Jenni Myllynen

Semanttinen web ja sukututkimus

Tietotekniikan
pro gradu -tutkielma
29. maaliskuuta 2007

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Jenni Myllynen

Yhteystiedot: jenni.myllynen@gmail.com

Työn nimi: Semanttinen web ja sukututkimus

Title in English: Semantic web and genealogy

Työ: Tietotekniikan pro gradu -tutkielma

Sivumäärä: 73+16

Tiivistelmä: Semanttinen web on nykyisen WWW:n laajennos, jonka tarkoituksena on tulevaisuudessa olla ihmisten lisäksi myös tietokoneiden käytettävissä. Tämän mahdollistaa tiedon esittäminen sellaisessa muodossa, että koneetkin pystyvät ymmärtämään tiedon merkityksiä. Semanttisen webin tekniikoita voidaan soveltaa useilla eri aloilla, ja tämän tutkielman tarkoituksena on tutkia soveltamismahdollisuuksia sukututkimuksessa. Empiirisessä osuudessa on tutkittu olemassa olevan sukutiedon muuntamista semanttiseen muotoon sekä semanttisten tekniikoiden hyödyntämistä sukutiedon käsittelyssä. Tutkimustulokset osoittavat käytettyjen tekniikoiden ja menetelmien mahdollistavan sukutiedon esityksen ja käsittelyn semanttisin keinoin.

English abstract: The Semantic Web is an extension of the current Web that is intended to be used not only by humans, but also by computers in the future. This will be enabled by representing the knowledge in such a way that also the machines are able to understand the meaning of the knowledge. The Semantic Web technologies can be applied to several domains and the objective of this thesis is to explore the application in genealogy. Conversion of the existing genealogical data into the semantic form and the processing of the converted data using the semantic techniques have been studied in the empirical part of the thesis. The results show that the technologies and methods being used enable the representation and processing of the genealogical data in a semantic way.

Avainsanat: semanttinen web, sukututkimus, RDF, ontologia, tietämysjärjestelmä, Protégé, GEDCOM

Keywords: semantic web, genealogy, RDF, ontology, knowledge-based system, Protégé, GEDCOM

Sisältö

1 Johdanto

Nykymuotoisen WWW:n (*World Wide Web*) yhtenä suurimpana ongelmana on tiedonhaun vaikeus. Tämä johtuu siitä, että WWW:n tietosisältö on pääasiassa tarkoitettu ihmisen luettavaksi, eivätkä koneet pysty ymmärtämään tiedon merkityksiä. Näin ollen hakukoneet eivät pysty myöskään löytämään tietoa järkevästi, koska ne yrittävät hakea tietoa vain tiettyihin hakusanoihin perustuen. Lisäksi WWW:n ongelmana on muun muassa epävarmuus sieltä löytyvän tiedon oikeellisuudesta.

Ratkaisuksi WWW:n ongelmiin on alettu kehittää semanttista webiä, josta pyritään tekemään nykyisen WWW:n kehittynyt laajennos. Semanttisessa webissä tieto esitetään sellaisella tavalla, että ihmisten lisäksi myös koneet ”ymmärtävät” tiedon merkityksiä ja pystyvät näin ollen hyödyntämään tietoa entistä tehokkaammin. Tämä mahdollistaa WWW:n nykyistä paljon monimuotoisemman käytön älykkäämpien palveluiden ja sovellusten tullessa mahdollisiksi. Etenkin tiedon etsiminen WWW:stä tulee helpottumaan tulevaisuudessa semanttisten hakujen kautta.

Semanttinen web siihen liittyvine tekniikoineen ja sovelluksineen on yhä kehitysvaiheessa, eikä vielä pystytäkään tarkasti arvioimaan, milloin näiden tekniikoiden käyttö tulee oleellisesti lisääntymään. Muutos kohti semanttista webiä tapahtunee kuitenkin vähitellen seuraavien vuosien kuluessa, kun semanttista sisältöä tulee yhä enemmän nykyisen sisällön rinnalle tai osittain sitä korvaamaan. Koko WWW:n muuttuminen semanttiseksi ei ole vielä tällä hetkellä kovin realistinen tavoite, mutta osittainen muutos esimerkiksi tietyillä sovellusaloilla voi hyvinkin toteutua lähitulevaisuudessa.

Tiedon koneellisen ymmärtämisen mahdollistaa tiedon esittäminen semanttisesti eli merkityksellisesti. Tämä tarkoittaa tiedon tallentamista sellaisessa muodossa, että varsinaisen ihmiselle luettavaksi tarkoitetun tiedon lisäksi se sisältää myös informaatiota omasta merkityksestään. Tällainen tiedon esitys edellyttää ensinnäkin järjestelmää tiedon ja sen merkityksen esittämiseen, ja lisäksi tietoon liittyvät käsitteet on pystyttävä määrittelemään yksiselitteisesti. Tiedon ja sen merkityksen esittämisen mahdollistaa semanttinen RDF-tietomalli (*Resource Description Framework*), ja tiedon käsitteiden määrityksen mahdollistavat ontologiat.

Semanttista webiä ja siihen liittyviä tekniikoita voidaan soveltaa useilla eri aloilla, ja tämän tutkielman tavoitteena onkin tutkia soveltamismahdollisuuksia sukututkimuksessa. Sukututkijat tallentavat nykyisin tietonsa sukututkimusohjelmistoi-

hin, joiden väliseen tiedonsiirtoon käytetään sukutiedon kuvaamiseen kehitettyä GEDCOM-standardia (*GEnealogical Data COMmunication*). Tiedon siirrettävyys on kuitenkin haasteellista, ja muun muassa siihen liittyviin ongelmiin semanttisen webin tekniikat voisivat vastata nykyisiä menetelmiä paremmin. Tutkielmassa tarkastellaan mahdollisuuksia muuntaa olemassa olevaa sukutietoa semanttiseen muotoon sekä semanttisten tekniikoiden hyödyntämistä sukutiedon käsittelyssä.

Tutkielma jakautuu eri lukuihin seuraavasti. Luvussa 2 tarkastellaan yleisesti semanttista webiä ja määritellään keskeisimpiä siihen liittyviä käsitteitä. Luvussa 3 kerrotaan semanttisen webin tiedonesitysmuodoista, kuten RDF-metatiedosta ja ontologiakielistä, sekä käydään läpi ontologioiden kehittämiseen liittyviä menetelmiä ja tekniikoita. Luvussa 4 perehdytään sukutiedon mallintamiseen sekä sukututkimuksessa nykyisin käytettäviin tiedonsiirron standardeihin ja ohjelmistoihin. Luvussa 5 kuvataan tutkielman empiirisessä osuudessa tehtyä tutkimusta sukutiedon esittämisestä semanttisen webin keinoin. Luku 6 on yhteenveto tutkielmasta.

2 Semanttinen web

Tässä luvussa kerrotaan semanttisesta webistä ja keskeisimmistä siihen liittyvistä käsitteistä yleisellä tasolla. Luvussa 2.1 tarkastellaan WWW:ssä esitettävän tiedon merkityksen koneellisen ymmärtämisen tärkeyttä sekä siihen liittyviä ongelmia. Luvussa 2.2 käsitellään metatietoa ja ontologioita, ja luvussa 2.3 kerrotaan lyhyesti tietämysjärjestelmistä ja niihin liittyvistä käsitteistä.

2.1 Merkityksen ymmärtämisestä

Suurin osa nykyisen WWW:n sisällöstä on suunniteltu ihmisten luettavaksi, eivätkä tietokoneet voi sisältöä tarkoituksenmukaisesti juurikaan hyödyntää. Koneet pystyvät kylläkin löytämään taitavasti esimerkiksi WWW-sivujen otsakkeita tai linkkejä toisille sivuille, mutta yleisesti ottaen ne eivät pysty ”ymmärtämään” tai tulkitsemaan WWW:ssä olevaa jäsentämätöntä tietoa [Berners-Lee ym., 2001]. WWW-sivujen sisällön koneellinen ymmärtäminen olisi kuitenkin hyvin tärkeää älykkäiden, helposti käytettävien WWW-sovellusten kehitykselle, sillä ihmisten lisäksi WWW:tä käyttävät nykyään yhä enenevässä määrin erilaiset hakukoneet, sähköisen kaupankäynnin agentit sekä monet muut koneet [Hyvönen, 2002, s. 3-4].

Vastauksena tälle koneellisen ymmärtämisen tarpeelle onkin alettu kehittää semanttista webiä eli niin sanottua merkitysten verkkoa. Semanttinen web on visio seuraavan sukupolven verkosta, jota käyttävät niin ihmiset kuin koneetkin. Se ei kuitenkaan ole erillinen verkko, vaan laajennos nykyisestä WWW:stä. Semanttisen webin standardien ja työkalujen avulla verkon sisällön merkityksiä voidaan esittää koneen ymmärtämässä muodossa, mikä mahdollistaa seuraavan sukupolven älykkäiden palvelujen ja sovelluksien toteuttamisen. Visio seuraavan sukupolven verkosta on kuitenkin muuttunut jo konkreettiseksi tavoitteeksi kansainvälisten tutkimus- ja standardointiprojektien ansiosta, ja näin ollen ensimmäiset askeleet nykyisen WWW:n laajentamisessa semanttiseksi on jo otettu [Hyvönen, 2002, s. 3] [Berners-Lee ym., 2001].

Tiedonhakuun liittyvää merkityksen ymmärtämisen ongelmaa voidaan lähestyä eri keinoin. Käyttäjän tarpeita tyydyttävää tietoa etsitään yleensä jostakin tietovarastosta, kuten tietokannasta. Onnistunut haku edellyttää, että tiedon merkityksiä voidaan jollain lailla käsitellä. WWW:ssä tiedonhaku perustuu linkkejä seuraavaan

etsintään tai hakukoneiden kautta sanahaulla tapahtuvaan etsintään. Tietoa voidaan myös yrittää tunnistaa käsittelemällä dokumentteja mekaanisesti siten, että niiden merkityksen rakenne selviää. Luonnollisen kielen yksittäisiä sanoja voidaan tunnistaa melko helposti, mutta kokonaisuuden ymmärtämiseksi pelkkä yksittäisten sanojen tunnistaminen ei riitä. Tekstistä voidaan kuitenkin etsiä myös muodollisia, lauseopillisia tai merkityksellisiä piirteitä, ja niiden perusteella tietovarastosta voidaan jollain lailla tunnistaa etsitty informaatio [Hyvönen, 2002, s. 4].

Kuvatuilla keinoilla ongelmaa lähestytään ihmisen tavoin, yrittämällä tulkita verkon sisältöjä sellaisina kuin ne ovat. Koneellinen tulkitseminen näillä keinoin on kuitenkin hyvin vaikeaa, sillä teksti on yleensä kirjoitettu luonnollisella kielellä, joka voi olla hyvinkin monimerkityksistä ja saattaa sisältää virheitäkin. Lisäksi suuri osa WWW:n sisällöstä ei edes ole tekstityyppistä, vaan sisältää esimerkiksi kuvia, ääniä, musiikkia, videoita tai ohjelmakomponentteja. Verkon sisältöjä tulkittaessa tarvitaan siis myös ihmisen yleistietoa, päättelykykyä ja kokemusmaailmaa. Tällaisen opettaminen tietokoneelle on erittäin hankalaa [Hyvönen, 2002, s. 5].

Jotta koneet saataisiin ymmärtämään WWW:n sisältöä, pitäisi kehittää yhä älykkäämpiä järjestelmiä. Tämä ei ole toisaalta kuitenkaan järkevää, koska ihminen voi tulla konetta puolitiehen vastaan ja helpottaa asian ratkaisua. Semanttisessa webissä ongelmaa lähestytäänkin tältä kantilta. Semanttiseen webiin liittyvissä tekniikoissa ideana on WWW:ssä olevan tiedon esittäminen sellaisessa muodossa, että tiedon merkitystä voidaan ymmärtää paremmin algoritmisin keinoin [Hyvönen, 2002, s. 5].

Tiedonhaun vaikeuden lisäksi WWW:n ongelmana on aina ollut epävarmuus sieltä löytyvän tiedon oikeellisuudesta. Ongelma piilee siinä WWW:n ydinajatuksessa ja vahvuudessa, että kuka tahansa voi julkaista tietoa ja mielipiteitään sekä päästä käsiksi toisten julkaisemaan tietoon. Semanttisessa webissä on kuitenkin tarkoitus pystyä lisäämään luottamusta toisten julkaisemaa tietoa kohtaan. Ideana on tarjota WWW:n käyttäjille mahdollisuus arvioida ja kommentoida WWW-sivuja. Tällaisen arvioinnin perusteella muiden käyttäjien on helpompi arvioida tiedon luotettavuutta. Arvioinnin suorittanut taho voidaan puolestaan tunnistaa niin sanotun digitaalisen allekirjoituksen avulla, ja arvioinnin merkitystä voidaan näin punnita [Hyvönen, 2002, s. 16-17].

2.2 Metatieto ja ontologiat

Semanttisen webin keskeisimpiä käsitteitä ovat metatieto ja ontologiat, jotka on esitelty seuraavissa alaluvuissa. Luvussa 2.2.1 määritellään metatiedon käsite ja esitellään lyhyesti tunnettu metatietostandardi Dublin Core. Luvussa 2.2.2 määritellään

ontologian käsite, esitellään lyhyesti tunnettuja ontologioita sekä selvitetään tarvetta ontologioiden käytölle.

2.2.1 Metatieto

Metatiedolla tarkoitetaan kuvailevaa tietoa toisesta tiedosta. Metatietoa voidaan käyttää moniin tarkoituksiin, esimerkiksi tiedonhakuun tai dokumenttien muutosten tai käytön seuraamiseeen. Käsitettä metatieto on alettu käyttää vasta Internet-aikakaudella kuvattaessa verkossa olevaa materiaalia, mutta itse metatietoa on ollut olemassa jo paljon ennen Internetiä ja WWW:tä. Perinteisesti metatietoa on kerätty esimerkiksi kirjastojen kortistoihin, jotka sisältävät kuvailevaa tietoa kirjoista tai muusta kirjastossa olevasta materiaalista. Kirjaan liittyvää metatietoa ovat esimerkiksi kirjan nimi, kirjailija, julkaisuvuosi, aihepiiri sekä sijaintitieto, jonka avulla kirja löytyy hyllystä. Kirjastojen kortistot eli metatietojärjestelmät ovat erillään varsinaisesta materiaalista, esimerkiksi kirjasta, mutta metatieto voi olla myös osa itse materiaalia [Hillmann, 2005]. WWW:ssä metatietoa voivat olla esimerkiksi verkossa olevan dokumentin otsikko, tekijä, viimeisimmän päivityksen ajankohta tai tekijänoikeus- ja lisenssitiedot [Manola ym., 2004, luku 1].

Maailmanlaajuinen kiinnostus metatietoa kohtaan on lisääntynyt elektronisten julkaisujen ja verkossa olevan kuvailemattoman aineiston lisääntymisen myötä. Tietomäärän koko ajan lisääntyessä verkosta on yhä vaikeampi löytää hakemaansa tietoa, koska olemassa olevat hakukoneet tarjoavat hakutulokseksi suuren määrän käyttäjän kannalta väärää tietoa. Metatietoon liittyvien standardien ja käytäntöjen yleistyessä tiedonhaku tietoverkoista tulee kuitenkin helpottumaan, sillä haku voidaan kohdistaa aikaisempaa tarkemmin johonkin tiettyyn dokumentin tietoon, esimerkiksi tekijään tai otsikkoon [Hillmann, 2005].

Dublin Core [DCMI, 2006] on tunnettu standardi metatiedon kuvaamiseen. Standardin tavoitteena on tarjota joukko termejä, joiden avulla voidaan kuvata tehokkaasti WWW:n sisältöjä ja niiden perusominaisuuksia. Dublin Coren perustermejä on 15 kappaletta, joita ovat muun muassa 'nimeke' (engl. *title*), 'tekijä' (*creator*), 'aihe' (*subject*), 'kuvaus' (*description*) ja 'julkaisija' (*publisher*) [Helsingin yliopiston kirjasto, 2002]. Tavoitteena on pitää termit mahdollisimman yksinkertaisina ja lukumäärältään vähäisinä, jotta kuka tahansa voisi käyttää standardia helposti [Hillmann, 2005]. Luvussa 3.2 esitetään, kuinka Dublin Corea voidaan hyödyntää metatiedon kuvaamisessa semanttisen webin tekniikoilla.

2.2.2 Ontologiat

Sana *ontologia* on lainattu tietotekniikkaan alun perin filosofiasta, jossa sillä tarkoitetaan ”filosofian osa-aluetta, joka tutkii olemisen ja olemassaolon käsitteitä ja olevaisen perimmäistä laatua” [Hetemäki, 1999]. Tietotekniikan ja erityisesti tekoälyn puolella ontologia-sanaa käytetään nykyisin laajasti, mutta vakiintunutta määritelmää sille ei ole kuitenkaan syntynyt. Kirjallisuudesta löytyykin useita hieman toisistaan poikkeavia määritelmiä [Guarino, 1997, s. 295]. Yleensä ontologia voidaan kuitenkin määritellä tiettyyn alaan liittyviä objekteja, niiden ominaisuuksia ja objektien välisiä suhteita kuvaavaksi teoriaksi [Chandrasekaran ym., 1999, s. 20]. Usein ontologiaa kutsutaan hieman yksinkertaistetusti käsitteistöksi tai sanastoksi.

Ontologian tietoteknisessä määritelmässä on edelleenkin nähtävissä määritelmän alkuperäinen, filosofinen tausta. Puhuttaessa olevaisesta filosofiassa tarkoitetaan olemassa olevaa todellisuutta, kaikkea olevaa. Tietoteknisessä määritelmässä tätä olevaista edustaa täsmälleen kaikki sellainen, joka pystytään jollain lailla esittämään ja mallintamaan objektien ja niiden välisten suhteiden kautta [Gruber, 1993]. Useimmiten ontologialla ei tietotekniikassa kuitenkaan edes yritetä kuvata kaikkea olemassa olevaa, vaan sitä käytetään lähinnä kuvaamaan johonkin tiettyyn sovellusalaan (engl. *domain*) liittyvää, mallinnettavissa olevaa tietoa. Täten ontologioilla kuvataan tietotekniikassa paljon suppeampia kokonaisuuksia kuin filosofiassa, vaikka lähtökohta luonnollisesti onkin sama molemmissa määritelmissä.

Ontologian avulla voidaan esittää jonkin sovellusalan – esimerkiksi biologian, elektroniikan tai taiteen – ammattikäsitteitä ja -tietämystä, yleistä arkitietämystä tai muunlaista tietämystä [Hyvönen, 2002, s. 14–15]. Tietämyksellä (engl. *knowledge*) tarkoitetaan ihmisen oppimisen ja kokemuksen kautta hankittua, sisäistettyä ja merkitykselliseksi osoittautunutta informaatiota, joka ontologiaksi mallinnettuna muuntuu sen koneella esitetyksi jäljitelmäksi [Kielikone Oy, 2006]. Ontologiat ovat hyvin erityyppisiä riippuen sovellusalasta, käyttötarkoituksesta ja tiedon esitysmekanismista [Hyvönen, 2002, s. 14]. Käsitteiden määrityksen lisäksi ontologiaan voi tyypillisesti kuulua myös joukko loogisia päättelysääntöjä, joiden avulla on mahdollista tehdä johtopäätöksiä käsitteiden välisten suhteiden perusteella [Berners-Lee ym., 2001].

Sovellusalan ontologia itsessään ei useinkaan ole tavoiteltava päämäärä, vaan ontologian muodostaminen vastaa tavallaan tietojoukon ja sen rakenteen määritystä, mikä tehdään, jotta tieto olisi jonkin ohjelman käytettävissä. Ongelmanratkaisumetodit, alasta riippumattomat sovellukset ja ohjelmistoagentit käyttävät ontologioita samalla tavalla kuin muutakin tietoa [Noy ym., 2001, s. 2].

Tunnettuja, laajoja ontologioita ovat muun muassa WordNet, RosettaNet ja Cyc [Hyvönen, 2002, s. 15]. WordNet [Miller ym., 2007] on suuri englannin kielen sanasto, joka sisältää tiedon sanojen merkityksestä ja synonyymeista. RosettaNet [RosettaNet, 2007] puolestaan on useiden teknologiayritysten yhteenliittymä, joka kehittää etenkin elektroniikkateollisuudessa käytettäviä sähköisen liiketoiminnan standardeja. Standardeihin kuuluu keskeisenä osana alakohtaisten sanastojen määritys. Cyc [Cycorp, Inc., 2007] on yleiskäyttöinen ontologia, joka määrittelee ihmisen arkitietämykseen liittyvää käsitteistöä. Eräs henkilöitä luokitteleva ontologia on FOAF (*Friend Of A Friend*) [FOAF, 2007], jonka avulla voi luoda ystävyysverkostoja. FOAF-ontologia mahdollistaa ihmisten, heidän keskinäisten suhteidensa sekä ihmisiin liittyvien tietojen määrityksen.

Yleensä ontologiat esitetään taksonomisesti siten, että käsitteet on luokiteltu järjestelmällisesti puu- tai verkkorakenteen muotoon. Rakenteen ylätasoilla termit ovat usein hyvin yleisiä ja alasta riippumattomia, ja mentäessä rakenteessa syvemmälle ne muuttuvat alakohtaisemmiksi [Chandrasekaran ym., 1999, s. 22]. Ontologiaa kuvataan yleensä joukkona luokkia (engl. *class*) ja niihin liittyviä ominaisuuksia (engl. *property*) [Noy ym., 2001, s. 3].

Tiettyyn alaan liittyvän käsitteistön määrittely vaatii yleensä huolellista analyysiä alalla esiintyvien käsitteiden ominaisuuksista ja käsitteiden keskinäisistä suhteista. Ontologista analyysiä tarvitaan, jotta tietämyksen rakenne tulisi konkreettisemmaksi ja selkeämmäksi. Tietyn alan ontologia muodostaa minkä tahansa tämän alan tiedon esitykseen liittyvän järjestelmän ytimen, eikä tietämystä voida kuvata tarkasti ilman ontologiaa tai vastaavaa käsitteistöä. Siispä kehitettäessä tehokasta järjestelmää tiedon esitykseen ensimmäisenä askeleena on suorittaa tehokas ontologinen analyysi kyseessä olevasta alasta [Chandrasekaran ym., 1999, s. 20–21]. Ontologian määrittelyn systemaattisuus ja täsmällisyys mahdollistavat myös käsitteistön automaattisen konepohjaisen tulkinnan, mikä on keskeistä tulevaisuuden WWW-ympäristössä [Hyvönen, 2002, s. 14].

