML Schema -datatyypin [6].

3.2.6 Luokkaominaisuus

OWL:ssä luokkaominaisuudet määritellään kuten mitkä tahansa luokat, mutta käyttäen ObjectProperty -konstruktiota. Määrittelyssä kerrotaan myös minkä luokan ilmentymiin ominaisuus on sovellettavissa sekä mitä arvoja (luokkia) ominaisuus voi saada. Nämä rajoitteet ilmaistaan RDF Schemasta tutuilla rdfs:domain- ja rdfs:range-elementeillä:

```
<owl:ObjectProperty rdf:about="Merkki"/>

<owl:ObjectProperty rdf:ID="Merkki">
    <rdfs:domain rdf:resource="Auto"/>
    <rdfs:range rdf:resource="Autonvalmistaja"/>
</owl:ObjectProperty>
```

Esimerkissä on määritelty ominaisuus "Merkki", jolla voi kuvata ainoastaan luokan "Auto"ilmentymiä ja joka voi ainoastaan saada arvokseen ainoastaan luokan "Autonvalmistaja"ilmentymiä.

3.2.7 Datatyyppiominaisuus

Datatyyppiominaisuus liittää yhteen luokan ilmentymän ja datan arvon: RDF-literaalin tai XML Schema -datatyypin. Myös datatyyppiominaisuuk-

sien määrittelyssä kerrotaan minkä luokan ilmentymiin ominaisuus on sovellettavissa sekä minkä tyyppisiä arvoja ominaisuus voi saada [6]. Rajoitteet tehdään samalla tavalla kuin luokkaominaisuuden määrittelyssä:

```
<owl:DataTypeProperty rdf:about="Valmistusvuosi"/>

<owl:DataTypeProperty rdf:ID="Valmistusvuosi">
    <rdfs:domain rdf:resource="Auto"/>
    <rdfs:range  rdf:resource="&xsd;gYear"/>
<owl:dataTypeProperty>
```

Esimerkissä määritetään ominaisuus "Valmistusvuosi", jolla voidaan kuvata luokan "Auto"ilmentymiä ja joka voi saada arvokseen XML Schema -standardissa määritetyn gYear-tyypin arvon. XML-Schema -standardi määrittelee kymmeniä eri datatyyppejä XML-dokumenteissa käytettäväksi. Kun käytetään määritettyjä datatyyppejä, XML-parseri pystyy tarkistamaan, onko annetut arvot sallittuja ja esitystapa oikea.

3.2.8 Aliominaisuudet

Myös ominaisuuksille voidaan määritellä aliominaisuuksia samaan tapaan kuin luokille voidaan määrittää aliluokkia. Aliominaisuus toteutetaan subPropertyOf-konstruktiolla, joka on toteutettu jo RDF Schemassa [6]. Alla olevassa esimerkissä lisätään ominaisuuteen "Valmistusvuosi"määritys, että se on ominaisuuden "AutonKuvailija"aliominaisuus:

```
<owl:Class rdf:ID="AutonKuvailija">

<owl:DataTypeProperty rdf:ID="Valmistusvuosi">
    <rdfs:subPropertyOf rdf:resource="AutonKuvailija">
    . . .
<owl:dataTypeProperty>
```

3.2.9 Kardinaalisuusrajoitteet

OWL antaa mahdollisuuden määrätä kardinaalisuusrajoitteita ominaisuuksien arvoille kun ominaisuutta sovelletaan tietyssä kontekstissa [6]. Voidaan esimerkiksi määrätä, että luokka "Ajoneuvo"on joukko asioita, joilla on aina vähintään kaksi arvoa "Renkaat-ominaisuudessa. Toisin sanoen, määritellään anonyymi aliluokka luokalle "Ajoneuvo"[6]. Alla oleva esimerkki selventää asiaa:

```
<owl:Class  rdf:ID="Ajoneuvo">
    <rdfs:subClassOf>
       <owl:Restriction>
           <owl:onProperty rdf:resource="Renkaat"
```

```
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2
            </owl:minCardinality>
        </owl:Restriction>
    </owl:subClassOf>
<owl:Class/>
```