Ontologioiden yhtenä päätavoitteena on mahdollistaa eri sovellusaloihin liittyvän tietämyksen jakaminen ja uudelleenkäyttö eri sovelluksissa. Kun tietyn alan ontologia on muodostettu, täsmentyy siinä koko alalle ominainen käsitteistö. Näin voidaan hyvinkin olettaa muiden saman alan sovellusten tarvitseman tietämyksen olevan pääosin samanlaista. Näin ollen muut voivat käyttää samaa ontologiaa, ja vältetään turha toisto käsitteistön analysoinnissa. Toisaalta ei ole lainkaan selvää, että jonkin alan ontologiaa voidaan aina käyttää monissa saman alan eri sovelluksissa. Alakohtainen tietämys riippuu hyvin paljon kulloisestakin tehtävästä, jota sovelluksen on tarkoitus toteuttaa [Guarino, 1997, s. 293].

Jotta tietämyksen jakaminen ja uudelleenkäyttö olisi mahdollista, täytyy löytää olemassa olevat ontologiat ja päästä niihin käsiksi. Tätä varten ontologioita onkin kerätty ontologiakirjastoihin, joista ne ovat yleisesti saatavilla. Vaikkei kirjastoista löytyisikään juuri etsittyä ontologiaa, saatetaan haluttuun lopputulokseen päästä helposti olemassa olevaa ontologiaa hieman muokkaamalla tai laajentamalla. Useat ontologiat ovat saatavilla sähköisessä muodossa [Noy ym., 2001, s. 6].

2.3 Tietämysjärjestelmät

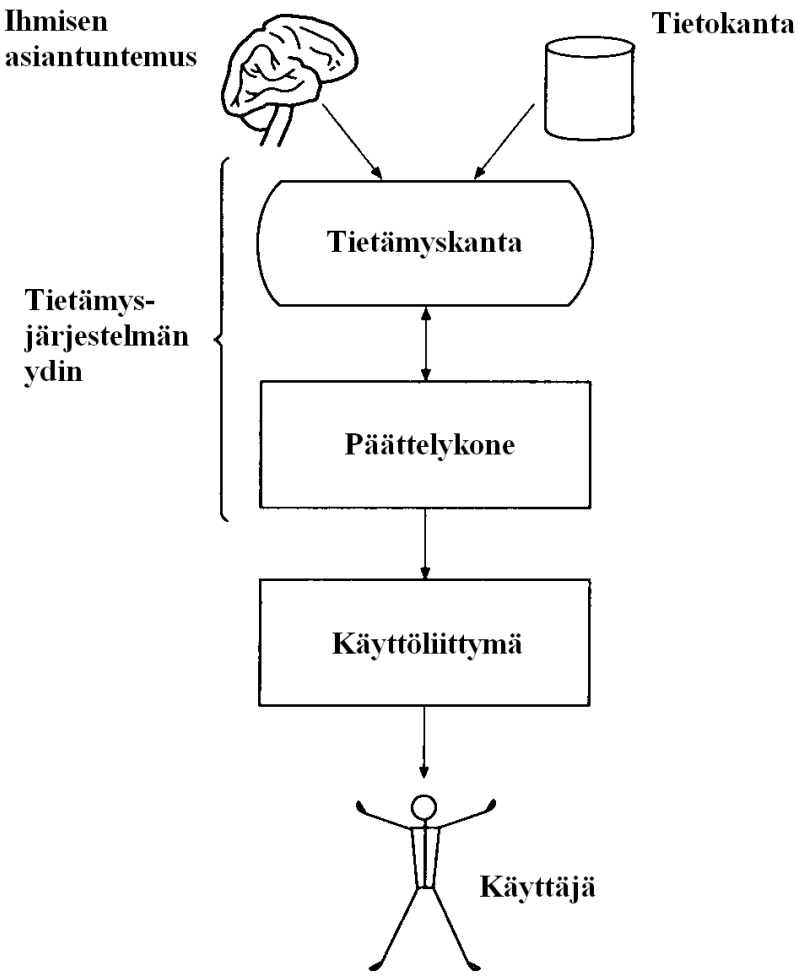
Semanttisessa webissä tietämys sekä sen esittäminen ja käsittely ovat tärkeässä roolissa. Tietämykseen pohjautuvia järjestelmiä onkin sen puitteissa kehitetty paljon, ja tässä luvussa esitellään tärkeimpiä näihin järjestelmiin liittyviä käsitteitä.

Alakohtaista tietämystä kuvaava ontologia ja sen luokkia edustavat ilmentymät (engl. *instance*) ominaisuuksineen muodostavat yhdessä tietämyskannan (engl. *knowledge base*). Ero ontologian ja tietämyskannan välillä on kuitenkin hiuksenhieno, koska joskus voi olla vaikea tulkita, missä ontologia loppuu ja missä tietämyskanta alkaa [Noy ym., 2001, s. 3]. Ongelman näiden erottamisessa voi aiheuttaa esimerkiksi se, että periaattessa samanlainen ontologia voidaan joissakin tapauksissa mallintaa eri tavoin luokkia ja ominaisuuksia määrittämällä, ja tällöin luokkien ilmentymienkin voidaan joskus tulkita olevan osa ontologiaa [Noy, 2005].

Tietämysjärjestelmä (engl. *knowledge-based system*) on tietojärjestelmä, johon kuuluu sovellusalaan liittyvä tietämyskanta sekä sitä käyttäviä ohjelmia, jotka käsittelevät ja ratkaisevat alaan liittyviä ongelmia [Stanford Medical Informatics, 2007b]. Tyypillisesti tietämysjärjestelmä käyttää ongelmien ratkaisuun päättelykonetta (engl. *inference engine*), joka tekee johtopäätöksiä tietämyskannassa olevan informaation perusteella. Kuvassa 2.1 on esitetty tietämysjärjestelmän rakenne. Kuvasta nähdään, että tietämysjärjestelmän ytimen muodostavat ihmisen asiantuntemuksesta ja tietokannan (engl. *database*) sisältämästä informaatiosta koostuva tietämyskanta sekä päättelykone, joihin käyttäjä pääsee käsiksi käyttöliittymän välityksellä [Martinsons, 1995, s. 21].

Asiantuntijajärjestelmä (engl. *expert system*) puolestaan on sellainen tietämysjärjestelmä, joka mallintaa jonkin sovellusalan asiantuntijoiden tietämystä pyrkien jäljittelemään heidän tietojaan ja taitojaan mahdollisimman hyvin [Martinsons, 1995, s. 20]. Asiantuntijajärjestelmät ovat ehkä tyypillisin esimerkki tietämysjärjestelmästä ennen semanttisen webin sovelluksia.

Tietämysjärjestelmiä on semanttisen webin puitteissa kehitetty useilla eri aloilla. Tämän tutkielman luvussa 5 tarkastellaan sukututkimusaiheista tietämysjärjes-



Kuva 2.1: Tietämysjärjestelmän rakenne [Martinsons, 1995, s. 21].

telmää, jonka tietämyskanta sisältää sukutietoa kuvailevan ontologian sekä tämän ontologian mukaan järjestettyä tietoa, josta voidaan tehdä johtopäätöksiä järjestelmään kuuluvan päättelykoneen avulla.

3 Semanttisen webin perustekniikat

Tässä luvussa käsitellään tärkeimpiä semanttiseen webiin liittyviä tekniikoita. Luvussa 3.1 tarkastellaan ensin perinteisiä tiedon esitystapoja, HTML:ää ja XML:ää. Luvussa 3.2 esitellään semanttinen RDF-malli tiedon esitykselle, ja luku 3.3 kertoo ontologiakielistä, joilla tietoa voidaan esittää RDF-mallin mukaisesti. Luvussa 3.4 kerrotaan lyhyesti päättelystä semanttisen webin näkökulmasta. Luvussa 3.5 kerrotaan ontologioiden suunnittelusta ja kehittämisestä hieman laajemmin, ja luvussa 3.6 tarkastellaan ontologioiden rakentamiseen tarkoitettuja editoreita.

3.1 HTML:stä XML:ään

Alun perin WWW kasvoi pääasiassa HTML:n ympärille. HTML-kieli (*HyperText Markup Language*) [Raggett ym., 1999] on hypertekstin merkkauskieli, jota WWW-selaimet pystyvät tulkitsemaan. Hypertekstillä tarkoitetaan tietokoneen luettavissa olevaa tekstiä, jonka sanoihin on liitetty viittauksia, joiden avulla lukija saa selityksiä tai voi siirtyä tarkastelemaan toisia kohtia tekstissä. Merkkauksella taas tarkoitetaan dokumentin osien erottelua ja niiden ominaisuuksien osoittamista käyttäen sovittuja merkintöjä, esimerkiksi muotoilua ohjaavia komentoja [Kielikone Oy, 2006].

HTML:ssä on ennalta määrätyt merkkaustunnisteet (engl. *tag*), jotka rajaavat rakenteisessa dokumentissa olevia elementtejä (engl. *element*). Elementti on dokumentin sisältöä ryhmittelevä rakenneosa, esimerkiksi otsikko, kappale, lista tai linkki. Alkutunniste aloittaa elementin ja lopputunniste lopettaa sen, ja niiden välissä on mielivaltaista dataa. Elementti merkkaustunnisteineen voidaan esittää yleisessä muodossa seuraavasti [Raggett ym., 1999, luku 3.2.1] :

```
<alkutunniste>(mielivaltaista dataa)</lopputunniste>
```

Tunnisteet ovat ennalta määrättyjä, ja selaimet kääntävät ne ennalta sovittuun muotoon. HTML suunniteltiin dokumentin rakennetta kuvaavaksi kieleksi, mutta sitä kuitenkin käytetään usein virheellisesti dokumentin muotoilun ja sommittelun määrittelyyn. Otsikon voi HTML:n avulla esittää esimerkiksi toisen tason otsikkomerkkauksella *h2* seuraavasti [Fensel, 2001, s. 51] :

```
<h2>Semanttinen web</h2>
```

HTML:ssä on monia hyviä puolia. Ensinnäkin se on niin yksinkertaista, että melkein kuka tahansa voi tehdä sen avulla kotisivun. Lisäksi HTML-sivuja esittäviä WWW-selaimia on ollut helppo kehittää kielen yksinkertaisuuden ansiosta. Juuri tämä HTML:n yksinkertaisuus on mahdollistanut WWW:n nopean kasvun siihen pisteeseen, mitä se tänä päivänä on. HTML on kuitenkin liian yksinkertainen kieli WWW:n laajempaa kehittymistä ajatellen [Fensel, 2001, s. 51].

HTML:n heikkoutena on, että se ei kerro juuri mitään tiedon merkityssisällöstä. Esimerkiksi edellä esitetyn otsikkomerkkauksen merkityksen tulkinta on hyvin epäselvä. Ainut, mitä otsikosta voidaan sanoa, on että se on toisen tason otsikkomerkkaus. Merkkaus ei siis kerro mitään itse otsikon sisällöstä ilman ihmisen tulkintaa. Tämä on suuri rajoitus tiedon automaattiselle haulle ja käsittelylle, ja siksi HTML onkin vakava rajoite mielekkäälle tiedonhaulle WWW:ssä. Lisäksi HTML:n avulla ei voi määritellä HTML:n kieliopin ulkopuolista rakenteista tietoa, minkä johdosta yleinen tiedonsiirto HTML:ää käyttäen on mahdotonta. Kaikki tieto täytyy ilmaista dokumentin rakennemäärityksen mukaisesti liittäen tieto esimerkiksi otsikoihin, taulukoihin tai kappaleisiin. HTML:n avulla voi esittää vain yksinkertaisia dokumentteja, eikä ennalta määrättyjen merkkaustunnisteiden lisäksi ole mahdollista määritellä uusia. Tämän vuoksi HTML rajoittaa pahasti edistyneempien WWW-sovellusten kehittymistä [Fensel, 2001, s. 52-53].

Semanttisen webin kannalta keskeistä metatietoa pystyy esittämään HTML:ssä *meta*-tunnisteella, jonka avulla voidaan määritellä (ominaisuus, arvo)-pareja. Esimerkiksi Dublin Core -standardin termit voidaan esittää myös HTML:n avulla [Raggett ym., 1999, luku 7]. HTML:ää voidaan käyttää metatiedon kuvaamiseen monissa tilanteissa, mutta se ei ole kuitenkaan läheskään niin ilmaisuvoimainen kuin luvussa 3.2 esitelty RDF-kieli, jolla metatietoa voidaan kuvata erityisen tehokkaasti.

HTML:n puutteista johtuen XML:n käyttö alkoi lisääntyä, sillä se on monin tavoin HTML:ää kehittyneempi kieli. XML (*eXtensible Markup Language*) [Bray ym., 2006] on yleiskäyttöinen merkkauskieli ja samalla metakieli merkkauskielten määrittelyyn. Metakieli tarkoittaa sellaista kieltä, jolla voidaan esittää jonkin toisen kielen ilmaisujen muodostamista koskevat säännöt [Kielikone Oy, 2006]. XML:n yhtenä tarkoituksena on helpottaa dokumenttien siirrettävyyttä eri alustoille ja eri tarkoituksiin.

Alun perin XML suunniteltiin vastaamaan laajamittaisen sähköisen julkaisemisen asettamiin haasteisiin, mutta nykyään sillä on yhä tärkeämpi rooli myös monentyyppisen tiedon siirrossa sekä WWW:ssä että muualla [Bray ym., 2006]. XML:n avulla tieto esitetään ja välitetään ohjelmistosovellukselta toiselle XML-

dokumentteina, joissa rakenneosat on nimetty ja merkattu sovitulla tavalla siten, että sovellukset pystyvät käsittelemään rakenneosia [Salminen, 2005, s. 1].

XML:ssä tunnisteiden nimiä ei ole mitenkään määritelty ennalta, ja täten tiedon merkitys voidaankin määritellä juuri tunnisteiden avulla. Esimerkiksi nimen voi kirjoittaa tunnisteen *name* avulla seuraavasti:

```
<name>Matti Meikäläinen</name>
```

XML mahdollistaa myös tiedon määrittämisen mielivaltaisina puu- tai verkkorakenteina. Tämä onnistuu sijoittamalla elementtien sisään toisia elementtejä, ja verkkorakenteet edellyttävät myös attribuuttien käyttöä. Verkkomainen rakenne voidaan toteuttaa linkityksen avulla *XML Linking Language* (XLink) -kieltä [DeRose ym., 2001] käyttäen. Mikäli haluttaisiin esittää esimerkiksi henkilön nimi ja puhelinnumero samaan puurakenteeseen kuuluvina, voitaisiin elementit *name* ja *phone* sijoittaa elementin *person* sisään seuraavasti [Fensel, 2001, s. 53] :

```
<person> <name>Matti Meikäläinen</name>
         <phone>123456</phone> </person>
```

Laajennettavuus on XML:n tärkeä ominaisuus. Se tarkoittaa sitä, että XML:n käyttäjät voivat määritellä omia XML-pohjaisia kieliään, joita kutsutaan *XML-sovelluksiksi*. Näissä kielen sovelluksissa hyödynnetään juuri sitä, että XML:n tunnisteita ei ole määrätty ennalta. Uutta XML-sovellusta määritettäessä elementtien nimet ja dokumenttirakenteet valitaan tiettyä sovellusalaa varten, ja täten XML voidaan siis mukauttaa erityisten tietorakenteiden vaihtoon ohjelmistojen välillä. Kuka tahansa voi määritellä ja ottaa käyttöön oman XML-sovelluksensa. Oman XML-kielen hyödyntäminen saattaa kuitenkin jäädä tehottomaksi, ellei käytettävissä ole kieltä tukevia ohjelmistoja. XML-sovelluksia kehitetäänkin paljon kansainvälisissä organisaatioissa ajatuksena se, että kieliä tukemaan voidaan sitten kehittää myös erikoisohjelmistoja [Salminen, 2005, s. 11, 14].

Esimerkkejä W3C:ssä (*World Wide Web Consortium*) kehitetyistä XML-sovelluksista ovat MathML (*Mathematical Markup Language*) matemaattisen tekstin esittämiseen, XML-Signature digitaalisiin allekirjoituksiin ja XHTML web-julkaisemiseen. XHTML on HTML-kielestä kehitetty niin, että kieli noudattaa XML:n sääntöjä, mutta siinä on kuitenkin samanlaiset rakenteet kuin HTML-kielessä [Salminen, 2005, s. 14].

XML-sovellusten määrittämisessä ongelmaksi voi muodostua sovellusten keskinäinen päällekkäisyys. Koska jokainen voi määrittää mielivaltaisesti omat XML-tunnisteensa ja -sovelluksensa, niin eri sovelluksissa on helposti päällekkäisyyttä.

Mikäli tällaisia sovelluksia halutaan käyttää yhdessä, täytyy ne erikseen muokata toisiaan vastaaviksi, mikä vaatii ylimääräistä työtä.

Mainittakoon tästä esimerkkinä muutama erilaisten verkkojen kuvaamiseen tarkoitettu XML-pohjainen merkkauskieli. Tällaisia ovat muun muassa RDF, XTM, XMI ja GXL. RDF on kieli WWW:n sisältöjen kuvaamiseen, jota voi käyttää esimerkiksi juuri XML:n syntaksin mukaisesti. RDF:stä on kerrottu tarkemmin seuraavassa luvussa. XTM (*XML Topic Maps*) [Pepper ym., 2001] on niin sanottu XML-aihekartta, jolla tiedon rakennetta voi kuvata vastaavalla tavalla kuin RDF:llä kin. XMI (*XML Metadata Interchange*) [OMG, 2005] puolestaan on kieli, jolla voidaan kuvata muun muassa UML-malleja XML-muodossa. GXL (*Graph Exchange Language*) [Holt ym., 2002] on XML-pohjainen merkkauskieli, jolla voidaan kuvata erilaisia verkkoja.

XML-dokumentin sisältämä tietokokonaisuus on usein tarkoitettu ensisijaisesti ihmiselle esitettäväksi. Tällöin sisällölle täytyy erikseen määritellä ulkoinen esitysmuoto, joka mahdollistaa dokumentin tietosisällön esittämisen ihmiselle havainnollisella tavalla. Tämä määritellään erikseen erityisten tyylisivujen avulla. Mikäli XML-dokumentin sisältämä tietokokonaisuus on kuitenkin tarkoitettu ainoastaan tietokoneohjelmien välillä siirrettäväksi, ei dokumentille ole tarpeen määritellä ulkoista esitysmuotoa lainkaan [Salminen, 2005, s. 10-11].

XML siis mahdollistaa sen, että käyttäjä voi lisätä dokumenttiin mielivaltaisia rakenteita, mutta rakenteiden merkitystä ei kuitenkaan ole mitenkään määritelty. Esimerkiksi ylläoleva henkilön nimen ja puhelinnumeron sisältävä rakenne näyttää varsin selkeältä, mutta rakenteen osasten nimeämisestä huolimatta kone ei pysty sen varsinaista merkitystä ymmärtämään. Merkityksen ymmärtämiseen tarvitaan semanttinen WWW-kieli, joista tunnetuin on seuraavassa luvussa esitelty RDF [Hyvönen, 2002].

3.2 RDF metatiedon esityskielenä

RDF (*Resource Description Framework*) [Manola ym., 2004] on WWW:n sisältöihin eli resursseihin liittyvän tiedon esittämiseen tarkoitettu kieli, jolla kuvataan erityisesti resursseihin liittyvää metatietoa. RDF on tarkoitettu tilanteisiin, joissa tiedon täytyy olla ennemminkin sovellusten käsiteltävissä kuin vain ihmisille esitettävässä muodossa. RDF tarjoaa yleisen kehyksen tiedon ilmaisemiseen siten, että se voidaan välittää sovellukselta toiselle ilman sen merkityksen häviämistä. Mahdollisuus siirtää tietoa eri sovellusten välillä merkitsee sitä, että tietoa voivat käyttää muutkin kuin vain alun perin kyseisen tiedon käsittelyyn tarkoitetut sovellukset.

Tässä luvussa kerrotaan RDF-tietomallista ja havainnollistetaan sen käyttöä. Luvussa 3.2.1 määritellään ensin lyhyesti RDF-mallin käytön mahdollistava URI-tunniste, ja luvussa 3.2.2 esitellään itse RDF-tietomalli sekä käydään läpi sen käyttöä ja erilaisia esitystapoja. Luvun teksti perustuu pääasiassa RDF:n määrittelydokumenttiin [Manola ym., 2004], joten lähdeviitteet on merkitty vain käytettäessä tästä poikkeavaa lähdettä.

3.2.1 URI-resurssitunnisteet

RDF perustuu ideaan, että asiat yksilöidään niiden WWW-tunnisteiden eli URI:en avulla. URI (*Uniform Resource Identifier*) on Internet-verkon resurssin tunnus, joka voi olla sijainnin ilmoittava resurssinpaikannin eli URL (*Uniform Resource Locator*) tai resurssin symbolinen nimi eli URN (*Uniform Resource Name*) [Berners-Lee ym., 2005]. Resurssiksi voidaan kutsua mitä tahansa sellaista asiaa, joka on tunnistettavissa URI-tunnisteen avulla.

URL-tunnisteita ovat nykyiset WWW-osoitteet, jotka kertovat WWW-sivun eli verkkoresurssin tarkan sijainnin. URN-tunniste sen sijaan voi määrittää yksikäsitteisesti ja pysyvästi jonkin resurssin, vaikka se ei olisikaan verkossa saatavilla. URN-tunnisteita ovat esimerkiksi kirjojen kansainvälinen numerotunnus ISBN (*International Standard Book Number*) sekä digitaalisten julkaisujen – esimerkiksi artikkelien – yksilöintitunnus DOI (*Digital Object Identifier*) [Varjonen, 2006].

URI-tunnisteille on ominaista, että henkilöt tai järjestöt voivat itsenäisesti luoda tunnisteita ja käyttää niitä yksilöimään erilaisia asioita. WWW:n kehittymisen kannalta on tärkeää, että sellaistakin tietoa voidaan yksilöidä, jota ei voida suoraan paikantaa verkossa. Tämän ansiosta RDF:n avulla voidaan esittää tietoa esimerkiksi verkko-ostospaikan myyntiartikkeliin liittyen – vaikkapa tuote-esittely, hinta tai saatavuustieto – tai kuvata WWW-käyttäjän mieltymyksiä tiedonvälitykseen liittyen.