Anonyymi aliluokka määrätään normaalilla subClassOf-konstruktiolla. Aliluokalle määritetään rajoite owl:Restriction-tagilla ja kardinaalisuusrajoite tässä tapauksessa minCardinality-elementillä. Kardinaalisuus määritetään koskemaan ominaisuutta "Renkaat"onProperty-elementillä. Kardinaalisuusrajoitteita voi määrätä myös maxCardinality-elementillä, joka määrää suhteen yläkardinaliteetin, sekä Cardinality-elementillä, joka määrää tarkan arvon mikä on luokan esiintymän ja ominaisuuden suhde. Voitaisiin esimerkiksi määrittää luokka "Moottoripyörä"siten, että seillä voisi olla tasan kaksi "Renkaat"ominaisuutta.

3.2.10 Ominaisuuden transitiivisuus, symmetrisyys, funktionaalisuus ja inversio

Ominaisuuden voi tarvittaessa määrittää transitiiviseksi, symmetriseksi tai funktionaaliseksi [6]. Näin tekemällä voidaan monipuolistaa ontologioista tehtäviä implisiittisiä päättelyitä. Transitiivinen ominaisuus voidaan määrittää seuraavasti ominaisuudelle P ja ilmentymille x,y ja z [6]:

P(x,y) ja P(y,z) -> P(x,z)

Esimerkiksi omnaisuus rdfs:subClassOf on transitiivinen. Itse määritetyn ominaisuuden voi määrittää transitiiviseksi määrittämällä ominaisuus luokan "TransitiveProperty"jäseneksi:

```
. . .
<rdf:type rdf:resource="&owl;TransitiveProperty"
. . .
```

ESIMERKKI?

Symmetrinen ominaisuus voidaan määrittää ominaisuudelle P ja ilmentymille x,y ja z [6]:

P(x,y) joss P(y,x)

Vastaavasti oman ominaisuuden voi määrittää symmetriseksi määrittämällä se luokan "SymmetricProperty"jäseneksi:

```
. . .
<rdf:type rdf:resource="&owl;SymetricProperty"
. . .
```

Esimerkiksi ominaisuus "Sisar"voitaisin määritellä symmetriseksi. Jos x on y:n sisar niin implisiittisesti voidaan päätellä, että y on x:n sisar.

Funktionaalinen ominaisuus voiadaan määrittää ominaisuudelle P, ilmentymille x,y ja z [6]:

P(x,y) ja P(x,z) -> y = z

Edellisten esimerkkien tapaan ominaisuuden voi määrittää funktionaaliseksi:

```
. . .
<rdf:type rdf:resource="&owl;FunctionalProperty"
. . .
```

Funktionaalinen ominaisuus voisi olla esimerkiksi opiskelijanumero. Jokainen opiskelijaluokan ilmentymä, joka olisi myös saman oppilaitoksen jäsen, voitaisiin identifioida opiskelijanumero-ominaisuuden perusteella, koska on olemassa ainoastaan yksi ominaisuuden ilmentymä, jolla on tietty opiskelijanumero.

INVERSIO JA FUNKTIONAALINEN INVERSIO???

3.2.11 Arvorajoitteet luokkaominaisuuksissa

OWL mahdollistaa rajoittaa ilmentymiä, joita voidaan antaa ominaisuuden arvoiksi luokkaominaisuuksia määritettäessä [6]. Tällä tavoin voidaan rajata aliluokkia näiden rajoitusten avulla. Esimerkiksi on mahdollista määrätä, että luokan "Auto"aliluokka siten, että ominaisuuden "Omistaja"arvona voi olla vain luokan "Henkilo"jäsen:

```
<owl:Class rdf:ID="Auto">
    <rdfs:subClassOf>
       <owl:Restriction>
           <owl:onProperty rdf:resource="Omistaja"/>
           <owl:allValuesFrom rdf:resource="Henkilo"/>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Edellisessä esimerkissä määritettiin, että auton omistaja on aina "Henkilo-luokan jäsen. OWL mahdollistaa myös hieman kevyemmän rajoitteen, jossa määrätään, että ominaisuuden jonkun arvon tulee olla määrätyn luokan jäsen [6]. Voidaan määrittää, että autojen omistajista ainakin yksi on luokan "Henkilo"jasen:

```
. . .
           <owl:onProperty rdf:resource="Omistaja"/>
           <owl:someValuesFrom rdf:resource="Henkilo"/>
. . .
```

Rajoite toteutetaan owl:someValuesFrom-tagilla.

On myös mahdollista määrätä ominaisuuden arvoksi täsmällisesti joku olemassa oleva resurssi [6]. Tällöin aliluokkamäärittelyyn lisätään owl:hasValue -elementti:

```
. . .
            <owl:onProperty rdf:resource="Omistaja"/>
            <owl:hasValue rdf:"Hansi"/>
. . .
```

3.2.12 Kompleksiset luokat

Kompleksiset luokat on erittäin vahva ja ilmaisuvoimainen OWL:n konstruktio [6]. Konstruktio tarjoaa mahdollisuuden määrittää luokkalausekkeita, joissa käytetään joukko-opista tuttuja operaattoreita *yhdistetä*, *leikkausta* ja *komplementtia*. Esimerkiksi voimme määrittää luokan "Lada-auto", joka on luokan "Auto"ja sellaisten asioiden, jotka ovat jollain tapaa "Lada", leikkaus:

```
<owl:Class rdf:ID="Lada-auto">
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Auto"/>
        <owl:Class rdf:about="Lada"/>
    </owl:intersectionOf>
</owl:Class>
```

Esimerkissä leikkaus-operaattori on ilmaistu owl:intersectionOf -elementillä. rdf:parseType="Collection"ilmoittaa parserille, että tämän elementin sisältämät elementit tulee tulkita lueteltuna kokoelmana.

Vastaavasti voidaan määrittää yhdisteluokka käyttämällä elementtiä owl:unionOf [6]. Esimerkkinä voidaan määrittää luokka "Maansiirtokoneet", jonka jäsenet ovat ainakin yhden luetellun luokan jäsen:

```
<owl:Class rdf:ID="Maansiirtokone"/>
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Kauhakuormaaja"/>
        <owl:Class rdf:about="Dumpperi"/>
    </owl:intersectionOf>
</owl:Class>
```

Esimerkissä luokkaan "Maansiirtokone"kuuluu ilmentymät, jotka ovat joko kauhakuormaajia, dumppereita tai peräti molempia.

Komplementtiluokka määritellään elementillä owl:complementOf [6]. Komplementtiluokan jäseniä määrittää se, että ne eivät missään tapauksessa ole komplementiksi ilmoitetun luokan jäseniä. Komplementtiluokalla on helppo esimerkiksi määrittää luokka, "Manuaalivaihteisto", joka määritelmällisesti on luokan "Automaattivaihteisto"komplementti:

```
<owl:Class ref:about="Automaattivaihteisto"/>

<owl:Class ref:about="Manuaalivaihteisto">
    <owl::complementOf rdf:resource="Automaattivaihteisto">
</owl:Class>
```

Esimerkkimäärityksessä luokkaan "Manuaalivaihteisto"kuuluvat vain ne ilmentymät, jotka eivät kuulu luokkaan "Automaattivaihteisto".

3.2.13 Ontologioiden yhdistäminen

Jotta ontologioiden luominen olisi mielekästä, tulee olla kyky yhdistää ontologioita toisiinsa jotta saadaan aikaan uusia laajempia ontologioita [6]. Ontologian otsikkotiedoissa voidaan import-lasuseella tuoda jonkun toisen jo olemassa olevan ontologian määritykset oman ontologian käyttöön. OWL tarjoaa myös elementtejä, joilla voidaan käyttää hyväksi jo olemassa olevia määrityksiä omien määrityksien tekemisessä [6].

owl:equivalentClass -elementillä voidaan oma luokkamääritys määrätä vastaamaan jotain olemassaolevaa määritystä:

```
<owl:Class rdf:ID="Auto">
    <owl:equivalentClass rdf:resource="&kulkuneuvot;Auto"/>
</owl:Class>
```

Esimerkissä siis oletetaan, että otsikkotiedoissa on import-määritys ontologialle "kulkuneuvot". Oma määritys luokalle "Auto"siis asetetaan vastaamaan tuodun ontologian määritystä luokalle "Auto".

OWL mahdollistaa myös määrittää yksittäisen ilmentymän vastaamaan toista ilmentymää [6]. Tämä tehdään sameAs-konstruktiolla:

```
<owl:Class rdf:ID="HansinSuosikkiAuto">
    <owl:sameAs rdf:resource="Lada1200A"/>
</owl:Class>
```

Tällä konstruktiolla ei ole suurtakaan käyttöä. Ainoa, mitä tässä ilmaistaan on, että Hansin suosikkiauto on Ladan klassikkomalli. Käänteinen ominaisuus sameAs-konstruktiolle on differentFrom [6]. Sen käyttö on samaan tapaan suoraviivaista eikä se trivialiteettien lisäksi määritä mitään oleellista.

Käyttökelpoisempi konsruktio on *AllDifferent*, jolla voidaan määrittää kokoelma ilmentymiä olemaan erillisiä toistensa suhteen [6]:

```
<owl:AllDifferent>
    <owl:distinctMembers rdf:parseType="Collection">
        <Henkilo rdf:about="Matti"/>
        <Henkilo rdf:about="Teppo"/>
        <Henkilo rdf:about="Seppo"/>
    </owl:distinctMembers>
</owl:AllDifferent>
```

Esimerkki kertoo, että jokainen Henkilon ilmentymä viittaa eri ilmentymään.

3.3 OWL 2

OWL 2 on laajennos OWL-kieleen, jolle on annettu W3C-suositus vuonna 2009 [**?**]. OWL 2:n ominaisuudet ovat suurilta osin samat kuin OWL 1:ssä, osa ainoastaan eri nimellä esitettynä . Myös OWL 2:n ytimen muodostaa XML/RDF ja mikä tärkeintä, OWL 2 on taaksepäin yhteensopiva OWL 1:n kanssa. Jokainen OWL 1 -ontologia on siis kelvollinen OWL 2 -ontologia.

OWL 2 lisää pääasiassa syntaktista sokeria OWL 1:n päälle. Se tarjoaa kuitenkin myös muutamia uusia toiminnallisuuksia [**?**]:

- *avaimet*, joka mahdollistaa sen, että ominaisuuksia (tai ominaisuuksien joukkoa) voidaan käyttää avaimena tunnistamaan luokan ilmentymää.
- *ominaisuusketjut*, joka mahdollistaa esimerkiksi määrittää ominaisuuden "Isovanhempi"kahden "Vanhempi-ominaisuuden ketjuksi
- *uusia datatyyppejä*, jotka mahdollistavat esimerkiki ominaisuuksille annettavien numeeristen arvojen rajoittamisen jollekin välille.
- *kvalifioituja kardinaalisuusrajoitteita*
- *asymetriset, refleksiiviset ja poissulkevat ominaisuudet.* Asymmetrisyys on vastakohta OWL 1:n symmetrisyydelle, reflektiivinen ominaisuus viittaa aina itseensä ja poissulkevat ominaisuudet eivät voi olla voimassa samaan aikaan.
- *parannellut kommentointimahdollisuudet*

Lähteet

[1] Antoniou, Grigoris ja Harmelen, Frank Van: *Web Ontology Language: OWL*. Handbook on ontologies, 2(September):91–110, 2009. http://www.springerlink.com/index/10.1007/978-3-540-92673-3.

[2] Berners-Lee, Tim, Hendler, James ja Lassila, Ora: *The Semantic Web*. Scientific American, 284(5):34–43, 2001. http://www.nature.com/doifinder/10.1038/scientificamerican0501-34.

[3] Brickley, Dan ja Guha, R V: *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation, 2009(10 February 2004), 2004. http://www.w3.org/TR/rdf-schema/.

[4] Manola, Frank ja Miller, Eric: *RDF Primer*. W3C Recommendation, 10(February):1–107, 2004. http://www.w3.org/TR/rdf-primer/.

[5] McGuinness, Deborah L ja Van Harmelen, Frank: *OWL Web Ontology Language Overview*. W3C recommendation, 10(February):1–22, 2004. http://www.w3.org/TR/owl-features/.

[6] Smith, Michael K, Welty, Chris ja McGuinness, Deborah L: *OWL Web Ontology Language Guide*. W3C Recommendation, 10(February):1–46, 2004. http://www.w3.org/TR/owl-guide/.