3.2.2 RDF-tietomalli

RDF-tietomallissa resursseja kuvataan yksinkertaisten väittämien avulla. Kuvattavilla asioilla on *ominaisuuksia*, jotka voivat saada jonkin *arvon*, ja nämä arvot määritetään juuri väittämien avulla. Kuvattavana asiana eli resurssina voisi olla vaikkapa WWW-sivu, ja sen ominaisuutena sivun tekijä. Kun tätä lähdetään mallintamaan RDF:n avulla, muodostetaan väittämä ensin luonnollisella kielellä, jolloin se voisi olla esimerkiksi seuraavanlainen:

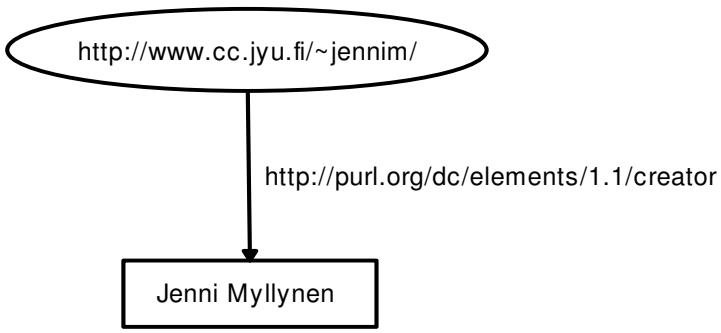
> Sivun http://www.cc.jyu.fi/~jennim/ tekijä on Jenni Myllynen.

Tässä väittämässä sivuun http://www.cc.jyu.fi/~jennim/ liittyy ominaisuus 'tekijä', joka saa arvon 'Jenni Myllynen'. RDF:ssä käytetään erityisiä termejä puhuttaessa väittämän eri osista. Se osa väittämää, joka yksilöi kuvattavan asian, on nimeltään *subjekti*. Subjektiin liittyvän ominaisuuden yksilöivää osaa kutsutaan *predikaatiksi* ja ominaisuuden arvon yksilöivää osaa puolestaan *objektiksi*. Näistä osista muodostuvaa kokonaisuutta kutsutaan (subjekti, predikaatti, objekti)-kolmikoksi, ja edellisen esimerkin tapauksessa kolmikko on seuraava:

> (http://www.cc.jyu.fi/~jennim/, tekijä, Jenni Myllynen)

Resursseihin liittyviä ominaisuuksia voidaan kuvata täsmällisemmin esimerkiksi Dublin Core -standardin mukaisilla termeillä. Ominaisuutta 'tekijä' voidaan kuvata sitä vastaavalla termillä 'creator', joka on yksilöity URI-tunnuksella http://purl.org/dc/elements/1.1/creator [DCMI, 2006].

Graafisesti väittämiä mallinnetaan RDF-kaaviona, joka koostuu solmuista ja niiden välisistä suunnatuista kaarista. RDF-kaaviossa subjektia ja objektia kuvataan solmulla, ja niiden suhdetta ilmaisevaa predikaattia kuvataan kaarella, joka suuntautuu aina subjektista objektiin. Jokainen väittämä esitetään siis (solmu, kaari, solmu)-kokonaisuutena [Klyne ym., 2004, luku 3.1]. Esimerkkiväittämä voidaan esittää RDF-kaaviona kuvan 3.1 mukaisesti.



Kuva 3.1: RDF-kaavio yhdestä väittämästä.

Esitetty väittämä on ihmislukijalle selkeä, mutta RDF:n tarkoituksena on tehdä väittämiä, joita koneet pystyvät käsittelemään. Jotta esimerkin kaltaiset väittämät saataisiin koneiden käsiteltävään muotoon, tarvitaan ensinnäkin koneellisesti käsiteltävien tunnisteiden järjestelmä, jolla voidaan tunnistaa väittämän subjekti, predikaatti ja objekti ilman sekaannuksen mahdollisuutta toiseen samalta näyttävään

tunnisteeseen, jota joku toinen saattaa jo käyttää WWW:ssä. Lisäksi tarvitaan koneen käsiteltävissä oleva kieli väittämien esittämiseen ja välittämiseen koneelta toiselle. Olemassa oleva WWW-arkkitehtuuri tarjoaa jo nämä molemmat ominaisuudet. URI-tunnisteiden avulla väittämän eri osat voidaan yksilöidä niin, ettei sekaannuksen vaaraa ole. RDF-väittämien koneelliseen esittämiseen ja koneiden väliseen siirtoon RDF käyttää XML-kieltä. RDF määrittelee erityisen XML-kielen sovelluksen, josta käytetään nimitystä RDF/XML.

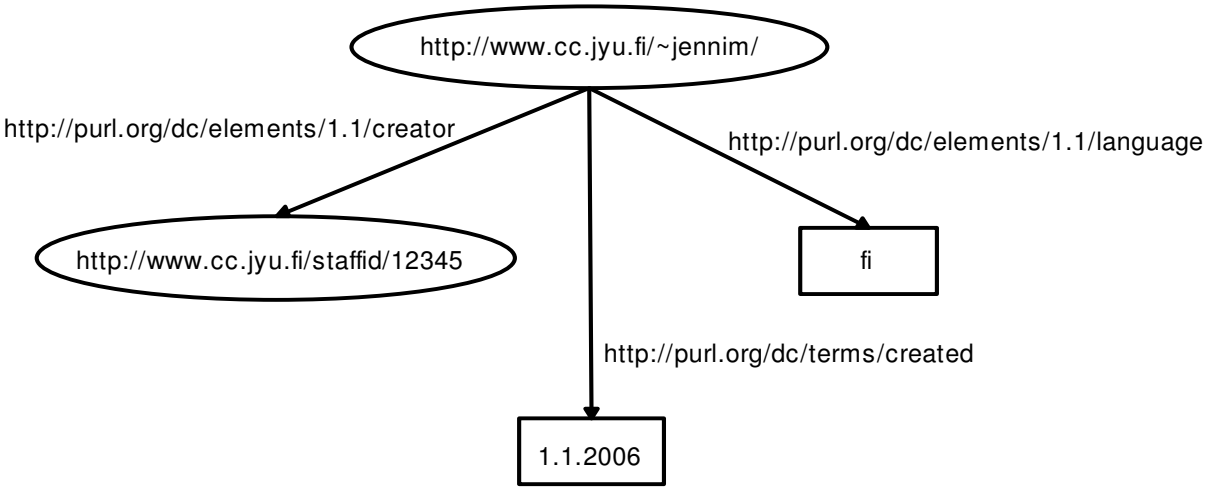
RDF:ssä on tarkoituksena, että väittämässä varsinaisen WWW-osoitteen eli subjektin lisäksi myös predikaatti ja objekti voidaan yksilöidä URI-tunnisteen avulla sen sijaan, että käytettäisiin esimerkiksi sanoja 'tekijä' ja 'Jenni Myllynen'. RDF-väittämässä predikaatti on ilmaistava aina URI-tunnisteella, kuten alun esimerkissäkin tehtiin kuvaamalla 'tekijää' Dublin Core -termillä 'creator'. Objektit voivat olla joko URI-tunnisteita tai vakioarvoja, joita kutsutaan literaaleiksi. Literaalit ovat merkkijonoja, joiden avulla voidaan esittää ominaisuuksien tiettyjä arvoja. Subjektin tai predikaatin esittämiseen literaaleja ei voi käyttää. RDF-kaavioita piirrettäessä URI-tunnisteen sisältävät solmut kuvataan ellipseinä, kun taas literaalin sisältävät solmut kuvataan suorakulmioina. Tämä nähtiin myös edellä olevassa kuvassa 3.1.

Tehdään nyt uusia väittämiä esimerkin WWW-sivuun liittyen. Ominaisuuksia voisivat olla vaikkapa sivun luontipäivämäärä sekä kieli, jolla sivun teksti on kirjoitettu, ja näistä saadaan seuraavat väittämät:

> `Sivun` http://www.cc.jyu.fi/~jennim/ `luontipäivämäärä on 1.1.2006.`
>
> `Sivun` http://www.cc.jyu.fi/~jennim/ `kieli on suomi.`

Näitä ominaisuuksia voidaan kuvata Dublin Core -termeillä 'created' ja 'language', jotka on yksilöity URI-tunnuksilla `http://purl.org/dc/terms/created` ja `http://purl.org/dc/elements/1.1/language` [DCMI, 2006]. Lisäksi esimerkeissä käytettyjä objekteja voidaan esittää sellaisessa muodossa, että ne ovat täysin yksiselitteisiä. Voidaan kuvitella, että Jyväskylän yliopiston henkilöt voitaisiin yksilöidä URI:n `http://www.cc.jyu.fi/staffid/` avulla, ja näin ollen Jenni Myllystä kuvaava URI voisi olla vaikkapa `http://www.cc.jyu.fi/staffid/12345`. Predikaatin `http://purl.org/dc/elements/1.1/language` tapauksessa literaali 'fi' on kansainvälisen standardin (ISO 639-1) mukainen kaksikirjaiminen koodi suomen kielelle.

Graafisessa esityksessä samaan kaavioon voidaan yhdistää useita eri väittämistä muodostuvia kolmikkoja. Kaikki kolme edellä esitettyä väittämää voidaan esittää RDF-kaaviona kuvan 3.2 mukaisesti.



Kuva 3.2: RDF-kaavio useista, samaan resurssiin liittyvistä väittämistä.

RDF-väittämien koneelliseen esittämiseen käytettävässä RDF/XML-syntaksissa etuna on yhteensopivuus XML-työkalujen kanssa. On kuitenkin huomattava, että RDF-tietomalli ei ole sidottu XML-perustaiseen syntaksiin, vaan myös muut kuin XML:ään perustuvat RDF:n esitystavat ovat mahdollisia [Antoniou ym., 2004, s. 62].

Seuraavassa esimerkissä on kuvattu edellä esitetyt kolme väittämää RDF/XML-muodossa:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dc2="http://purl.org/dc/terms/">
  <rdf:Description rdf:about="http://www.cc.jyu.fi/~jennim/">
    <dc:creator rdf:resource="http://www.cc.jyu.fi/staffid/12345"/>
    <dc2:created>1.1.2006</dc2:created>
    <dc:language>fi</dc:language>
  </rdf:Description>
</rdf:RDF>
```

RDF:ssä graafinen esitys on väittämän ensisijainen esitystapa, mutta joskus kaavioiden piirtäminen voi olla hankalaa, joten väittämät voidaan esittää myös vaihtoehtoisella tavalla. Tähän käytetään aiempana mainittua kolmikkoesitysmuotoa

siten, että jokainen kaaviossa esitetty väittämä kirjoitetaan (subjekti, predikaatti, objekti)-kolmikoksi. Tätä esitystapaa käytettäessä väittämän eri osat ovat juuri tässä järjestyksessä, ja URI-tunnisteet merkitään kulmasulkeisiin. Esimerkiksi kolme yllä esitettyä väittämää voitaisiin esittää kolmikkoina seuraavasti:

```
<http://www.cc.jyu.fi/~jennim/> <http://purl.org/dc/elements/1.1/creator>
  <http://www.cc.jyu.fi/staffid/12345> .

<http://www.cc.jyu.fi/~jennim/> <http://purl.org/dc/terms/created>
  "1.1.2006" .

<http://www.cc.jyu.fi/~jennim/> <http://purl.org/dc/elements/1.1/language>
  "fi" .
```

Tässä kolmikkoesitysmuodossa kukin kolmikko vastaa RDF-kaavion yhtä kaarta. Kolmikossa on tieto kaaren alku- ja loppusolmusta sekä itse kaaresta. Kolmikoilla voidaan siis esittää täsmälleen sama tieto kuin väittämän graafisella esityksellä.

RDF:llä voidaan kuvata resursseja yksinkertaisten väittämien avulla, mutta se ei kuitenkaan itsessään tarjoa keinoa määritellä alakohtaista käsitteistöä eli ontologiaa. Tällaista kuitenkin tarvitaan resurssien tehokkaaseen kuvaamiseen. RDF:n lisäksi tarvitaankin siis vielä erillinen kieli, jolla voidaan määrittää käsitteitä ja niiden välisiä suhteita.

3.3 Ontologioiden esityskielistä

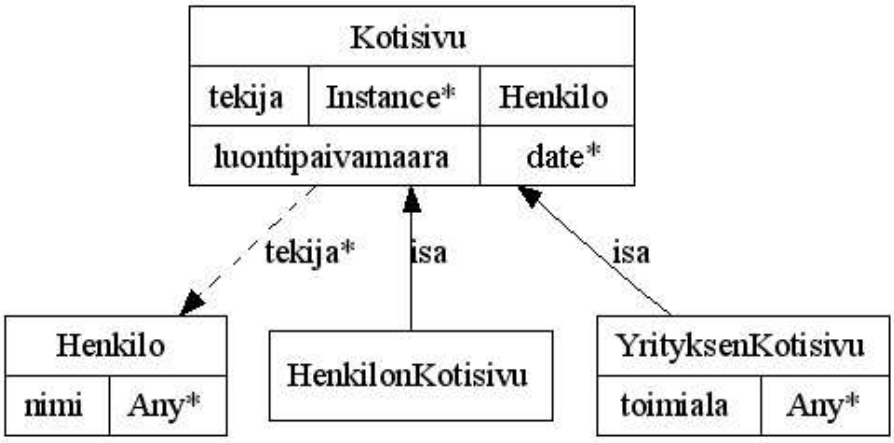
Ontologioita voidaan kuvata erilaisten ontologiakielten avulla. Viimeisen vuosikymmenen aikana ontologioiden kehittämiseen on käytetty lukuisia eri kieliä, jotka eroavat paljon toisistaan. Kielet voidaan jakaa kahteen pääryhmään, perinteiset ontologiakielet ja WWW-pohjaiset ontologiakielet. Perinteiset kielet on kehitetty tietyille käyttäjäyhteisöille ennen WWW:n yleistymistä, kun taas WWW-pohjaiset kielet on kehitetty nimenomaan semanttista webiä silmällä pitäen [Corcho ym., 2001, s. 50].

Tässä tutkimuksessa keskitytään tarkastelemaan WWW-pohjaisia RDFS- ja OWL-ontologiakieliä. Luvussa 3.3.1 kuvataan lyhyesti RDFS-kieli, ja luvussa 3.3.2 vastaavasti OWL-kieli.

3.3.1 RDF Schema

RDF Schema -kieli [Brickley ym., 2004] eli lyhyemmin RDFS mahdollistaa RDF-kuvauksissa käytetyn sanaston määrityksen. RDFS:ää voidaan sanoa RDF:n semanttiseksi laajennokseksi, sillä sen avulla voidaan määrittää ontologista tietoa RDF-kielen rakenteita käyttäen. RDFS mahdollistaa sovellusalaan liittyvien luokkien ja ominaisuuksien määrittelyn, ja RDF-väittämät voidaan sitten kirjoittaa näiden määrittelyjen pohjalta tekemällä väittämän resursseista luokkien ilmentymiä.

Määritellään esimerkkinä muutama WWW-sivuihin ja henkilöön liittyvä luokka sekä näihin liittyviä ominaisuuksia RDFS:ää käyttäen. Tarkastellaan ensin esimerkissä käytettävää luokkajakoa, joka näkyy kuvassa 3.3.



Kuva 3.3: Kotisivuihin liittyvät luokat ja niiden ominaisuudet.

Kuvassa on esitetty esimerkkiontologian luokat ja niihin liittyvät ominaisuudet sekä näiden väliset suhteet. Luokat ja niiden ominaisuudet on merkitty kaavioon suorakulmioilla, luokkien periytyvyys yhtenäisillä nuolilla ja luokkien välistä suhdetta kuvaava ominaisuus katkoviivoitetulla nuolella. Kuten kuvasta nähdään, esimerkin luokkia ovat *Kotisivu* ja sen aliluokat (*is a*) *HenkilonKotisivu* ja *YrityksenKotisivu* sekä luokka *Henkilo*. Kotisivuihin liittyviä ominaisuuksia ovat *tekija*, *luontipaivamaara* sekä *toimiala*, ja henkilöön liittyvänä ominaisuutena on *nimi*. Kunkin ominaisuuden sallittu tyyppi eli arvojoukko on merkitty sen nimen perään. Ominaisuus *tekija* voi saada arvokseen *Henkilo*-luokan ilmentymän (*Instance*) ja *luontipaivamaara* päivämäärä-tyyppisen (*date*) arvon. Ominaisuuksille *nimi* ja *toimiala* arvojoukkoa ei ole määritelty eli ne voivat saada minkä tahansa (*Any*) arvon. Kaikki ominaisuudet on merkitty tähdellä eli niitä voi olla luokan yhdellä ilmentymällä useita.

Seuraavassa esimerkissä yllä oleva luokkarakenne on määritelty RDFS:ää käyt-

täen [Manola ym., 2004, luku 5] :

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/kotisivut">

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

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

  <rdfs:Class rdf:ID="HenkilonKotisivu">
    <rdfs:subClassOf rdf:resource="#Kotisivu"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="YrityksenKotisivu">
    <rdfs:subClassOf rdf:resource="#Kotisivu"/>
  </rdfs:Class>

  <rdfs:Datatype rdf:about="&xsd;date"/>

  <rdf:Property rdf:ID="tekija">
    <rdfs:domain rdf:resource="#Kotisivu"/>
    <rdfs:range rdf:resource="#Henkilo"/>
  </rdf:Property>

  <rdf:Property rdf:ID="luontipaivamaara">
    <rdfs:domain rdf:resource="#Kotisivu"/>
    <rdfs:range rdf:resource="&xsd;date"/>
  </rdf:Property>

  <rdf:Property rdf:ID="toimiala">
    <rdfs:domain rdf:resource="#YrityksenKotisivu"/>
  </rdf:Property>

  <rdf:Property rdf:ID="nimi">
    <rdfs:domain rdf:resource="#Henkilo"/>
  </rdf:Property>

</rdf:RDF>
```

RDFS-määrityksen jälkeen luokista voidaan tehdä ilmentymiä RDF-kielellä seuraavan esimerkin mukaisesti [Manola ym., 2004, luku 5.2] :

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:esim="http://example.org/schemas/kotisivut#"
  xml:base="http://example.org/ilmentyma">

  <esim:Henkilo rdf:about="http://www.cc.jyu.fi/staffid/12345">
    <esim:nimi>Jenni Myllynen</esim:nimi>
  </esim:Henkilo>

  <esim:HenkilonKotisivu rdf:ID="jenninSivu">
    <esim:tekija rdf:resource="http://www.cc.jyu.fi/staffid/12345"/>
    <esim:luontipaivamaara
      rdf:datatype="&xsd;date">2006-01-01</esim:luontipaivamaara>
  </esim:HenkilonKotisivu>
</rdf:RDF>
```

Sovellusalan luokkien, niihin liittyvien ominaisuuksien, ominaisuuksien sallittujen arvojen sekä joidenkin muiden perusmäärittelyjen avulla ontologialle saadaan määriteltyä jo hyvä runko. RDFS mahdollistaakin RDF-sanastojen kuvailun kohtuullisesti, mutta usein niitä olisi hyödyllistä pystyä kuvaamaan RDFS:ää tarkemminkin. Tästä johtuen on kehitetty joitakin RDFS:ää rikkaampia ontologiakieliä, jotka perustuvat myös RDF-tietomalliin. Tällainen kieli on esimerkiksi seuraavassa luvussa esitelty OWL [Manola ym., 2004, luku 5.5].

3.3.2 OWL

Sekä eurooppalaiset että yhdysvaltalaiset tutkimusryhmät olivat havainneet tarpeen RDFS:ää tehokkaammalle ontologiakielelle, ja heidän yhteisenä hankkeenaan syntyikin eurooppalaisen OIL:n (*Ontology Inference Layer*) ja yhdysvaltalaisen DAML:n (*DARPA Agent Markup Language*) yhdistelmänä DAML+OIL-kieli. Tätä alettiin kehittää eteenpäin tavoitteena standardoitu ja yleisesti tunnettu ontologiakieli erityisesti semanttisen webin tarpeisiin, ja tämän kehittelyn tuloksena syntyi OWL [Antoniou ym., 2004, s. 109].

OWL-kielellä (*Web Ontology Language*) [Bechhofer ym., 2004] ontologiaa pystyy kuvailemaan tarkemmin kuin RDFS:llä. Esimerkiksi seuraavassa luetellut ominaisuudet ovat sellaisia, että ne ovat mahdollisia OWL:ssä, mutta eivät RDFS:ssä [Manola ym., 2004, luku 5.5] [Bechhofer ym., 2004, luku 4] :

- Ominaisuuden arvojoukon mahtavuudelle (engl. *cardinality*) eli arvojoukon sisältämien alkioiden lukumäärälle voidaan asettaa rajoitteita (engl. *restriction*). Esimerkki: Kotisivulla voi olla korkeintaan yksi luontipäivämäärä.
- Ominaisuus voidaan määrittää transitiiviseksi (engl. *transitive*), symmetriseksi (engl. *symmetric*), funktionaaliseksi (engl. *functional*) tai käänteiseksi (engl. *inverse*). Tällöin ominaisuudelle *O* ja mille tahansa ilmentymille *x*, *y* ja *z* ovat voimassa seuraavat lauseet:
  - Jos ominaisuus määritellään transitiiviseksi, niin tällöin pätee lause *jos O(x,y) ja O(y,z) niin O(x,z)*. Esimerkki: Jos toimiala A on osa laajempaa toimialaa B, ja B on osa laajempaa toimialaa C, niin silloin myös A on osa C:tä.
  - Jos ominaisuus määritellään symmetriseksi, niin tällöin pätee lause *O(x,y) jos ja vain jos O(y,x)*. Esimerkki: Yritys A liittyy samaan toimialaan kuin yritys B jos ja vain jos B liittyy samaan toimialaan kuin A.
  - Jos ominaisuus määritellään funktionaaliseksi, niin tällöin pätee lause *jos O(x,y) ja O(x,z) niin y=z*. Esimerkki: Jos yrityksen A suurin asiakas on B, ja toisaalta yrityksen A suurin asiakas on myös C, niin silloin voidaan päätellä B:n ja C:n tarkoittavan samaa asiakasta, vaikka niillä olisi eri nimi.
  - Jos ominaisuudet *O1* ja *O2* määritellään käänteisiksi, niin tällöin pätee lause *O1(x,y) jos ja vain jos O2(y,x)*. Esimerkki: Yrityksellä A on työntekijä B jos ja vain jos työntekijä B työskentelee yrityksessä A.
- Tietty ominaisuus voidaan määritellä jonkin luokan ilmentymien yksilölliseksi avaintunnisteeksi.
- Kaksi eri luokkaa (jotka on määritelty eri URI:a käyttäen) voidaan määrittää edustamaan samaa luokkaa.
- Kaksi eri ilmentymää (jotka on määritelty eri URI:a käyttäen) voidaan määrittää edustamaan samaa ilmentymää.

- Ominaisuuden arvojoukolle tai sen mahtavuudelle voidaan määritellä sellaisia rajoitteita, jotka riippuvat siitä luokasta, johon ominaisuus liittyy. Esimerkki: Jos kotisivuun kuuluu siihen liittyvien henkilöiden lukumäärän kertova ominaisuus, voisi se henkilön kotisivulla saada vain yhden arvon, kun taas yrityksen kotisivulla tämä sama ominaisuus voisi saada useampia arvoja.

- Luokka voidaan määritellä muiden luokkien erilaisina yhdistelminä, esimerkiksi yhdisteenä (engl. *union*) tai leikkauksena (engl. *intersection*), tai kaksi luokkaa voidaan määritellä erillisiksi (engl. *disjoint*), jolloin mikään resurssi ei voi olla näiden molempien luokkien ilmentymä.

OWL:stä on olemassa kolme eritasoista versiota eli alikieltä. Ilmaisuvoimaltaan (engl. *expressiveness*) heikoimmasta vahvimpaan ne ovat OWL Lite, OWL DL ja OWL Full. Vahvempi sisältää kaikki heikomman kielen ominaisuudet ja lisäksi kehittyneempiä kuvailuominaisuuksia. Jos siis ontologia on tehty esimerkiksi OWL DL -kielellä, on se tällöin myös OWL Full -kielinen. Siirryttäessä heikommasta kielestä vahvempaan ilmaisuvoima toisaalta lisääntyy, mutta samalla laskennallinen tehokkuus heikkenee. Mitä tehokkaammin kielellä pystytään tietoa esittämään, sitä huonommin sitä pystyy käsittelemään laskennallisesti, mikä vaikeuttaa esimerkiksi päättelyn tekemistä [McGuinness ym., 2004, luku 1.3].

OWL Lite on tarkoitettu melko yksinkertaisten ontologioiden kehittämiseen, joissa luokkahierarkian lisäksi voi olla määriteltyinä joitakin rajoitteita. OWL DL -kielessä ilmaisuvoima on niin huipussaan kuin mahdollista laskennallisuuden kuitenkin säilyessä hyvänä. OWL DL sisältää kaikki OWL-kielen rakenteet, mutta niitä voi käyttää ainoastaan rajoitetulla tavalla. OWL Full -kielessä on suurin mahdollinen ilmaisuvoima, mutta toisaalta laskennallisuudesta ei ole mitään varmuutta. On epätodennäköistä, että mikään päättelykone pystyisi tekemään aukottomasti johtopäätöksiä kaikkiin OWL Full -kielen ominaisuuksiin liittyen. OWL Full -kielessä on esimerkiksi käytössä metaluokat, eli on mahdollista, että luokka on jonkin toisen luokan ilmentymä, mikä taas ei OWL DL:ssä ole mahdollista. OWL Full -kieltä voidaan pitää RDFS:n laajennuksena, kun taas OWL Lite ja OWL DL -kieliä voidaan pitää RDFS:n rajoittuneen version laajennuksena [McGuinness ym., 2004, luku 1.3].

Seuraavassa esimerkissä on määritelty edellisessä alaluvussa käytetty luokkarakenne OWL-kielisenä siten, että esimerkkiä on laajennettu käyttäen joitakin OWL-kielen mahdollistamia rakenteita [Bechhofer ym., 2004] :

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
]>

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

  <owl:Class rdf:ID="Henkilo"/>

  <owl:Class rdf:ID="Kotisivu">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#luontipaivamaara"/>
        <owl:maxCardinality
          rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:ID="HenkilonKotisivu">
    <rdfs:subClassOf rdf:resource="#Kotisivu"/>
  </owl:Class>

  <owl:Class rdf:ID="YrityksenKotisivu">
    <rdfs:subClassOf rdf:resource="#Kotisivu"/>
    <owl:disjointWith rdf:resource="#HenkilonKotisivu"/>
  </owl:Class>

  <rdfs:Datatype rdf:about="&xsd;date"/>

  <owl:ObjectProperty rdf:ID="tekija">
    <rdfs:domain rdf:resource="#Kotisivu"/>
    <rdfs:range rdf:resource="#Henkilo"/>
  </owl:ObjectProperty>
```

```
  <owl:DatatypeProperty rdf:ID="luontipaivamaara">
    <rdfs:domain rdf:resource="#Kotisivu"/>
    <rdfs:range rdf:resource="&xsd;date"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="toimiala">
    <rdfs:domain rdf:resource="#YrityksenKotisivu"/>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="nimi">
    <rdfs:domain rdf:resource="#Henkilo"/>
  </owl:DatatypeProperty>

  <owl:ObjectProperty rdf:ID="samaToimiala">
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdf:type rdf:resource="&owl;SymmetricProperty"/>
    <rdfs:domain rdf:resource="#YrityksenKotisivu"/>
    <rdfs:range rdf:resource="#YrityksenKotisivu"/>
  </owl:ObjectProperty>

</rdf:RDF>
```

Verrattuna sivun 20 esimerkkiin luokkarakennetta on laajennettu ja täsmennetty käyttäen joitakin OWL:n mahdollistamia lisämäärittelyjä. Kotisivulle on asetettu, että sillä voi olla täsmälleen yksi luontipäivämäärä. Henkilön ja yrityksen kotisivut on määritelty erillisiksi. Lisäksi yrityksen kotisivulle on määritelty uusi ominaisuus *samaToimiala*, jolla voidaan viitata toiseen, saman toimialan yrityksen kotisivuun. Tämä ominaisuus on määritelty transitiiviseksi ja symmetriseksi.

OWL-kielikään tuskin jää semanttisen webin viimeiseksi ontologiakieleksi. Jo nyt on havaittu tarvetta joillekin uusille ominaisuuksille, ja lisää puutteita löydetään varmasti jatkossakin [Antoniou ym., 2004, s. 144].

3.4 Päättelystä

Semanttisen webin sovelluksissa päättelyllä on usein keskeinen rooli. Ontologiakielet mahdollistavat yleensä tiedon tallentamisen sellaisessa muodossa, että tietämyskannassa olevasta tiedosta voidaan tehdä johtopäätöksiä. Niiden perusteella tietämyskantaan voidaan sitten tallentaa uutta tietoa. Yleensä päättely tapahtuu erillisen päättelykoneen avulla. Päättelyä voidaan tehdä joko ontologian luokkiin tai niiden

ilmentymiin liittyen. Ontologiasta pystytään tarkistamaan muun muassa sen sisäinen yhtenäisyys, jolloin voidaan löytää luokkamäärittelyjen mahdolliset ristiriidat. Ilmentymiä voidaan päättelyn avulla muun muassa luokitella tiettyihin luokkiin kuuluviksi.

Erilaisia sääntöpohjaisia päättelyjärjestelmiä on olemassa lukuisia, mutta niiden kyky viestiä keskenään sellaisella tavalla tai siinä laajuudessa, että ne voisivat rutiininomaisesti käyttää toistensa tuloksia omassa toiminnassaan, on rajoittunutta. Yksi semanttisen webin ensisijaisista päämääristä onkin päättelyjärjestelmien yhteentoimivuuden parantaminen, ja tähän liittyen on alettu kehittää viime aikoina standardiehdotuksia [O'Connor ym., 2005]. Mitään semanttisen webin päättelyjärjestelmää ei ole vielä standardoitu viralliseksi, mutta tarve standardille on huomattu, ja sen määrittäminen onkin W3C:n tavoitteena. Päättelyjärjestelmien moninaisuus kuitenkin hankaloittaa standardin kehittelyä, koska eri tahot ovat kiinnostuneita erityyppisistä päättelymahdollisuuksista [Hawke ym., 2005].

Tärkeä askel kohti järjestelmien parempaa yhteentoimivuutta WWW:ssä on SWRL-kieli (*Semantic Web Rule Language*) [Horrocks ym., 2004], joka on kehitetty nimenomaan semanttisen webin sääntökieleksi [O'Connor ym., 2005]. SWRL:llä voi kirjoittaa OWL-kielisen ontologian käsitteitä käyttäen päättelysääntöjä, joista voidaan edelleen tehdä johtopäätöksiä jonkin päättelykoneen avulla. SWRL perustuu OWL-kielen ja RuleML-kielen yhdistelmään. RuleML (*Rule Markup Language*) [Boley, 2006] on merkkauskieli, jolla voidaan määritellä päättelysääntöjä.

SWRL:llä kirjoitettujen päättelysääntöjen avulla pystyy tekemään johtopäätöksiä tietämyskannassa olevista luokkien ilmentymistä. Päättelyllä voidaan siis muodostaa uutta tietoa sääntöjen ja tietämyskannassa jo olemassa olevan, ilmentymiin liittyvän tiedon avulla. SWRL:n määritys ei aseta rajoituksia sen suhteen, kuinka johtopäätöksiä tulisi SWRL-muotoisista säännöistä tehdä, joten päättelyyn voi käyttää mitä tahansa tehtävään soveltuvaa päättelykonetta. Myös SWRL-sääntöjen muokkaamisen voi toteuttaa vapaasti valitsemallaan tavalla. Näin ollen SWRL tarjoaa hyvän lähtökohdan sääntöjärjestelmien ja semanttisen webin integroimiselle [O'Connor ym., 2005].

Päättelytapahtumassa muodostetaan ensin päättelysäännöt, joita päättelykone vertaa sitten olemassa olevaan tietoon, ja lopuksi päättelykoneen tekemät johtopäätökset eli uusi tieto lisätään tietämyskantaan. Kuten monissa muissakin sääntökielissä, SWRL-kielessä päättelysäännöt kirjoitetaan (ehto, seuraus)-pareiksi. Säännöt kirjoitetaan ontologian käsitteitä käyttäen, joko luokkien ja ominaisuuksien nimien tai ilmentymien avulla ilmaistuina. SWRL-päättelyn avulla voidaan esimerkiksi luoda uusia suhteita ilmentymien välille liittämällä ilmentymiä toisiinsa tietyn ominai-

suuden kautta. Jos säännön mukaan kaksi ilmentymää liittyvät toisiinsa kyseisen ominaisuuden kautta, asetetaan päättelyn tuloksena tämä ominaisuus kohdalleen kaikille niille ilmentymille, jotka toteuttavat säännön [O'Connor ym., 2005]. Tämän tutkielman empiirisessä osuudessa käytetään tämäntyyppistä SWRL-päättelyä, ja siitä kerrotaan tarkemmin luvussa 5.4.3.

3.5 Ontologian suunnittelu ja kehittäminen

Ontologian suunnittelu muistuttaa oliopohjaista ohjelmistosuunnittelua, mutta merkittäviä erojakin on, sillä näkökulma luokkien ja niiden välisten suhteiden suunnittelussa on täysin erilainen. Oliopohjainen ohjelmointi keskittyy pääasiassa luokan metodeihin, ja ohjelmoija tekeekin suunnitelmat luokan toiminnallisiin ominaisuuksiin perustuen. Ontologioiden suunnittelussa sitä vastoin kiinnitetään huomiota luokan rakenteellisiin ominaisuuksiin. Näin ollen samaan sovellusalaan liittyvät ontologiamäärittely ja oliopohjainen luokkamäärittely ovat erilaisia eri suunnittelulähtökohdista johtuen [Noy ym., 2001, s. 2].

Ontologiaa ajatellaan usein puurakenteena, jossa aliluokka perii ominaisuuksia yhdeltä yliluokalta. Tällainen rakenne ei kuitenkaan riitä useinkaan kuvaamaan objektien keskinäisiä suhteita tarpeeksi kattavasti, joten ontologian kuvaamiseen tarvitaan monimutkaisempaa rakennetta. Esimerkiksi moniperintää ei pysty kuvaamaan puurakenteella, vaan sen mahdollistaa suunnattu, syklitön verkko (engl. *directed acyclic graph*, DAG). Ontologioita voidaan esittää tätäkin monimutkaisemmalla, syklisellä rakenteella, mutta tällöin esimerkiksi ontologinen päättely voi olla hankalaa [Wang ym., 2006, s. 9].

Ontologian muodostamiseen kuuluu useita vaiheita. Ensin täytyy määritellä ontologian luokkahierarkia, minkä jälkeen luokille voidaan määritellä ominaisuuksia sekä mahdollisesti niihin liittyviä rajoitteita ja muita lisämäärityksiä. Ontologioiden muodostamisessa ominaisuuksien määrittelyt ovat kuitenkin itse asiassa riippumattomia luokkien määrittelyistä. Ominaisuuksia voidaan siis määrittää, vaikkei tiedettäisi mihin luokkaan se liittyy. Jos ajatellaan esimerkiksi viinien luokittelua, kaikkia viinejä edustaa viini-luokka, ja tietyt viinit ovat sitten tämän luokan ilmentymiä. Luokalla voi olla aliluokkia, jotka edustavat käsitteitä, joille voidaan määrittää yliluokkaansa yksityiskohtaisempia ominaisuuksia. Viini-luokka voidaan esimerkiksi jakaa aliluokkiin punaviinit, valkoviinit ja roseeviinit. Vaihtoehtoisesti viini-luokka voitaisiin jakaa aliluokkiin kuohuviinit ja ei-kuohuviinit. Viinin ominaisuuksia voisivat olla esimerkiksi tuoksu, maku ja valmistaja [Noy ym., 2001, s. 3].

Ontologian määrittämisen jälkeen luodaan varsinainen tietämyskanta antamal-

la ominaisuuksille arvoja ilmentymäkohtaisesti. Ontologian rakentaminen ja ilmentymien tietojen syöttö tapahtuvat yleensä molemmat ontologiaeditoreiden avulla, joista kerrotaan tarkemmin seuraavassa luvussa. Editoreissa luokkia ja niiden ilmentymiä on usein mahdollista luoda ja muokata myös samanaikaisesti toisin kuin oliopohjaisessa ohjelmistokehityksessä, jossa luodaan perinteisesti ensin luokka, ja vasta sen jälkeen voidaan luoda luokkaa edustavia ilmentymiä eli olioita [Stanford Medical Informatics, 2007b]. Lisäksi ontologiaeditoreissa ilmentymille voidaan antaa ominaisuuksien arvoja riippumatta siitä, tiedetäänkö ilmentymien luokkaa vai ei. Mikäli luokkaa ei etukäteen tiedetä, voidaan arvojen määrityksen jälkeen tehdä luokittelu päättelykoneella, joka annettujen arvojen perusteella mahdollisesti tunnistaa ilmentymän johonkin luokkaan kuuluvaksi. Tämä onkin juuri oleellinen ero ontologian ja olio-ohjelmoinnin välillä. Ontologiaan pohjautuvassa tietämyskannassa ilmentymän luokkaa ei siis tarvitse välttämättä tietää etukäteen, toisin kuin olio-ohjelmoinnissa.

Onnistuneen tietämyskannan kehittäminen koostuu monista asioista. Ensinnäkin kehitettävä sovellus ja tietämyskannan odotettavissa olevat käyttötavat tulee suunnitella hyvin etukäteen. Useimmiten tämä tarkoittaa työskentelyä sovellusalan asiantuntijoiden kanssa, joilla on joukko ongelmia, jotka voitaisiin ratkaista tietämyskantatekniikan avulla. Ontologiaa laadittaessa sovellusala tulisi mallintaa siten, että ontologian avulla voidaan muodostaa sopiva ja tarkoituksenmukainen tiedonsyöttötyökalu tietylle käyttäjäryhmälle. Lisäksi ontologian suunnittelussa täytyy ottaa huomioon myös ongelmajoukko ja ongelmanratkaisumetodit sekä sovelluksen asettamat ajonaikaiset vaatimukset [Stanford Medical Informatics, 2007b].

Ontologioiden kehittämiseen liittyy monia käytännön vaikeuksia. Käsitteistöjen yhtenäistäminen on yleensä vaikeaa johtuen eri tahojen ja sidosryhmien erilaisista tarpeista ja mieltymyksistä. Lisäksi ontologioilla on tapana tulla hyvin laajoiksi. Ne myös muuttuvat ajan kuluessa, ja täten niiden hallinta saattaa osoittautua vaikeaksi. Esimerkkinä tästä mainittakoon valtioiden ontologia, johon kuului Tšekkoslovakia vuoteen 1993 asti, jolloin valtio jakautui kahtia. Tšekkoslovakia-termiä on ontologiassa kuitenkin käytetty viittaamaan suureen määrään tietoa, vaikkei valtiota ole enää edes olemassa [Hyvönen, 2002, s. 16].

Ontologiaeditoreiden lisäksi on olemassa muitakin työkaluja ontologioiden työstämiseen, esimerkiksi niiden yhdistämiseen on tehty joitakin järjestelmiä [Corcho ym., 2001, s. 37]. Vaikka tällaiset järjestelmät ovat kehittyneet viime vuosina, niin siitä huolimatta ontologisen tiedon käsin kokoamisen vaihe on yhä työläs ja vaivalloinen. Semanttisen webin menestyksen ja kasvun kannalta olisi kuitenkin tärkeää pystyä kehittämään alakohtaisia ontologioita nopeasti ja

halvalla. Tämän vuoksi myös ontologioiden generointia ohjelmallisesti on tutkittu ja siihen on tehty joitakin sovelluksia [Clerkin ym., 2001] [Maedche ym., 2001]. Ontologioiden kehittämistekniikat, niiden yhteinen käyttö ja yhdistäminen kuten myös ontologioiden hallinta muodostavat varmasti suuren haasteen tulevaisuudessa [Hyvönen, 2002, s. 16].

3.6 Ontologiaeditorit

Ontologioiden rakentamiseen on tehty useita ohjelmistotyökaluja eli ontologiaeditoreita. Varsinaisten ontologiaeditoreiden lisäksi myös monissa olio-ohjelmointikielten – kuten SmallTalk 80:n ja C++:n – ohjelmistokehitysympäristöissä luokkahierarkioita pystyy muokkaamaan graafisesti, mikä voidaan tulkita ontologian kehityksenä. Tällaiset editorit on kuitenkin suunniteltu ohjelmoijien käyttöön, eikä niitä ole varsinaisesti tarkoitettu esimerkiksi jonkin sovellusalan asiantuntijoille [Eriksson ym., 1999, s. 5-6].

Ontologioiden ja niihin liittyvien järjestelmien kehittäminen tapahtuu yleensä kuitenkin ryhmätyönä, johon osallistuvat niin ohjelmoijat kuin sovellusalan asiantuntijatkin, joilla ei välttämättä ole kovin paljon tuntemusta tietokoneohjelmista. Ontologiaeditorit onkin yleensä suunniteltu ohjaamaan näitä molempia tahoja järjestelmäkehitysprosessin aikana [Stanford Medical Informatics, 2007b]. Editoreissa on yleensä graafinen käyttöliittymä, mikä mahdollistaa ontologioiden luomisen tarvitsematta käyttää suoraan jotakin ontologiakieltä [Corcho ym., 2001, s. 37].

Tietämysjärjestelmän rakentaminen ja ylläpito on usein hyvin kallista. Ontologiaeditorin, tietämyskannan ja päättelykoneen avulla ohjelmoijat kuitenkin pystyvät usein käyttämään alakohtaisia ontologioita ja ongelmanratkaisumetodeja uudelleen, mikä lyhentää ohjelmointiin ja ylläpitoon käytettyä aikaa ja täten myös kustannuksia. Useat sovellukset voivat käyttää samaa alakohtaista ontologiaa ratkaistakseen erilaisia ongelmia, ja vastaavasti samaa ongelmanratkaisumetodia voidaan käyttää eri ontologioiden kanssa [Stanford Medical Informatics, 2007b].

Ontologiaeditorit eroavat paljon toisistaan, sillä jotkut niistä on tarkoitettu vain ontologisten perustoimintojen suorittamiseen, kun taas joillakin voi tehdä monimutkaisempiakin toimintoja. Joillakin editoreilla voi itse ontologian eli luokkarakenteen rakentamisen lisäksi syöttää myös tietoa luokkien ilmentymistä. Mikäli tämä ei ontologiaeditorissa ole mahdollista, tarvitaan lisäksi ilmentymien määrittelyyn oma, erillinen sovelluksensa. Eräs tärkeä, erottava tekijä editoreiden välillä on käytettävä ontologiakieli. Lisäksi mahdollisuudet siirtää tietoa toiseen editoriin (engl. *export*) sekä vastaanottaa tietoa toisesta editorista (engl. *import*) vaih-

televat suuresti. Koska käytössä on tällä hetkellä useita eri ontologiakieliä, eikä pystytä varmuudella ennustamaan mitkä niistä jäävät käyttöön, olisi ontologian hyvä olla siirrettävissä toiseen, mahdollisesti eri kieltä käyttävään editoriin [Silvonen ym., 2002, s. 143, 148–149]. Toisaalta, vaikkei editori tukisi haluttua kieltä, ei ontologian toiselle kielelle kääntäminenkään ole yleensä vaikea tehtävä [Noy ym., 2001, s. 6]. Jotkut editorit sisältävät myös päättelykoneen tai rajapinnan erilliseen päättelykoneeseen, ja tällöin niillä on mahdollista tehdä ontologiapohjaisia kyselyjä [Silvonen ym., 2002, s. 143].

Ontologiaeditoreista lähempään tarkasteluun on tässä tutkimuksessa valittu Protégé [Stanford Medical Informatics, 2007a], joka on editoreista varmastikin tunnetuin. Kyseistä järjestelmää on kehitetty pitkään, ja sillä on laaja käyttäjäkunta. Näin ollen sen voidaan olettaa kehittyneen sellaiseen pisteeseen, että sen käyttö on varmaa ja ongelmatonta, minkä vuoksi se on valittu tässä tutkimuksessa käytettäväksi. Protégé tarjoaa perustan erilaisille tietämysjärjestelmille, ja sillä voidaan rakentaa graafisesti eri aloihin liittyviä ontologioita, määrittää tiedonsyöttölomakkeita sekä syöttää varsinaista dataa eli luokkien ilmentymiin liittyvää tietoa. Protégé on ilmainen, avoimen lähdekoodin ohjelmisto.

Protégé on kehitetty Yhdysvalloissa, Stanfordin yliopiston lääketieteellisellä osastolla alun perin biolääketieteeseen liittyvien ontologioiden rakentamiseen ja esittämiseen. Nykyisin se on kuitenkin laajassa käytössä, ja sen avulla tehtyjä sovelluksia käytetään ongelmanratkaisuun ja päätöksentekoon monella eri sovellusalalla [Knublauch ym., 2004, s. 231]. Sitä käytetään muun muassa lukuisilla verkkosivustoilla sekä tutkimuslaboratorioissa ympäri maailman. Protégé onkin tällä hetkellä ehkä laajimmin käytetty ohjelmisto elektronisten tietämyskantojen ja semanttisen webin sovellusten luomisessa. Protégén yhtenä etuna muihin ontologian mallinnustyökaluihin verrattuna on mahdollisuus esittää tietoa useilla eri ontologiakielillä – esimerkiksi RDFS:llä ja OWL:llä – kun taas useissa muissa työkaluissa on keskitytty ainoastaan yhteen kieleen [Silvonen ym., 2002, s. 147].

Ontologian rakentamisessa täytyy yleensä ottaa huomioon myös kehitysprosessin muut näkökohdat. Protégé onkin suunniteltu tukemaan tietämyskannan iteratiivista kehittämistä, joka koostuu ontologioiden ja muiden tietämysjärjestelmän osien jaksottaisesta tarkistuksesta ja parantamisesta. Luokkien ilmentymien tietoja voidaan syöttää tietämyskantaan tiedonsyöttölomakkeiden kautta, jotka tehdään sovellusalakohtaisiksi ja sovellusalan asiantuntijoille helppokäyttöisiksi. Ontologiaa ja testiarvoilla täytettyjä lomakkeita näytetään usein sovellusalan asiantuntijoille tai tuleville käyttäjille kehityksen eri vaiheissa, jotta he voivat arvioida tietämyskannan ominaisuuksia ja sille asetettujen vaatimusten toteutumista. Kun tie-

tämyskannassa alkaa olla hieman enemmän tietoa, olisi sitä hyvä testata myös käytettävän sovelluksen tai ongelmanratkaisumetodin kanssa. Lopulta sovellusta tulisi testata loppukäyttäjillä. Nämä testaukset eri vaiheissa johtavat yleensä väistämättä muutosten tekoon sekä ontologian rakenteessa että tiedonsyöttölomakkeissa. Muutokset ontologiaan voivat tulla etenkin myöhemmissä vaiheissa kalliiksi, sillä tietyntyyppiset muutokset voivat johtaa jopa koko tietämyskannan uudelleenrakentamiseen. Lopullinen sovellus on ohjelma, joka yhdistää Protégén avulla tehdyn tietämyskannan sopivien ongelmanratkaisumetodien kanssa siten, että loppukäyttäjät voivat käyttää Protégén tietämyskantaa alaan liittyvien ongelmien ratkaisuun [Stanford Medical Informatics, 2007b].

Protégén komponenttipohjainen arkkitehtuuri mahdollistaa sen, että sovellukseen voi lisätä uusia toimintoja ottamalla käyttöön sopivia lisäohjelmia (engl. *plug-in program*), joita löytyy Protégén omasta lisäohjelmakirjastosta. Niiden avulla pystytään muun muassa esittämään tietokannan sisältöä graafisesti, keräämään tietoa ohjelman ulkoisista lähteistä, yhdistämään ontologioita puoliautomaattisesti sekä tarkistamaan ontologioiden rajoitteita päättelykoneen avulla [Corcho ym., 2001, s. 42]. Tässä tutkimuksessa käytettäviä lisäohjelmia ovat rajapinnan SWRL-kieltä ymmärtävään päättelykoneeseen tarjoava SWRLTab sekä tietokannan sisältöä graafisesti esittävät OntoViz ja Jambalaya. Protégé-editoria ja sen soveltuvuutta sukutiedon käsittelyyn tarkastellaan tarkemmin empiiristä tutkimusta koskevassa luvussa 5.

4 Tietotekniikan hyödyntäminen sukutiedon käsittelyssä

Tässä luvussa kerrotaan ensin sukututkimuksesta yleisellä tasolla, ja tarkastellaan sitten tarkemmin tietotekniikan hyödyntämistä sukutiedon käsittelyssä. Luvussa 4.1 tarkastellaan tietotekniikan yleistymisen vaikutusta sukututkimukseen. Luvussa 4.2 esitellään sukututkimusohjelmistoissa käytettävä tiedonsiirron standardi GEDCOM sekä joitakin uudempia standardiehdotuksia. Luvussa 4.3 tehdään katsaus nykyisiin sukututkimusohjelmistoihin.

4.1 Tietokoneistunut sukututkimus

Sukututkimus eli genealogia on historiatiede, jossa pääasiallisesti arkistolähteitä käyttäen laaditaan luetteloita sukulaisuuden mukaan samaan ryhmään kuuluvista henkilöistä. Tulokset ovat perusaineksena selvitettäessä yhteiskunnan kehittymistä, yhteisöjen sosiologista ja taloudellista rakennetta sekä oikeus- ja valtiotieteen tai perinnöllisyystieteen ongelmia [Luttinen ym., 1988, s. 18]. Sukututkimus on kuitenkin yksipuolista, jos halutaan vain laatia sukuluetteloita ja löytää mahdollisimman suuri määrä suvun jäseniä. Pelkkä esi- tai jälkipolvien kartoittaminen merkitsisi tyytymistä kuivakkaaseen nimien ja vuosilukujen muistiin merkitsemiseen. Tutkijalle onkin antoisampaa rajata itse valitsemiensa kriteerien mukaan henkilömäärää, jotta hän voisi syventyä tutkimansa suvun henkilöiden elämänvaiheisiin [Suomen Sukututkimusseura, 1999, s. 128, 131].

Sukututkimusta tehdessä on tärkeää merkitä myös kaikki tietolähteet muistiin. Aineistoa kerätessä tulee aina tarkasti merkitä kaikki arkistot, asiakirjat ja muut lähteet, joista tietoa on löytynyt. Lähteiden ilmoittaminen antaa tutkimukselle uskottavuutta ja auttaa muita tutkijoita etsimään mahdollista lisätietoa. Lähteitä ovat muun muassa kirkonarkistot, tuomioistuinten arkistot, lääninhallinnon arkistot, sotilasasiakirjat, maanmittausarkistot, siirtolaisia koskevat asiakirjat sekä suullinen perimätieto [Suomen Sukututkimusseura, 1999, s. 3–4, 131].

Ennen tietokoneiden käyttöä sukututkimuksen tulosten kirjaaminen oli työlästä sukutaulujen ja elämäkertojen käsin kirjoittamista. Kun tietokoneet saatiin tutkimuksen apuvälineiksi, muuttuivat tulosten järjestäminen, erittely ja

esittäminen merkittävästi. Tietotekniikan avulla on mahdollista pitää järjestyksessä hyvinkin suurta tietomäärää ja ottaa helposti esille halutut tiedot [Suomen Sukututkimusseura, 1999, s. 128].

Markkinoilla on ollut parin vuosikymmenen ajan sukututkimusta avustavia ohjelmistoja, joita käyttäen sukututkija on voinut tallentaa tutkimustuloksiaan yksityiseen tietokantaansa ja tuottaa tiedoista raportteja tutkimustarpeisiin tai julkaistavaksi. Ulkomaisista ohjelmistoista suomalaisiin oloihin sovitettuja laitoksia sekä kokonaan kotimaisin voimin kehitettyjä ohjelmistoja on ollut jo pitkään saatavilla useampiakin. Nykyisin yhä useampi sukukirja on tuotettu paino-originaalia myöten valmiiksi sukututkimusohjelmiston avulla [Sippu, 2000]. Sukututkimusohjelmistoista kerrotaan tarkemmin luvussa 4.3.

Sukututkimus on yleistynyt merkittävästi tekniikan kehittymisen ja Internetin myötä, kun koko tiedonkulun ketju lähteiden tutkimisesta sukutietojen tallentamiseen on helpottunut. Internetistä löytyy sukutietoa sisältäviä julkisia tietokantoja, harrastajien kotisivuja, keskusteluryhmiä ja postituslistoja, jotka mahdollistavat tutkijoiden välisen, maailmanlaajuisen yhteydenpidon. Myös sähköposti helpottaa tutkijoiden työtä, kun yhteyttä voi pitää maailmanlaajuisesti aiempaa nopeammin niin sukulaisiin kuin toisiin tutkijoihinkin [Palander, 1999, s. 30–31]. Sukututkimuksen lähdeaineistoa on myös siirretty sähköiseen muotoon, mikä voi nopeuttaa tiedon keruuta ja täten koko tutkimusprosessia ratkaisevasti [Sippu, 2000].

4.2 Sukutiedon siirtämisen standardeja

Sukututkimusohjelmistoja tutkimuksen apuvälineenä käytettäessä saattaa ilmetä tarve siirtää ohjelmiston sisältämää tietoa toiseen ohjelmistoon. Tarve saattaa syntyä tutkijan vaihtaessa käyttämäänsä ohjelmistoa tai hänen lähettäessään tietoa toiselle, eri ohjelmistoa käyttävälle tutkijalle. Toisaalta tutkija saattaa samanaikaisestikin käyttää useaa eri ohjelmistoa eri tarkoituksiin ja hyödyntää näin eri ohjelmistojen parhaimmat puolet. Yhtä ohjelmistoa saatetaan käyttää esimerkiksi sukutiedon varsinaiseen keräämiseen ja toista ohjelmistoa vaikkapa sukutaulujen tulostamiseen. Jotta ohjelmistojen välinen tiedonsiirto voisi onnistua, täytyy olla määriteltynä yhteinen tapa, jolla tieto välitetään ohjelmistolta toiselle.

Tiedonsiirtoon onkin määritelty standardeja, joista tärkein on seuraavassa alaluvussa 4.2.1 esitelty GEDCOM, jota käytetään useimmissa sukututkimusohjelmistoissa. Toisessa alaluvussa 4.2.2 on kerrottu uudemmista, XML-kieleen pohjautuvista standardiehdotuksista. Luvun 4.2.1 teksti perustuu pääasiassa GEDCOMin määrittelydokumenttiin [LDS Church, 1996], ja luvun 4.2.2 teksti GEDCOM XML:n

määrittelydokumenttiin [LDS Church, 2001], joten lähdeviitteet on merkitty vain käytettäessä näistä poikkeavia lähteitä.

4.2.1 GEDCOM

GEDCOM (*GEnealogical Data COMmunication*) [LDS Church, 1996] on sukuhistoriallisen tiedon esitystavan määrittävä kansainvälinen standardi, joka mahdollistaa eri sukututkimusohjelmistojen välisen tiedonsiirron. Itse asiassa GEDCOMia ei ole määritelty viralliseksi standardiksi missään standardointijärjestössä, mutta yleisyytensä vuoksi siitä käytetään standardi-nimitystä. GEDCOM-standardin on kehittänyt yhdysvaltalaisen Myöhempien Aikojen Pyhien Jeesuksen Kristuksen Kirkon eli mormonikirkon sukututkimusosasto tarkoituksenaan tarjota joustava ja yhtenäinen malli sukutiedon vaihtoon eri ohjelmistojen kesken. Standardin tavoitteena on edistää sukutiedon jakamista tutkijoiden kesken ja kehittää sukututkimukseen liittyviä ohjelmistoja sellaiseen suuntaan, joka auttaa suku-, historian- ja muiden tutkijoiden työtä.

Jotta GEDCOM-standardia voitaisiin hyödyntää, täytyy käytetyn sukututkimusohjelmiston tukea standardin käyttöä. Mormonikirkko on itse kehittänyt yhden sukututkimusohjelmiston, joka luonnollisestikin tukee standardia. GEDCOMista on kuitenkin tullut ohjelmistoissa yleisemminkin käytetty standardi, ja nykyään useimmat sukututkimukseen kehitetyt ohjelmistot tukevat GEDCOM-standardin mukaista tiedonsiirtoa.

GEDCOMia käytettäessä ohjelmistoon syötetty sukutieto muunnetaan ensin lähettävän ohjelmiston päässä GEDCOM-muotoiseksi tiedostoksi, joka muunnetaan jälleen vastaanottavalla puolella sukututkimusohjelmiston ymmärtämään muotoon. Muunnoksen sukututkimusohjelmiston ymmärtämästä muodosta GEDCOM-tiedostoksi ja päinvastoin hoitaa itse sukututkimusohjelmisto, mikäli se tukee GEDCOM-tiedonsiirtoa.

GEDCOMista on julkaistu useita versioita, joista viimeisin on vuonna 1996 julkaistu *The GEDCOM Standard Release 5.5*. Dokumentin määrittelyissä GEDCOMia on kuvattu kahdella eri tasolla. Matalamman tason GEDCOM-tietoformaatti (engl. *data format*) määrittelee yleiskäyttöisen tiedon esitysmuodon, jolla voi esittää mitä tahansa rakenteista tietoa. Siinä määritellään rakenteisen tiedon syntaksi eli muodollinen rakenne sekä tiedon osasten tunnistaminen. Tietoformaatissa tieto määritellään yleisellä tasolla, eikä se ota kantaa tiedon merkitykselliseen sisältöön. Näin ollen formaatin avulla voidaan käsitellä myös muuta kuin sukutietoa. Korkeampaa tasoa kuvaa GEDCOM-muoto (engl. *form*), joka määrittelee tiettyyn alaan liit-

tyvän tiedon esittämisen. Dokumentissa esitellään GEDCOM-muoto *Lineage-Linked GEDCOM Form* (suom. sukujuuret yhdistävä GEDCOM-muoto), jolla kuvataan juuri sukutietoa.

GEDCOMissa sukutiedot mallinnetaan kahdentyyppisinä tietueina, jotka koostuvat ominaisuuksille varatuista kentistä. *Henkilötietue* mallintaa yksittäistä miestai naispuolista tai sukupuoleltaan tuntematonta henkilöä, jonka tutkija arvelee olleen olemassa. Henkilötietueen kentissä esitetään lähteistä selville saatuja henkilön ominaisuuksia, kuten nimi, sukupuoli, syntymä ja kuolema sekä muita henkilön elämänvaiheisiin liittyviä tapahtumia paikka- ja aikamääreineen. Henkilötietue voi sisältää myös esimerkiksi henkilön ammatin sekä vapaamuotoista elämäkerrallista tekstiä. *Perhetietue* mallintaa ydinperhettä eli isää, äitiä ja heidän yhteisiä lapsiaan. Perhetietueen kentissä esitetään isälle ja äidille yhteiset ominaisuudet kuten avioliitto. Perheen jäsenroolit esitetään kolmentyyppisinä kytköksinä perhetietueen ja henkilötietueiden välillä. Kytkös yhdistää henkilön perheeseen isän, äidin tai lapsen roolissa [Sippu, 2000].

Seuraavassa esimerkissä on kuvattu GEDCOM-muotoinen henkilötietue:

```
0 @I1@ INDI
1 NAME Matti Matias/Meikäläinen/
1 SEX M
1 BIRT
2 DATE 1 JAN 1900
2 PLAC Jyväskylä
1 DEAT
2 DATE 31 DEC 1980
2 PLAC Jyväskylä
1 FAMC @F1@
1 FAMS @F2@
```

Kunkin rivin alussa oleva numero kuvaa tiedon hierarkiatasoa. Alemman tason eli suuremman numeron osoittaman rivin tieto liittyy edellisen hierarkiatason tietoon. 0 on ylin taso, joka aloittaa uuden tietueen. Tässä ensimmäinen rivi aloittaa henkilöön (engl. INDIvidual) liittyvän tietueen. Aloitusrivillä määritellään myös henkilö-ID, joka tässä on 'I1'. Aloitusrivin jälkeen henkilön ominaisuuksia määritellään kentässä olevan tunnisteen ja sille mahdollisesti annetun arvon avulla.

Henkilöön liittyviä tunnisteita ovat esimerkiksi tässä näkyvät nimi (NAME), sukupuoli (SEX), syntymä (BIRTh) ja kuolema (DEATh). Syntymä ja kuolema ovat tapahtumia, joille voidaan määrittää tarkempia tietoja alemman hierarkiatason kentil-

lä. Tässä tapahtumille on määritelty aikaan (DATE) sekä paikkaan (PLACe) liittyvät tiedot. Kaksi alinta tunnistetta ovat viitteitä perhetietueisiin, jotka liittävät henkilön tiettyyn perheeseen joko lapsen (FAMChild) tai puolison (FAMSpouse) roolissa. Tässä henkilö on liitetty perheisiin, joiden perhe-ID:t ovat 'F1' ja 'F2'.

Seuraavassa esimerkissä on kuvattu GEDCOM-muotoinen perhetietue:

```
0 @F2@ FAM
1 HUSB @I1@
1 WIFE @I2@
1 CHIL @I3@
1 CHIL @I4@
1 MARR
2 DATE 1 JAN 1930
2 PLAC Jyväskylä
1 DIV
2 DATE 31 DEC 1960
2 PLAC Jyväskylä
```

Perhetietueessa (FAMily) on pääasiassa viitteitä henkilöihin, jotka kuuluvat kyseiseen perheeseen. Esimerkin perheeseen kuuluvat aviomies (HUSBand), vaimo (WIFE) ja kaksi lasta (CHILd), joihin on viitattu heidän ID-tunnuksillaan. Lisäksi perheelle on merkitty tapahtumat avioliitto (MARRiage) ja avioero (DIVorce) sekä niihin liittyvät aika- ja paikkatiedot.

Näissä esimerkeissä on käytetty vain muutamia GEDCOM-standardin useista kymmenistä eri tunnisteista, eikä niihin tässä tutkimuksessa perehdytäkään tämän tarkemmin.

GEDCOM-standardista huolimatta tietojen siirrettävyys ohjelmistojen välillä on käytännössä melko heikkoa. Monen ohjelmiston tietokantamalli on sisällöltään suppeampi kuin GEDCOM-malli, eikä sellaisen ohjelmiston tietokantaan voi siirtää tietoa laajempaa mallia käyttävän ohjelmiston tietokannasta, ainakaan osaa alkuperäisen tietokannan rakenteesta hävittämättä. Joissakin ohjelmistoissa taas käytetään GEDCOMin epästandardeja laajennoksia, joiden mukaisia tiedostoja ainoastaan kyseinen ohjelmisto pystyy häviöttömästi tuomaan tietokantaan [Sippu, 2000].

Lisäksi GEDCOM-standardin määrittely on melko monimutkainen ja osittain epäselvä, mistä johtuen sitä on tulkittu joiltakin osin hieman eri tavoin eri sukututkimusohjelmistoissa. Tämä saattaa aiheuttaa ongelmia tiedon siirrettävyydessä tai muussa käsittelyssä, mikä havaittiinkin tämän tutkimuksen empiirisessä osassa. Kohdatuista ongelmista kerrotaan tarkemmin luvussa 5.3.

Myös GEDCOM-standardin tietomallia on kritisoitu muun muassa siitä, että se pohjautuu liiaksi vanhanaikaiseen käsitykseen perheestä ja avioliitosta, ja näin ollen epätavallisempien perherakenteiden kuvaaminen saattaa olla ongelmallista. Esimerkiksi sellaista perhettä, jossa lapsen toinen vanhempi on tuntematon, saattaa olla hankala mallintaa [Sippu, 2000].

4.2.2 XML-pohjaiset standardiehdotukset

GEDCOM-standardi on kehittynyt nykyiseen muotoonsa 15 vuoden aikana. Sukutiedon tekniikka, käyttö ja ymmärtäminen ovat kuitenkin muuttuneet paljon tuona aikana ja muuttuvat yhä, ja siksi onkin yritetty kehittää ehdotuksia uudeksi standardiksi sukutiedon kuvaamiseen. Uudet määritykset pohjautuvat XML-kieleen, josta kerrottiin luvussa 3.1. Näissä XML-sovelluksissa on määritelty joukko sukutietoa kuvaavia tietotyyppejä sekä eri tietotyyppien välisiä mahdollisia suhteita. Tietotyyppejä voivat olla esimerkiksi henkilö, tapahtuma, päivämäärä tai paikka. XML-sovelluksen määrittelyn jälkeen sukututkimusohjelmistot voivat sitten mallintaa sukutietoa sovelluksen mukaisesti. XML-pohjaisia määrityksiä on kehitetty useampia.

Tärkeimpänä XML-määrityksenä voidaan pitää Myöhempien Aikojen Pyhien Jeesuksen Kristuksen Kirkon sukututkimusosaston vuonna 2001 julkaisemaa luonnosta GEDCOMin uudistetusta versiosta, GEDCOM XML:stä. *The GEDCOM XML Specification* [LDS Church, 2001] on sukutiedon käsittelyyn kehitetty XML-sovellus. Sen avulla pystyy siirtämään suurin piirtein samaa tietoa kuin perinteisellä GEDCOMilla, mutta XML-versio on puhtaampi ja selkeämpi kuin perinteinen. Joitakin vähäpätöisiä ja harvoin käytettyjä perinteisen standardin ominaisuuksia ei XML-versiossa enää ole. Vaikka GEDCOM XML eroaa perinteisestä GEDCOMista sekä syntaksiltaan että loogiselta rakenteeltaan, sitä ei kuitenkaan pidetä aivan uutena määrityksenä, vaan GEDCOMin kehittyneempänä muotona.

GEDCOM ja GEDCOM XML pystyvät molemmat välittämään sukutietoa, mutta XML-versiolla on merkittäviä etuja perinteiseen GEDCOMiin nähden. Ensinnäkin XML on laajasti tunnettu kieli. Perinteistä GEDCOMia käytettäessä täytyy ensin opetella sen erikoinen syntaksi, ennen kuin voi ymmärtää sen sisältöä. GEDCOMin XML-versio sen sijaan on jo alusta alkaen ymmärrettävää, mikäli tuntee XML-kieltä. Koska XML on jo laajalti hyväksytty ja käytössä, se tuo myös mukanaan helposti käytettävissä olevat hyötyohjelmat, työkalut ja rajapinnat. XML:ään liittyen on vakiintunut standardeja, jotka ratkaisevat monia sukututkimusohjelmistoihin liittyviä ongelmia kuten kansainvälisten merkistöjen käsittelyyn liittyvät hankaluudet.

GEDCOM XML pohjautuu samaan vanhanaikaiseen tietomalliin kuin perintei-

nen GEDCOM, minkä vuoksi se ei saanut kehitysyhteisöltä kovin positiivista vastaanottoa. Määrityksestä ei olekaan ilmestynyt vuoden 2001 julkaistun luonnoksen jälkeen uudempaa versiota, ja vaikuttaa siltä, että uuden standardin kehittely olisi ainakin toistaiseksi pysähdyksissä, tai jopa täysin keskeytetty [Sugimoto, 2006].

GEDCOM XML:n lisäksi on kehitetty useita muitakin XML-pohjaisia määrityksiä [Cover, 2005]. Niistäkin suurin osa perustuu tietomalliltaan perinteiseen GEDCOM-standardiin [Sugimoto, 2006]. Mikään määrityksistä ei ole kuitenkaan ehtinyt ainakaan vielä vakiintua sukutiedon uudeksi siirtostandardiksi, joten nykyisistä ohjelmistoista suurin osa käyttää edelleen perinteistä GEDCOM-standardia.

4.3 Sukututkimusohjelmistot

Sukututkimusohjelmistot ovat pääasiassa sukututkijoiden käyttöön tarkoitettuja ohjelmistoja, joiden avulla voidaan ylläpitää tietokantaa tutkituista henkilöistä, perheistä ja suvuista. Varsinaisessa tutkimuksen teossa ohjelmistoilla on melko pieni rooli, sillä tutkimus tapahtuu nykyäänkin pääasiassa erilaisia arkistolähteitä tutkimalla. Ohjelmistot ovat kuitenkin tutkimuksessa tärkeä apuväline, sillä niiden avulla sukutiedot saadaan helposti järjestettyä ja laitettua talteen. Tallennetuista tiedoista voidaan tuottaa erilaisia yhteenvetoja esimerkiksi tekstimuotoisten raporttien ja graafisten kaavioiden muodossa. Nämä auttavat näkemään sukututkimuksessa saavutetut tulokset havainnollisessa muodossa, kuten myös huomaamaan esimerkiksi tiedoista puuttuvat sukuhaarat.

Sukutieto on mallinnettava sukututkimusohjelmistossa siten, että tieto välittyy käyttäjälleen mahdollisimman täydellisenä ja vääristymättömänä. Tyypillisen ohjelmiston tietokanta on suunniteltu ensi sijassa johtopäätösten eli sukututkimuksen tulosten esittämiseen. Keskeisimpiä johtopäätöksiä ovat lähdemaininnoista sukutieteellisen päättelyn kautta kootut henkilöt ja perheet sekä niihin liitetyt ominaisuudet [Sippu, 2000].

Sukututkimusohjelmiston tiedonkäsittelytoiminnot voidaan jakaa karkeasti päivitys-, selailu- ja raportointitoimintoihin. Kaikki sukututkimusohjelmistot sisältävät nämä kaikki toiminnot jossain muodossa, mutta niiden tarkempi muoto ja käynnistystapa riippuvat ohjelmistossa sovelletusta tietomallista ja käyttöliittymästä [Sippu, 2000].

Päivitystoimintoja ovat uuden henkilön, perheen tai lähdetiedon syöttäminen tai näihin liittyvien ominaisuuksien myöhemmin tapahtuva lisäys, muutos tai poisto. Ohjelmistoihin tallennetaan ensin yksittäisiä henkilöitä, jotka sitten yhdistetään toisiinsa lapsina, vanhempina tai puolisoina, ja näin muodostuu perheitä ja sukuja.

Päivitystä on myös henkilön irrottaminen perheestä, lähteen kytkeminen henkilön tai perheen ominaisuuteen, lähdekytköksen irrottaminen sekä henkilön, perheen tai lähdetiedon poisto. Useimmissa ohjelmistoissa näitä tietoja muokataan graafisen, pääosin lomakepohjaisen käyttöliittymän välityksellä [Sippu, 2000].

Mainittujen perustoimintojen lisäksi välttämättömiä päivitystoimintoja ovat kahden henkilön yhdistäminen yhdeksi henkilöksi ja kahden perheen yhdistäminen yhdeksi perheeksi. Näitä toimintoja tarvitaan tietokantoja yhdistettäessä tai kun tutkija havaitsee päätyneensä samaan henkilöön eri teitä. Näin voi tapahtua, mikäli henkilön aiemmin tallennetut tiedot ovat eronneet siinä määrin myöhemmin tallennetuista, ettei henkilöiden identtisyys ole paljastunut tietoja uudestaan syötettäessä. Yhdistämisen käänteistoimintoa eli jakamista tarvitaan silloin, kun tutkija havaitsee sekoittaneensa kahden tai useamman eri henkilön tai perheen tietoja samaan tietueeseen [Sippu, 2000].

GEDCOM-tiedoston lataus eli tietojen tuonti ohjelmistoon on päivitystoimenpide, joka kerralla sijoittaa tietokantaan joukon uutta tietoa. Ohjelmisto luo kullekin tietokantaan tuotavalle GEDCOM-tietueelle uuden tunnisteen, joka ei ennestään ole tietokannassa käytössä. Mikäli siirto tehdään tietokantaan, joka sisältää jo ennestään tietoa, tuonti synnyttää yleensä tarpeen yhdistää tietokannan entisiä tietueita sinne tuotujen uusien kanssa [Sippu, 2000].

Selailutoimintoihin kuuluu esimerkiksi perushaku, jossa henkilö tai henkilöjoukko haetaan esille jonkin ominaisuuden kuten nimen tai asuinpaikan perusteella. Selailutoimintoihin kuuluu myös sukulaissuhteen perusteella eteneminen, jolloin esillä olevan henkilön isä, äiti, aviopuolisot tai lapset haetaan esille. Joissakin ohjelmistoissa haun tuloksena saatua henkilö- tai perhejoukkoa voidaan edelleen karsia, tulosjoukko voidaan väliaikaisesti nimetä myöhempää käsittelyä varten, ja joukosta voidaan poimia yksittäisiä henkilöitä tai perheitä tarkempaan tarkasteluun tai päivitettäväksi. Lähdetieto voidaan hakea esille joko lähteen ominaisuuden perusteella tai seuraamalla kytköstä henkilöstä tai perheestä [Sippu, 2000].

Raportointitoiminnot tekevät tietynlaisen tekstimuotoisen tai graafisen koosteen sukutietokannasta. Jokaiseen sukututkimusohjelmistoon olennaisesti kuuluvia raportointitoimintoja ovat esimerkiksi kokonaisten esi- ja jälkipolvitaulustojen tulostukset. Esipolvitaulustossa esitetään lähtöhenkilö sekä hänen vanhempansa ja esivanhempansa, kun taas jälkipolvitaulustossa esitetään lähtöhenkilö puolisoineen ja lapsineen. Julkaistavaksi tarkoitettujen raporttien tuottaminen vaatii ohjelmiston yhteentoimivuutta jonkin tekstinkäsittelyohjelman kanssa. Tietokannasta tuotetaan silloin tekstinkäsittelyohjelman komentoja ja sukutaulujen tekstiä sisältävä tiedosto, joka muotoiluohjelman läpi ajettuna muuntuu tulostuskelpoiseksi. Joitakin su-

kututkimusohjelmistoja on jopa kehitetty erityisesti sukukirjan tekoa varten. Pelkästään sukututkimusprosessin avustamiseen tarkoitetut raportit tuotetaan sitä vastoin puhtaaseen tekstimuotoon, jota on vaivatonta selailla myös tietokoneella minkä tahansa ohjelman alaisuudessa [Sippu, 2000].

Raportointitoimintoihin kuuluu myös tietojen vedostus eli vienti GEDCOM-tiedostoon tutkijan määräämästä tietokannan osasta. Muodostettavan GEDCOM-tiedoston tulee luonnollisesti aina olla sellainen, että sen kaikkien tietueiden kytkinkenttien tulee viitata tiedoston sisään. Ohjelmiston tulee siis poistaa henkilö- ja perhetietueiden vedoksista kaikki kytkökset tietueisiin, joita ei vedosteta [Sippu, 2000].

Sukututkimusohjelmistoja on tarjolla lukuisia, niin kotimaisia kuin ulkomaisiakin. Kotimaisissa ohjelmistoissa etuna on ainakin se, ettei niissä ole ongelmia ä- ja ö-kirjainten kanssa, kuten ulkomaisissa saattaa olla [Palander, 1999, s. 31]. Tosin monissa ulkomaisissakin ohjelmistoissa aakkostoon on kiinnitetty huomiota, ja niistä saattaa täten löytyä tuki myös skandinaavisille kirjaimille. Joihinkin ulkomaisiin ohjelmistoihin on myös kehitetty laajennoksia eri kielille, jolloin käyttöliittymän voi saada esimerkiksi suomenkielisenä. Näin ollen ulkomaiset ohjelmistot ovat sukututkijalle varteenotettava vaihtoehto kotimaisten ohjelmistojen rinnalla. Suurin osa ohjelmistoista on maksullisia, mutta esimerkiksi GEDCOM-standardin kehittäneen Myöhempien Aikojen Pyhien Jeesuksen Kristuksen Kirkon tekemä Personal Ancestral File (PAF) -ohjelmisto [LDS Church, 2006] on ilmainen.

Eri sukututkimusohjelmistoissa on paljon samoja perustoimintoja, mutta suuria eroavaisuuksiakin ohjelmistojen väliltä löytyy. Riippuu paljolti tutkijan tarpeista, millainen ohjelmisto hänelle parhaiten sopii. Erottavia tekijöitä ohjelmistojen välillä ovat muun muassa käytön helppous, talletettavien henkilöiden mahdollinen enimmäislukumäärä, kuvan, äänen ja videokuvan tallennusmahdollisuus, GEDCOM-yhteensopivuus, hakuominaisuudet tietokannasta, erilaiset raportointimahdollisuudet sekä mahdollisuus suunnitella omia tulosteita. Esimerkiksi mahdollisuudet tulostaa tiedostoon vaihtelevat jonkin verran, ja kaikki ohjelmistot eivät tue esimerkiksi sellaisia tärkeitä tulostusmuotoja kuin RTF (*Rich Text Format*) ja HTML. Lisäksi huomionarvoista on, että joidenkin ohjelmistojen tekijöillä on todellista kiinnostusta ohjelmiston jatkuvaan kehittämiseen, ja he saattavat olla aktiivisessa vuorovaikutuksessa ohjelmiston käyttäjien kanssa. Tällaisessa tapauksessa ohjelmiston mahdolliset puutteet tai viat voivat korjaantua melko nopeastikin [Palander, 1999, s. 31–33].

Perinteisten ohjelmistojen lisäksi nykyään löytyy yhä enenevässä määrin myös WWW-pohjaisia sukututkimusohjelmistoja. Niissä on usein samat perusominaisuudet kuin perinteisissäkin ohjelmistoissa, ja lisäksi etuna on muun muassa se, että

tiedot ovat suoraan WWW:ssä toisten tutkijoiden ulottuvilla. Esimerkkinä WWW-pohjaisesta ohjelmistosta mainittakoon PhpGedView [PhpGedView, 2007], joka on avoimen lähdekoodin ohjelmisto.

Tarkastellaan seuraavaksi neljää suomalaista, perinteistä sukututkimusohjelmistoa hieman tarkemmin. Kyseessä ovat ohjelmistot Genus [Citius Solutions Oy, 2007], Juuret [Consegur Oy, 2007], SukuJutut [Luhtasaari, 2007] ja Sukuohjelmisto [Kaila, 2007]. Nämä ovat Windows-käyttöjärjestelmään sopivia, mutta muihinkin käyttöjärjestelmiin on olemassa sukututkimusohjelmistoja.

Kaikissa neljässä ohjelmistossa on runsaasti ominaisuuksia, ja melko selviä erojakin niiden väliltä löytyy. Käyttöliittymä on kussakin ohjelmistossa luonnollisestikin erilainen, ja näin ollen käytön helppous vaihtelee jonkin verran. Toiminnallisia eroja on muun muassa tulosteiden ja kaavioiden monipuolisuudessa sekä mahdollisuudessa tallentaa ääni- ja videotiedostoja. Lisäksi jotkut näistä sisältävät hieman erikoisempia lisäominaisuuksia. Kaikki neljä ohjelmistoa tukevat GEDCOM-standardin mukaista tiedonsiirtoa [Palander, 1999, s. 32–33].

Seuraaviin taulukoihin on koottu joitakin näiden ohjelmistojen ominaisuuksia ohjelmistokohtaisesti. Ominaisuuksista on mainittu esimerkinomaisesti lähinnä joitakin tärkeimpiä tai erikoisempia, ja tiedot perustuvat pääosin ohjelmistotoimittajien ilmoittamiin tietoihin. Ominaisuudet on jaettu kunkin ohjelmiston osalta neljään alakohtaan. Kolme ensimmäistä kohtaa, ”Päivitystoiminnot”, ”Selailutoiminnot” ja ”Raportointitoiminnot”, noudattaa edellä kuvattua tiedonkäsittelytoimintojen jakoa, ja neljännessä kohdassa esitellään muita ominaisuuksia.

Genus-ohjelmistosta on viimeisimpänä ilmestynyt versio Genus Senior 2.24. Ohjelmistoa koskevat tiedot perustuvat ohjelmistotoimittajan [Citius Solutions Oy, 2007] sekä Suomen Sukututkimusseuran [Suomen Sukututkimusseura, 2005] ilmoittamiin tietoihin.

Taulukko 4.1: Genus-ohjelmiston ominaisuuksia.

Päivitys-toiminnot	Perustietojen lisäksi henkilölle voi tallentaa rajattoman määrän vapaamuotoista tekstiä omilla otsikoillaan. Tietoa voidaan tallentaa numeerisiin ja loogisiin kenttiin, ja lisäksi on mahdollista tallentaa kuvaa ja ääntä sekä liittää lähdetietoja henkilön perustietoihin ja vapaamuotoiseen tekstiin. Henkilöille voidaan tallentaa myös osoitetiedot, jolloin Genus voi toimia sukuseuran jäsenrekisteriohjelmana.
Selailu-toiminnot	Tietojen haku on mahdollista henkilön kaikilla tiedoilla ja niiden yhdistelmillä. Tiedoista voidaan selvittää kahden henkilön välinen sukulaisuussuhde ja lähimmät yhteiset esivanhemmat sekä hakea lähisukulaisia, esivanhemmat ja jälkeläiset. Graafisesti esitetystä sukupuusta käsin voidaan valita eri henkilöitä ja tutkia heidän tietojaan.
Raportointi-toiminnot	Graafisesti voidaan esittää perhetaulu, esivanhemmat ja jälkeläiset. Tietoja voidaan tulostaa eri hakutulosten mukaisesti. Tulokset voi esittää myös vieraalla kielellä sanaston avulla.
Muita ominaisuuksia	Eri tietoja voi suojata salasanalla, määrittämällä esimerkiksi koko ohjelmiston käytön tai tiettyjen tiedostojen ja lähdetietojen katselun salaiseksi. Ohjelmistosta on saatavana 250 henkilön, 1000 henkilön tai rajattoman henkilömäärän versio. Ohjelmisto sisältää vapaasti muokattavan historiankirjan, johon on kirjattu yli 150 tärkeää historiantapahtumaa.

Juuret-ohjelmistosta on viimeisimpänä ilmestynyt versio Juuret 2.1. Ohjelmistoa koskevat tiedot perustuvat ohjelmistotoimittajan [Consegur Oy, 2007] sekä Suomen Sukututkimusseuran [Suomen Sukututkimusseura, 2005] ilmoittamiin tietoihin.

Taulukko 4.2: Juuret-ohjelmiston ominaisuuksia.

Päivitys-toiminnot	Perustietojen lisäksi henkilölle voi tallentaa rajattoman määrän vapaamuotoista tekstiä, ja myös kuvan, äänen tai videokuvan tallentaminen on mahdollista. Henkilön syntymä-, kaste-, vihki- ja kuolinpaikkaan liittyvät tiedot sekä muuttotiedot voidaan syöttää talokohtaisesti.
Selailu-toiminnot	Haun avulla pystyy analysoimaan sukujen henkilösuhteita ja hakemaan yksittäisiä sukuryppäitä. Hakutapana toimii graafinen sukupuu, jossa voidaan edetä joko vanhempien, lasten, sisarusten tai puolisoiden suuntaan. Analyysitoimintojen avulla voidaan laskea tilastotietoja sukuhaarasta tai koko aineistosta. Henkilön sukulaiset voidaan eritellä siten, että etsitään esimerkiksi esivanhemmat n:ssä polvessa, serkut, pikkuserkut, kälyt, miniät tai langot.
Raportointi-toiminnot	Esi- ja jälkipolvista pystyy tekemään monenlaisia perusraportteja ja -kaavioita sekä tulostamaan sukupuita. Sukupuun vieressä voidaan lisäksi tarkastella henkilöiden elinaikaan liittyviä historian tapahtumia ja verrata näin suvun tapahtumia Suomen ja maailman tapahtumiin. Talo-tulosteessa on eriteltynä kunkin talon perustietojen lisäksi talon historiaan liittyvät tapahtumat ajanmukaisessa järjestyksessä. Ohjelmisto sisältää myös kattavan luettelon seurakuntien ja muiden henkilörekistereitä pitävien tahojen osoitteista. Osan tiedoista voi tallentaa RTF- tai HTML-muodossa.
Muita ominaisuuksia	Ohjelmisto sisältää runsaasti erilaisia sukututkijalle tärkeitä aputietoja, kuten tiedot saatavilla olevista mikrofilmeistä, kylähakemiston, sukututkimuksen monikielisen sanaston, karttoja, tietoja sukuseuroista ja sukututkimukseen liittyvästä kirjallisuudesta sekä kalenterin.

SukuJutut-ohjelmistosta on viimeisimpänä ilmestynyt versio SukuJutut 11.6. Ohjelmistoa koskevat tiedot perustuvat ohjelmistotoimittajan [Luhtasaari, 2007] sekä Suomen Sukututkimusseuran [Suomen Sukututkimusseura, 2005] ilmoittamiin tietoihin.

Taulukko 4.3: SukuJutut-ohjelmiston ominaisuuksia.

Päivitys-toiminnot	Perustietojen lisäksi henkilölle voidaan tallentaa useita kuva-, ääni- ja videotiedostoja. Taloihin liittyvää tietoa voi syöttää monipuolisesti, ja taloja ja henkilöitä voi liittää toisiinsa. Huomautuksia ja lähdetietoja voi syöttää lähes jokaiseen tietokenttään. Lisätietoa voi kirjoittaa runsaasti sekä henkilö- että perhekohtaisesti.
Selailu-toiminnot	Henkilöitä voi etsiä lähes kaiken tallennetun tiedon avulla. Ohjelmiston avulla on mahdollista tehdä raportti kahden eri suvun eroavaisuuksista sekä kahden suvun välisistä avioliitoista. Tuloksia voi katsella useasta erilaisesta näytöstä. Ohjelmisto näyttää tietoja muun muassa sukupuun, ympyränmuotoisen esivanhempien taulun tai aikajanan muodossa. Jälkipolvia voi tarkastella esimerkiksi puurakenteena valokuvien kera.
Raportointi-toiminnot	Tiedoista voidaan tulostaa esimerkiksi esi- ja jälkipolvet erilaisina kaavioina tai tauluina, sukupuu, sukukirja, samana päivänä syntyneet, irralliset henkilöt ja perheet, henkilölistaus, ikäjakauma, osoite- ja puhelintiedot tai syntymäpäivät. Osan tiedoista voi tallentaa RTF- tai HTML-muodossa.
Muita ominaisuuksia	Ohjelmistoa on kehitetty erityisesti sukukirjan tekoa varten, ja sukukirja voidaan tulostaa joko sukuhaaroittain tai sukupolvittain. Monikieliset sukukirjat sekä sukukirjan täydennysosan tekeminen on myös mahdollista.

Sukuohjelmisto-nimisestä ohjelmistosta on viimeisimpänä ilmestynyt versio Sukuohjelmisto 2004. Ohjelmistoa koskevat tiedot perustuvat ohjelmistotoimittajan [Kaila, 2007] sekä Suomen Sukututkimusseuran [Suomen Sukututkimusseura, 2005] ilmoittamiin tietoihin.

Taulukko 4.4: Sukuohjelmisto-ohjelmiston ominaisuuksia.

Päivitys-toiminnot	Perustietojen lisäksi henkilölle voi tallentaa rajattoman määrän vapaata tekstiä. Henkilö- tai tietomäärän suhteen ohjelmistossa ei ole rajoituksia. Myös muiden kohteiden kuin henkilöiden lisääminen tietokantaan on mahdollista, esimerkiksi kouluihin ja niiden oppilaisiin liittyen voi tallentaa tietoa.
Selailu-toiminnot	Ohjelmisto mahdollistaa yksinkertaisten, mutta laajojenkin graafisten esitysten tekemisen erilaisina sukupuurakenteina.
Raportointi-toiminnot	Tiedoista voidaan tulostaa esimerkiksi esipolvitauluja, jälkipolvitauluja, graafisia sukupuita sekä henkilö- ja paikkakuntahakemistoja. Tekstiraportit pystytään muuntamaan Word-tekstinkäsittelyohjelman ymmärtämään muotoon, ja myös HTML-muotoisten raporttien tekeminen on mahdollista.
Muita ominaisuuksia	Ohjelmiston tietokantana toimii Microsoft Access -relaatiotietokanta. Mikäli käyttäjällä on Access-ohjelma, pystyy hän tutkimaan sillä sukutietokantaa. Ohjelmiston mukana on apuohjelma, jolla voi tehdä XML-kielisen varmuuskopion.

5 Semanttisten tekniikoiden soveltaminen sukututkimuksessa

Tässä luvussa selvitetään, kuinka semanttisen webin tekniikat soveltuvat sukutiedon esittämiseen ja käsittelyyn. Luvussa 5.1 kerrotaan sukutiedon semanttisesta esityksestä yleisesti, ja luvussa 5.2 esitellään tässä tutkimuksessa käytettävä sukuontologia. Luvussa 5.3 kuvataan, kuinka olemassa oleva sukutieto saadaan muunnettua semanttiseen muotoon muunnosohjelmaa käyttäen, ja luvussa 5.4 tarkastellaan muunnetun sukutiedon esittämistä Protégé-ontologiaeditorilla. Luvussa 5.5 kerrotaan tutkimuksen aikana kohdatuista haasteista, ja lopuksi luvussa 5.6 arvioidaan tutkimuksessa käytettyjen menetelmien ja työkalujen sopivuutta sukutiedon käsittelyssä.

5.1 Sukutiedon esittäminen semanttisesti

Semanttisen webin hyödyntämistä sukututkimuksessa on tutkittu viime vuosina jonkin verran. Tutkimuksessa on keskitytty muun muassa erilaisten sukuontologioiden kehittämiseen [Dean, 2002] [Zandhuis, 2005], sukutiedon muuntamiseen vanhasta GEDCOM-muodosta semanttiseen muotoon [Dean, 2003] [Askren, 2007] sekä sukutiedon automaattiseen hakuun WWW:stä semanttisin keinoin [Walker, 2004].

Semanttiseen kieleen siirtyminen sukutiedon esittämisessä tuo mukanaan paljon uusia mahdollisuuksia verrattuna perinteiseen GEDCOM-muotoiseen tai uudempaan XML-muotoiseen esitystapaan. Sukututkimuksen tekeminen voisikin tämän myötä helpottua huomattavasti. Semanttisilla tekniikoilla voitaisiin esimerkiksi hakea sukutietoa sisältäviltä WWW-sivuilta helposti tietoa omasta suvusta ja löytää puuttuvia sukulinjoja, tehdä automaattisesti erilaisia johtopäätöksiä sukutiedosta tai vaikkapa yhdistää sukutieto johonkin muuhun, aiheeseen liittyvään ontologiaan, jolloin voitaisiin yhdistää eri tutkimusalojen tietoja toisiinsa.

Jotta sukutietoa voitaisiin esittää semanttisessa muodossa, täytyy ensin olla määriteltynä ontologia, jonka rakennetta itse sukutieto noudattelee. Sukuontologia kuvaa sukutiedon käsitteitä ja niiden välisiä mahdollisia suhteita. Sukutieto sopii luonnollisesti esitettäväksi ontologian avulla, sillä se on tyypiltään hierarkkista ja esitetäänkin yleensä puumallin avulla. Ontologiakirjastoista löytyy joitakin sukutiedon

kuvaamiseen tarkoitettuja ontologioita, mutta sukutiedon esittämistä ja käsittelyä ontologiaeditorilla ei ole kuitenkaan tutkittu kovin paljon.

5.2 Sukuontologia

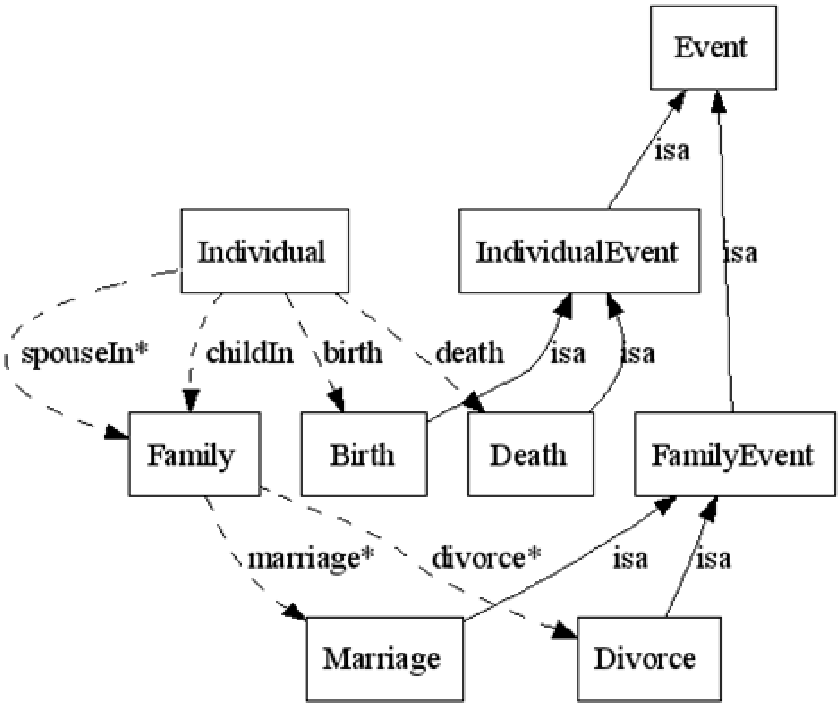
Sukutietoa kuvaavia ontologioita löytyy valmiina muutamia, ja tässä tutkimuksessa käytetäänkin erästä niistä. Sukutietoa voidaan esittää semanttisesti millaisen sukuontologian mukaisesti tahansa, mutta olemassa olevan GEDCOM-muotoisen sukutiedon muuntaminen semanttiseksi on helpompaa, mikäli ontologia noudattaa GEDCOMin tietomallia. Tähän tutkimukseen on valittu juuri tämäntyyppinen ontologia [Dean, 2002], joka kuvaa GEDCOMin tietomallia OWL-kielellä. Ontologia on tehty lähinnä kokeilukäyttöön, eikä se määrittele GEDCOM-standardia täydellisesti. Siinä on kuitenkin kuvattu olennaisimmat standardiin kuuluvat osat, joten se on täysin riittävä tämän tutkimuksen tarkoituksiin.

Kuvassa 5.1 on esitetty tässä tutkimuksessa käytettävän sukuontologian luokkien ja ominaisuuksien välisiä suhteita havainnollistava kaavio, joka on tehty Protégén OntoViz-lisäohjelmalla [Sintek, 2006]. Luokat on merkitty kaavioon suorakulmioilla, luokkien periytyvyys yhtenäisillä nuolilla ja luokkien välisiä suhteita kuvaavat ominaisuudet (engl. *object property*) katkoviivoitetuilla nuolilla.

Keskeisimpiä luokkia ontologiassa ovat *Individual* ja *Family*. Muita luokkia ovat tapahtumien yliluokka *Event* sekä sen aliluokat (*is a*) *IndividualEvent* ja *FamilyEvent* sekä edelleen näiden aliluokat *Birth* ja *Death* sekä *Divorce* ja *Marriage*. Henkilöön liittyviä ominaisuuksia ovat *birth*, *death*, *childIn* ja *spouseIn* ja perheeseen liittyviä *divorce* ja *marriage*. Tähdellä merkittyjä ominaisuuksia voi luokan yhdellä ilmentymällä olla useita, kun taas muita ominaisuuksia voi olla korkeintaan yksi.

Ontologiassa on edellä kuvattujen, eri luokkia yhdistävien ominaisuuksien lisäksi myös niin kutsuttuja tietotyyppi-ominaisuuksia (engl. *datatype property*). Ne saavat arvoikseen esimerkiksi teksti- tai lukuarvoja toisin kuin edellä mainitut ominaisuudet, jotka liittyvät toisten luokkien ilmentymiin. Kuvassa 5.2 näkyvät kaikki kuhunkin luokkaan liittyvät ominaisuudet sekä niiden arvojen sallitut tyypit.

Tästä kaaviosta nähdään, että edellisessä kaaviossa kuvattujen ominaisuuksien lisäksi ontologiassa on henkilöön (*Individual*) liittyvät ominaisuudet *givenName*, *surname*, *name*, *sex* ja *title* sekä tapahtumaan (*Event*) liittyvät ominaisuudet *date* ja *place*. Henkilöllä on siis useampia nimeen liittyviä ominaisuuksia. Tarkoituksena on, että nimen voi määrittää joko kokonaan ominaisuuden *name* avulla, tai etu- ja sukunimen erikseen ominaisuuksien *givenName* ja *surname* avulla, tai käyttää näitä molempia tapoja yhdessä. Ominaisuuden *title* avulla voi tarvittaessa kuvata henkilön so-



Kuva 5.1: GEDCOM-ontologian perusrakenne.

Family		
marriage	Instance*	Marriage
divorce	Instance*	Divorce

Individual		
givenName	String	
surname	String	
name	String	
sex	String	
title	String	
spouseIn	Instance*	Family
childIn	Instance	Family
birth	Instance	Birth
death	Instance	Death

Event	
date	String
place	String

Kuva 5.2: GEDCOM-ontologian kuhunkin luokkaan liittyvät ominaisuudet ja niiden sallitut tyypit.

siaalista asemaa tai esimerkiksi roolia kuningasperheessä. Ontologian kehittäjä oli käyttänyt sukutiedosta esimerkkinä Euroopan kuninkaallisia, joten se selittää kyseisen ominaisuuden käytön tässä ontologiassa [Dean, 2003].

GEDCOM-ontologia on OWL-kielinen. Siinä on käytetty luokkien määritykseen RDFS:n mukaista määrittelyä *rdfs:Class*, mikä on sallittua OWL Full -kielessä [Bechhofer ym., 2004, luku 3.1]. Protégé on kuitenkin ensisijaisesti tarkoitettu käytettäväksi siten, että OWL-ontologiaan ei sekoiteta RDFS-kielen rakenteita [Stanford Medical Informatics, 2007c], vaikka se itse OWL-kielen puolesta olisikin mahdollista. Tällaisen ontologian pystyy kylläkin tuomaan Protégéhen, mutta sen käsittelyssä voi olla hieman ongelmia. Sukuontologiakaan ei tämän vuoksi näy Protégéssa kovin hyvin, sillä esimerkiksi ilmentymien tietojen syöttösivulla Protégé ei tunnista automaattisesti ominaisuuksien tyyppejä. Ne täytyisi määrittää erikseen, jotta tietoja pääsisi syöttämään. Ongelma saadaan kuitenkin ratkaistua muuttamalla ontologia puhtaasti OWL-kieliseksi eli OWL DL -muotoon. Tämä onnistuu muuttamalla luokkamääritykset muodosta *rdfs:Class* muotoon *owl:Class*. Muutoksen jälkeen ontologia saadaan näkymään siististi Protégéssa.

Tämä ontologia on siis pelkistetty versio GEDCOM-standardista. Tässä tutkimuksessa ontologiaa täydennetään, mutta ei kuitenkaan kohti täydellisempää GEDCOM-standardia, vaan ontologiaan lisätään muita, hyödyllisiksi arvioituja ominaisuuksia. Ontologiaan tehtävät täydennykset kuvataan luvussa 5.4.

5.3 Sukutietokannan muuntaminen semanttiseksi

Sukutieto on nykyisin koottu pääasiassa ohjelmistoilla, joista tiedon saa ulos GEDCOM-muodossa. Sukutietokannan muuntaminen semanttiseksi vaatii sukutiedon muunnosta GEDCOMista jollekin semanttiselle kielelle. Muuntamiseen on tehty joitakin ohjelmia, ja erästä niistä hyödynnetäänkin tässä tutkimuksessa. Edellä kuvatun GEDCOM-sukuontologian kehittäjä on tehnyt myös GEDCOM to DAML -ohjelman [Dean, 2003], joka on alun perin tehty muuntamaan GEDCOM-muotoista tietoa DAML-kielelle. Ohjelma tekee muunnoksen GEDCOM-sukuontologiaan pohjautuen, joka on nykyisellään OWL-kielinen. Koska DAML on OWL-kielen edeltäjä ja siten rakenteeltaan samantyyppinen ja koska muunnos tehdään OWL-kieliseen ontologiaan pohjautuen, tapahtuu tiedon muunnos itse asiassa OWL-kielelle.

GEDCOM-sukuontologiaa tiedonmuunnoksessa käytettäessä perinteinen tietomalli pysyy samana, ja sukutieto vain esitetään eri kielellä. Näin ollen edellä, luvussa 4.2.1, mainitut tietomallin rakenteeseen liittyvät ongelmat pysyvät ennallaan. Sukuontologian lailla muunnosohjelmakin on tehty lähinnä kokeilukäyttöön, ja se

muuntaa OWL-muotoon vain tärkeimmät GEDCOM-standardiin kuuluvat osat. Ohjelma on kuitenkin sopiva työkalu tämän tutkimuksen tarkoituksiin, koska sen avulla sukutiedon tärkeimmät osat saadaan muutettua semanttisiksi.

Tutkimuksessa käytetään sukutiedosta esimerkkinä kuvitteellista sukua. Sukutiedot syötetään ensin Juuret-sukututkimusohjelmistoon ja tällä ohjelmistolla tiedoista tehdään GEDCOM-tiedosto. Sen jälkeen GEDCOM-tiedosto muunnetaan OWL-muotoon GEDCOM to DAML -muunnosohjelmalla. Muunnoksen onnistumiseksi tähän ohjelmaan pitää kuitenkin tehdä joitakin muutoksia, jotta se saadaan käsittelemään Juuret-ohjelmiston tuottamaa GEDCOM-tiedostoa oikein. Ongelmana on, että eri sukututkimusohjelmistoissa GEDCOM-standardia on tulkittu joiltakin osin eri tavoin, ja tässä tapauksessa GEDCOM-tiedosto on erilainen kuin mitä muunnosohjelma olettaa sen olevan.

Yhtenä ongelmana muunnoksessa on, että muunnosohjelma ei ymmärrä Juuret-ohjelmiston tuottamaa GEDCOM-tiedostoa avioeroon liittyvän DIV-tunnisteen kohdalla, ja ohjelman suoritus keskeytyy tämän vuoksi virheilmoitukseen. Tästä syystä muunnosohjelmaan täytyy tehdä pieni muutos, jotta se saadaan käsittelemään avioeroon liittyvä tunniste oikein.

Juuret-ohjelmiston tekemässä GEDCOM-tiedostossa perheen tiedot on esitetty seuraavasti:

```
0 @F10@ FAM
1 HUSB @I28@
1 WIFE @I27@
1 MARR
1 DIV
2 DATE 1986
1 CHIL @I29@
```

Tässä DIV-tunniste sekä sen jäljessä oleva ajankohdan kertova DATE-tunniste ja mahdollisesti paikan kertova PLAC-tunniste kertovat avioeron tapahtuneen. Mikäli avioeroa ei olisi tapahtunut, ei koko DIV-tunnistetta näytettäisi. Joidenkin ohjelmistojen tekemissä GEDCOM-tiedostoissa avioero kuitenkin esitetään siten, että DIV-tunniste merkitään aina, ja sen perässä on joko 'Y' tai 'N' sen mukaan, onko avioeroa tapahtunut (*Yes*) vai ei (*No*). GEDCOM-standardin mukaan avioero esitetään kuten Juuret-ohjelmiston tapauksessa, mutta mikäli tapahtuman ajankohtaa eikä paikkaa tiedetä, merkitään pelkkä DIV-tunniste ja annetaan sille arvoksi 'Y' [LDS Church, 1996].

Muunnosohjelman lähdekoodi näyttää DIV-tunnisteen käsittelyn osalta seuraavalta [Dean, 2003] :

```
else if (child.tag.equals("DIV") && child.value.equals("Y"))
{
  addEvent(model,
           uri,
           child,
           gedcom_200101.Divorce,
           gedcom_200101.divorce);
}
```

Tässä oletetaan standardin vastaisesti, että GEDCOM-tiedostosta löytyisi jokaisen avioeron kohdalta 'DIV Y'. Kun muunnosohjelma ei tätä löydä, jää DIV-tunniste kokonaan käsittelemättä, mikä sekoittaa ohjelman suorituksen. Tämä ongelma saadaan korjattua poistamalla *else if* -lauseen ehdosta Y:tä koskeva osa.

Toisena ongelmana muunnoksessa on, että naishenkilön nimi tulee OWL-tiedostoon väärin sellaisissa tapauksissa, joissa naisen sukunimi on muuttunut hänen mennessään naimisiin. Juuret-ohjelmistossa on oletuksena, että henkilön kaikki nimet tallennetaan GEDCOM-tunnisteeseen NAME. Tällöin GEDCOM-tiedostoon syntyy kaksi NAME-tunnistetta, joista ensimmäiseen tulee henkilön etunimet sekä alkuperäinen sukunimi ja toiseen uusi sukunimi. Muunnosohjelma ei kuitenkaan osaa näitä tällaisena käsitellä, vaan se tallentaa jälkimmäisen arvon edellisen päälle, jolloin nimeksi tulee pelkästään henkilön uusi sukunimi muiden nimien hävitessä kokonaan.

Muunnosohjelma olettaa nimen olevan ilmaistuna joko tunnisteiden NAME, GIVN (given name) ja SURN(ame) avulla

```
1 NAME Jonna Maria/Mäkinen/
2 GIVN Jonna Maria
2 SURN Mäkinen
```

tai pelkästään tunnisteella NAME ilmaistuna

```
1 NAME Jonna Maria/Mäkinen/
```

eikä siinä ole varsinaista avionimelle tarkoitettua kohtaa.

Juuret-ohjelmiston asetuksista pystyy määrittelemään minkänimisiin kahteen GEDCOM-tunnisteeseen nimitiedot tallennetaan. Ongelmana kuitenkin on, että se tekee molemmista 1-tason tunnisteita, joten vaikka toiseen valittaisiinkin NAMEn lisäksi vaikkapa SURN-tunniste, niin Juuret-ohjelmiston tekemä GEDCOM-tiedosto ei silti ole samanlainen kuin mitä muunnosohjelma olettaa.

Juuret-ohjelmistosta käsin dataa ei siis pysty saamaan välivaiheessa sellaiseksi, että henkilön nimi saataisiin oikein näkyviin muunnoksen jälkeen. Tästä syystä muunnosohjelmaan on tehtävä lisää muutoksia. Määritetään Juuret-ohjelmistossa nimet talletettaviksi tunnisteisiin NAME ja SURN, ja lisätään muunnosohjelmaan sellainen kohta, joka tarkistaa 1-tasolta myös tunnisteen SURN.

Nyt nimet voidaan siis tallentaa muunnosohjelman ymmärtämällä tavalla seuraavasti:

```
1 NAME Jonna Maria/Mäkinen/
1 SURN /Uusitalo/
```

Tämä ei ole virallisen GEDCOM-standardin mukainen esitystapa, mutta sen ansiosta nimi saadaan nyt kunnolla näkyviin myös OWL-tiedostossa.

Seuraavassa on ote Juuret-ohjelmiston tuottamasta GEDCOM-tiedostosta niin henkilön kuin perheenkin tietojen osalta:

```
0 @I1@ INDI
1 NAME Jonna Maria/Mäkinen/
1 SURN /Uusitalo/
1 SEX F
1 BIRT
2 DATE 15 APR 1970
2 PLAC Jyväskylä
1 FAMS @F28@
1 FAMC @F1@

...

0 @F1@ FAM
1 HUSB @I4@
1 WIFE @I3@
1 MARR
1 CHIL @I2@
1 CHIL @I1@
```

Muunnosohjelman lähdekoodiin tehtyjen muutosten jälkeen kaikki tallennettu sukutieto saadaan muunnettua onnistuneesti OWL-kielelle eli semanttiseen muotoon. Seuraavassa on ote muunnosohjelman tuottamasta OWL-tiedostosta, joka kuvaa edellisen esimerkin henkilöä ja perhettä:

```
<a:Individual rdf:about="#@I1@"
    a:name="Jonna Maria/Mäkinen/"
    a:sex="F"
    a:surname="/Uusitalo/">
    <a:spouseIn rdf:resource="#@F28@"/>
    <a:childIn rdf:resource="#@F1@"/>
    <a:birth rdf:resource="#event18"/>
</a:Individual>

...

<a:Birth rdf:about="#event18"
    a:date="15 APR 1970"
    a:place="Jyväskylä"/>

...

<a:Family rdf:about="#@F1@">
    <a:marriage rdf:resource="#event40"/>
</a:Family>
```

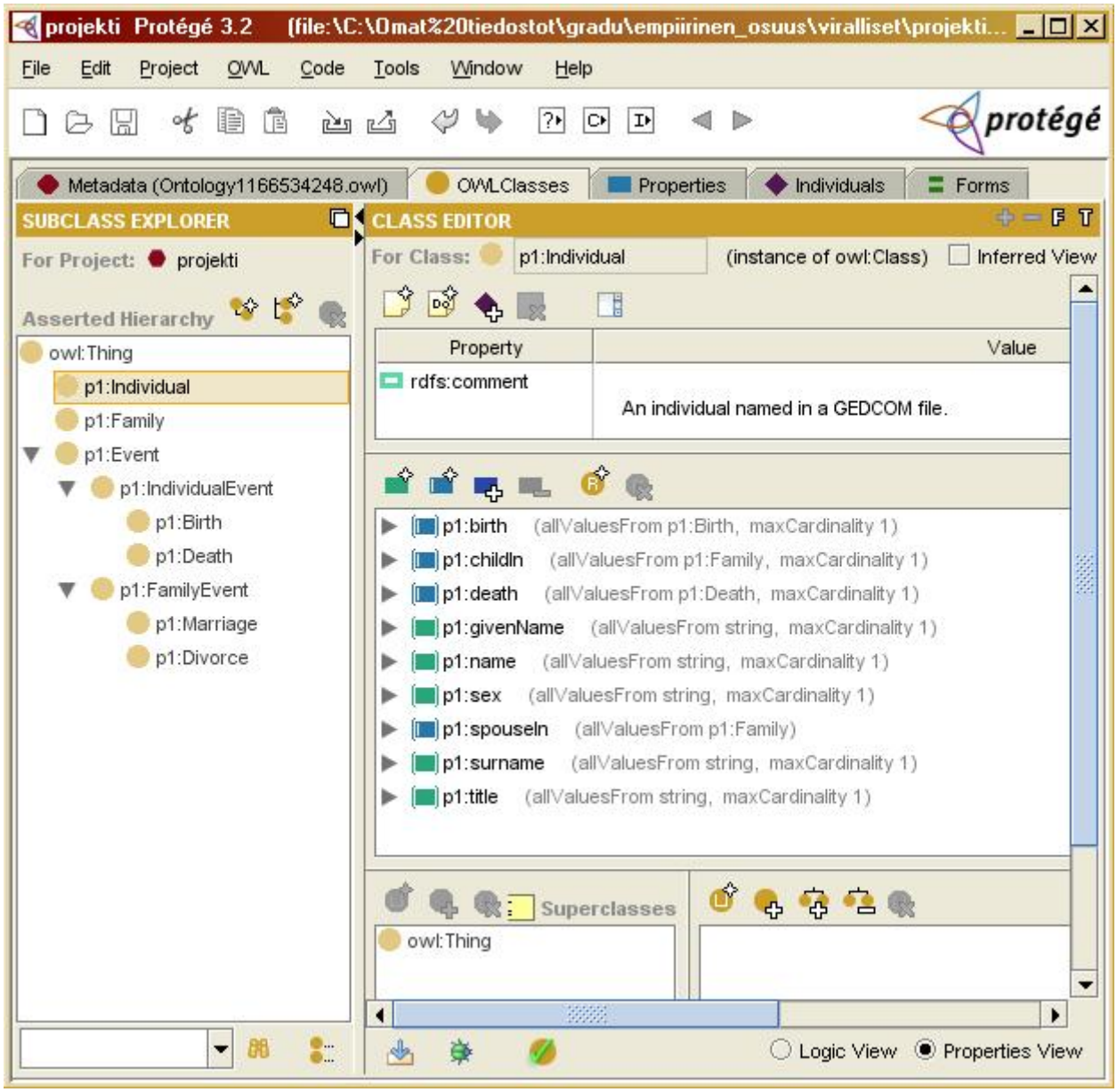
5.4 Sukutiedon esittäminen ja käsittely Protégéssa

Kun sukutieto on saatu muunnettua semanttiseen muotoon, voidaan se tuoda Protégé-ohjelmistoon. Tässä luvussa kuvataan, kuinka sukutieto saadaan näkyviin ja millaisia toimenpiteitä voidaan tehdä, jotta tiedot näkyisivät selkeämmin. Luvussa 5.4.1 esitellään lyhyesti Protégén käyttöliittymä, luvussa 5.4.2 kuvataan sukutiedon näkyvyyttä sekä toimenpiteitä näkyvyyden parantamiseksi, ja luvussa 5.4.3 tehdään lisäyksiä ontologiaan sekä käytetään päättelykonetta sukutiedon täydentämiseksi.

5.4.1 Protégén käyttöliittymä

Ontologian luokkien ja niiden ilmentymien käsittely tapahtuu Protégén graafisen käyttöliittymän avulla, jossa toiminnot on jaettu useille päällekkäisille välilehdille. Kuvassa 5.3 näkyy Protégén käyttöliittymä. Projektiin kuuluvien ontologioiden

hallinta (*Metadata*), luokkien määrittely (*OWL Classes*), ominaisuuksien määrittely (*Properties*), luokkien ilmentymien tietojen syöttäminen (*Individuals*) sekä tiedonsyöttölomakkeiden muokkaaminen (*Forms*) ovat kaikki Protégéssa omilla välilehdillään. Kuvan tilanteessa on juuri avoinna *OWL Classes* -välilehti, jossa näkyvät sukuontologian kaikki luokat vasemmassa reunassa ja *Individual*-luokkaan liittyvät ominaisuudet oikealla puolella.



Kuva 5.3: Näkymä Protégén käyttöliittymästä. Avoinna *OWL Classes* -välilehti.

OWL Classes -välilehdellä määritetään ontologian luokat ja luokkahierarkia, ominaisuudet ja niiden sallitut arvot, luokkien väliset suhteet sekä suhteisiin liittyvät ominaisuudet. *Properties*-välilehdellä ominaisuuksien määrityksiä voi tarkentaa. Ontologiaa rakennettaessa Protégé muodostaa samanaikaisesti oletuslomakkeita, jotka sisältävät kutakin luokkaa vastaavien ominaisuuksien arvojen syöttökentät. *Forms*-välilehdellä näitä lomakkeita voi muokata mallin pohjalta halutunlaiseksi muuttamalla esimerkiksi kenttien paikkoja lomakkeella, niiden kokoa tai muita ominaisuuksia. *Individuals*-välilehteä käytetään luokkien ilmentymien eli varsinaisen datan syöttöön.

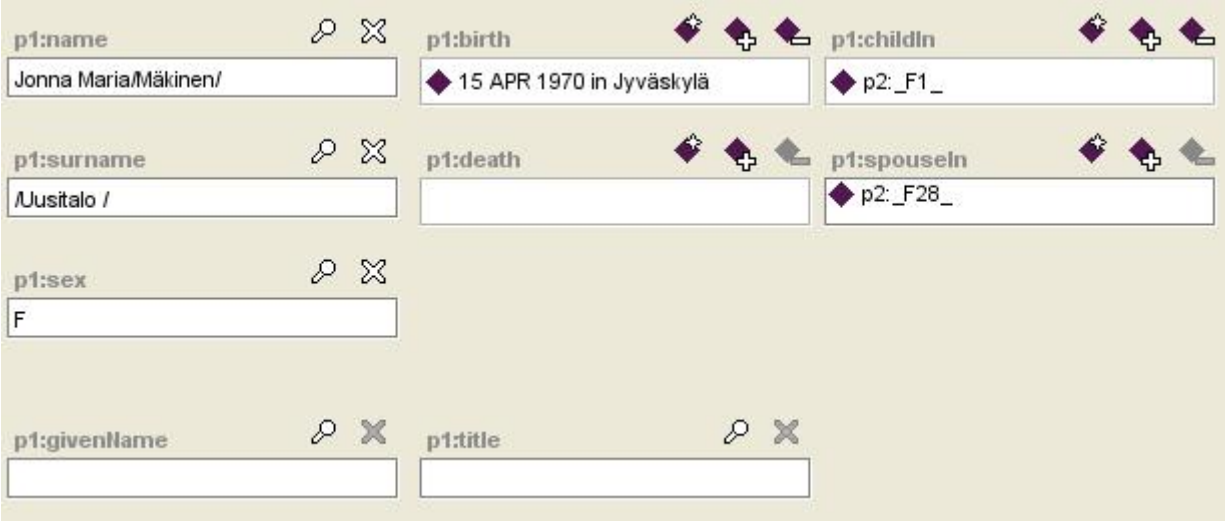
5.4.2 Sukutiedon näkymät

Kun sukutieto on muunnettu OWL-kielelle, pystytään tiedot saamaan näkyviin Protégéssa. Ongelmana kuitenkin on, että monimutkainen luokkarakenne hankaloittaa tietojen katselua. Henkilön tietoja katsellessa esimerkiksi syntymäaika (*birth*) ei näy suoraan, vaan sille varatussa kentässä näkyy ainoastaan kyseisen tapahtuman sisäinen koodi. Henkilön tietojen näkyminen on esitetty kuvassa 5.4.

Kuva 5.4: Henkilön tiedot ennen muutoksia.

Kullekin kentälle voidaan kuitenkin määritellä erikseen yksi tai useampikin ominaisuus, joiden arvot kentässä esitetään. Kun syntymäaikakenttään määritellään näytettäviksi arvoiksi päivämäärä ja paikka, ja kenttien keskinäistä järjestystä vaihdetaan selkeämmäksi, saadaan henkilön tiedot siistimmin näkyviin. Tietojen näkyminen muutosten jälkeen on esitetty kuvassa 5.5.

Nyt olemassa olevat tiedot on saatu hieman siistimmin näkyviin, mutta tietojen katseleminen on vieläkin melko hankalaa luokkarakenteesta johtuen. Esimerkiksi



Kuva 5.5: Henkilön tiedot ensimmäisten muutosten jälkeen.

perhesuhteita tarkasteltaessa henkilön tiedoissa näkyy vain sen perheen sisäinen koodi, jossa henkilö on lapsen (*childIn*) tai puolison (*spouseIn*) roolissa. Nämä koodit näkyvät kuvassa 5.5. Perheen puolelta tietoja tarkasteltaessa ei näe edes koodeista, keitä perheeseen kuuluu, vaan tiedoissa näkyvät ainoastaan tapahtumien avioliitto (*marriage*) ja avioero (*divorce*) sisäiset koodit. Nämä koodit näkyvät kuvassa 5.6. Näin ollen esimerkiksi tiettyyn perheeseen kuuluvien henkilöiden etsiminen on erittäin työlästä, koska perheeseen kuuluminen pitäisi tarkistaa erikseen jokaisen henkilön *childIn* tai *spouseIn* -kentästä.

Kuva 5.6: Perheen tiedot ennen muutoksia.

5.4.3 SWRL-päättelysäännöt

Tietojen näkyvyyttä pystytään parantamaan lisäämällä ontologiaan uusia ominaisuuksia ja antamalla niille arvot SWRL-päättelysääntöjen avulla. Lisätään ensin perheluokalle *Family* uudet ominaisuudet *hasChild* ja *hasSpouse*, joiden tarkoituksena on näyttää perheeseen kuuluvat lapset ja puolisot. Määritellään lisäksi sellaiset rajoitteet, jotka määrittävät uusien ominaisuuksien kohdetyypin olevan *Individual*. Uusien ominaisuuksien lisäämisen jälkeen muutetaan vielä perhesivun ulkoasua, jotta tiedot näkyisivät selkeämmin.

Seuraavaksi käytetään päättelykonetta, jotta saadaan asetettua kaikille henkilöille uudet ominaisuudet. Protégéhen on kehitetty SWRLTab-lisäohjelma [O'Connor, 2007], joka mahdollistaa SWRL-päättelysääntöjen muodostamisen. SWRLTab näkyy omalla välilehdellään, jonka saa näkyviin Protégén asetusten kautta. Ennen kuin päättelysääntöjä voidaan alkaa kirjoittaa, SWRL täytyy aktivoida. Tietämyskantaan otetaan sukuontologian rinnalle SWRL-ontologia, joka sisältää päättelysääntöjen kirjoittamiseen tarvittavia luokkia ja ominaisuuksia. Tallennusvaiheessa SWRL-säännöt tallentuvat tietämyskannan OWL-tiedostoon SWRL-ontologian mukaisina ilmentyminä.

SWRLTab mahdollistaa SWRL:n ja päättelykoneen välisen yhteentoimivuuden. Tällä hetkellä Protégé tukee SWRL-yhteyttä Jess-koneeseen [Friedman-Hill, 2006], joka on yksi yleisimmistä päättelykoneista. Tulevaisuudessa Protégéhen saatetaan kehittää yhteyksiä muihinkin SWRL:ää tukeviin päättelykoneisiin. Päättelysäännöt kirjoitetaan SWRLTab-välilehdellä, jonka jälkeen ne voidaan välittää Jessille. Jess muodostaa sääntöjen perusteella uutta dataa, jonka jälkeen datan voi tuoda tietämyskantaan. Käyttäjä hallitsee Protégén ja päättelykoneen välistä tapahtumasarjaa ja voi edetä siinä vaiheittain. Tapahtumasarja koostuu olemassa olevan datan ja SWRL-sääntöjen latauksesta Protégésta päättelykoneeseen, päättelykoneen suorittamasta sääntöjen tarkistuksesta ja toteuttamisesta, päättelytulosten tarkastelusta sekä päättelyn tuloksena syntyneen uuden tiedon tallennuksesta tietämyskantaan.

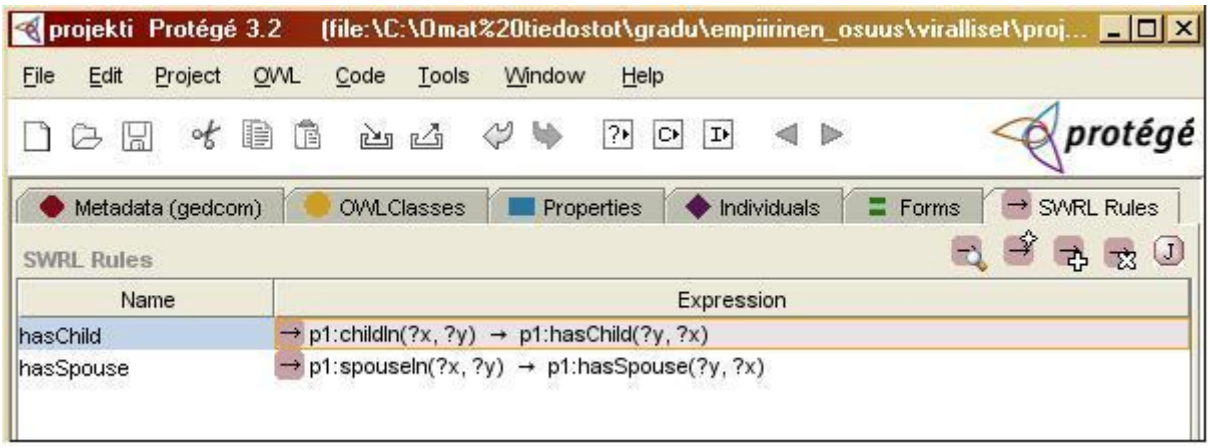
Kirjoitetaan nyt päättelysäännöt, joiden avulla henkilöille voidaan asettaa uudet ominaisuudet *hasChild* ja *hasSpouse*. Sanallisesti säännöt voidaan ilmaista seuraavasti:

```
Jos henkilö X on lapsi perheessä Y, niin silloin
perheessä Y on lapsi X.

Jos henkilö X on puoliso perheessä Y, niin silloin
perheessä Y on puoliso X.
```

SWRL-kielellä nämä säännöt kirjoitetaan jo olemassa olevien ominaisuuksien nimiä käyttäen seuraavasti:

```
childIn(?x,?y) → hasChild(?y,?x)

spouseIn(?x,?y) → hasSpouse(?y,?x)
```

Kuvassa 5.7 näkyy Protégén SWRLTab-välilehti, johon säännöt on kirjoitettu.



Kuva 5.7: SWRL-säännöt Protégéssa ominaisuuksien *hasChild* ja *hasSpouse* päättelemiseksi.

Kun säännöt välitetään Jess-päättelykoneelle ja liitetään sen tekemät johtopäätökset osaksi tietämyskantaa, saadaan uudet perhetiedot näkyviin kuvan 5.8 osoittamalla tavalla.

Kuva 5.8: Perheen tiedot muutosten jälkeen.

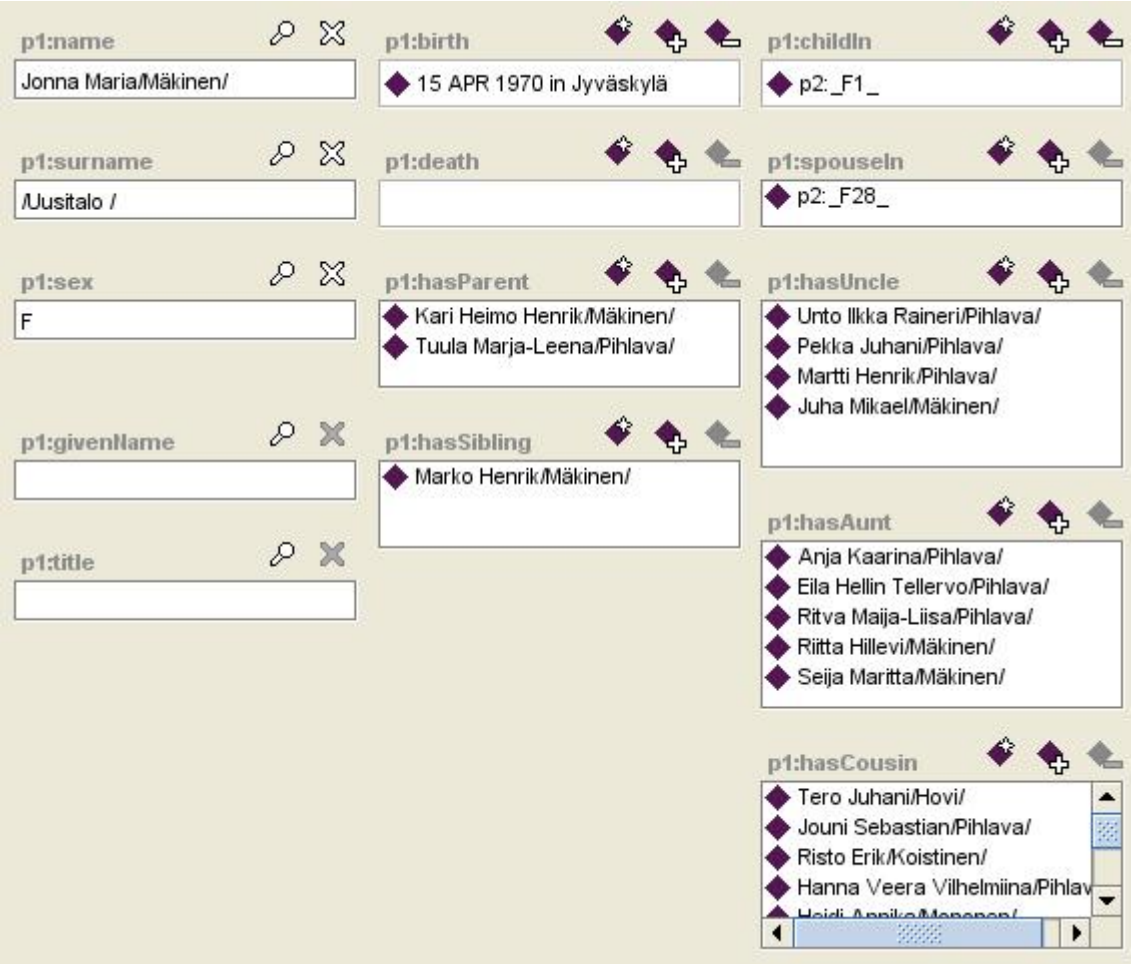
Tehdään seuraavaksi lisää päättelysääntöjä, jotka määrittävät henkilöiden sukulaisuussuhteita. Määritellään säännöt vanhempien, sisarusten, setien ja enojen, tätien sekä serkkujen päättelemiseksi. Nämä säännöt kirjoitetaan SWRL-kielellä seuraavasti:

```
childIn(?x,?y) ∧ hasSpouse(?y,?z) → hasParent(?x,?z)

hasParent(?x,?y) ∧ hasParent(?z,?y) ∧ differentFrom(?x,?z)
→ hasSibling(?x,?z) ∧ hasSibling(?z,?x)

hasParent(?x,?y) ∧ hasSibling(?y,?z) ∧ sex(?z,"M")
→ hasUncle(?x,?z)

hasParent(?x,?y) ∧ hasSibling(?y,?z) ∧ sex(?z,"F")
→ hasAunt(?x,?z)

hasParent(?x,?y) ∧ hasSibling(?y,?z) ∧ hasParent(?a,?z)
→ hasCousin(?x,?a)
```

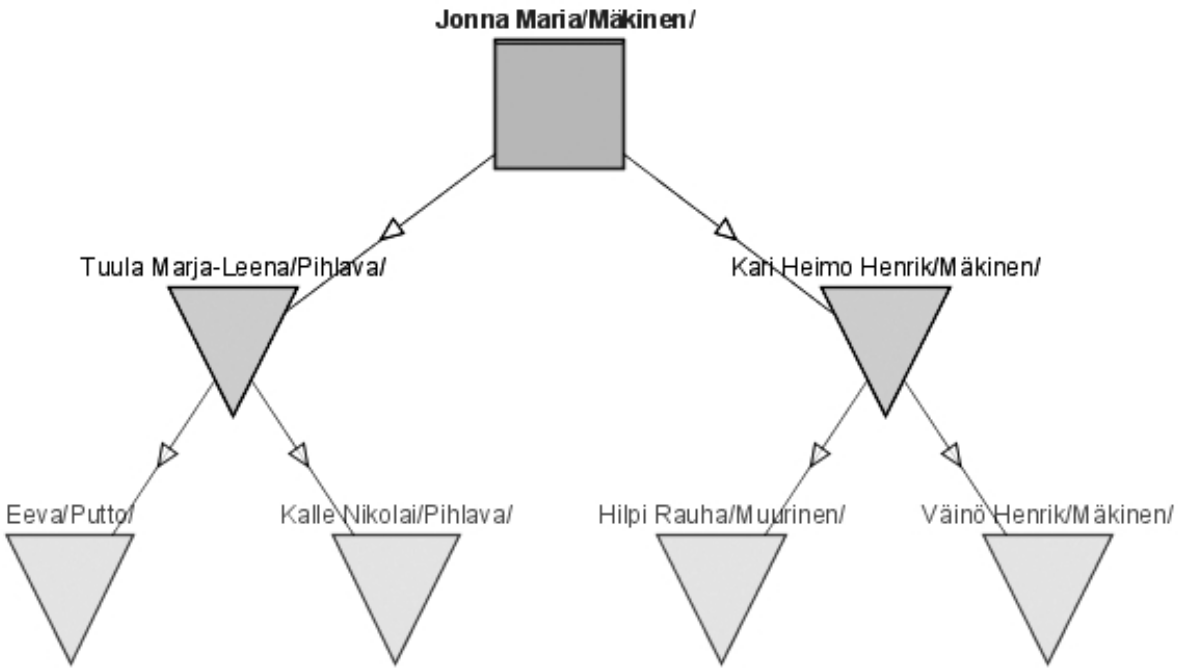
Vastaavasti voitaisiin määrittää mitä tahansa muitakin suhteita. Sisarusten päättelyyn tehdyssä säännössä käytetään *differentFrom*-muuttujaa, joka vertaa henkilöiden ilmentymiä toisiinsa ja tarkistaa, ovatko henkilöt samoja vai eivät. Jotta tämä vertailu saadaan toimimaan niin, että päättelykone tulkitsee kaikki tietämyskannan henkilöt eri yksilöiksi, täytyy ne Protégéssa vielä määritellä sellaisiksi OWL:n *AllDifferent*-rakenteella ennen päättelyn tekemistä.

Tämän määrityksen jälkeen voidaan tehdä jälleen päättely Jessillä. Kun uudet tiedot on saatu tietämyskantaan, näyttävät henkilötiedot kuvan 5.9 mukaisilta. Sukuontologia kaikkine tässä luvussa tehtyine lisäyksineen on kuvattu OWL-kielellä liitteessä A.

Sukutietoa pystyy selaamaan ja etsimään Protégéssa useilla eri tavoilla. Lisäksi sukutiedosta voi tehdä erilaisia graafisia esityksiä, esimerkiksi kuvan 5.10 mukaisen sukupuun, joka on tehty Protégén Jambalaya-lisäohjelmalla [CHISEL, 2007].



Kuva 5.9: Henkilön tiedot ja uudet sukulaisuussuhteet.



Kuva 5.10: Henkilön esivanhemmat sukupuuna.

5.5 Haasteita

Sukutiedon muuntaminen semanttiseen muotoon, sen siirtäminen Protégé-ohjelmistoon ja erilaisen toimintojen suorittaminen Protégéssa ei onnistunut kaikilta osin aivan niin helposti kuin mitä ensi alkuun olisi olettanut. Tässä luvussa kerrotaan joistakin haasteista ja ongelmista, joita eri vaiheiden aikana tuli vastaan.

Ensinnäkin edellä kuvatut muutokset sukuontologiaan ja sitä käyttävään muunnosohjelmaan vaativat jonkin verran lisätyötä. Protégén kyvyttömyys näyttää RDFS-määrittelyjä sisältävä OWL-ontologia kunnolla ihmetytti hieman, sillä Protégéa mainostetaan OWL-kieltä tukevana editorina, ja näin ollen sen olettaisi pystyvän käsittelemään ongelmitta kaikenlaisia OWL-ontologioita. Muutokset muunnosohjelmaan olivat sitä vastoin ehkä odotettavissakin, kun ottaa huomioon vaihtelevuuden GEDCOM-standardin tulkinnassa.

Protégén yleisessä käytössä oli hieman hankaluuksia, jotka tuntuivat välillä vaikeuttavan tutkimuksen tekemistä kohtuuttomankin paljon. Esimerkiksi ontologioiden tuontiin liittyi paljon epäselvyyksiä. Tarkoituksena oli saada tuotua sekä sukuontologia että sen luokkien ilmentymät eli itse sukutieto Protégéhen. Useampia tiedostoja Protégé-projektiin tuotaessa ei kuitenkaan ollut lainkaan selvää, miten ne olisi parasta tuoda, sillä niiden tiedot näkyivät kunnolla vain tuotaessa tiedostot

juuri tietyssä järjestyksessä ja tietyllä tavalla. Lisäksi tuotavan tiedoston käsittelyssä oli ongelmia riippuen siitä, oliko kyseessä paikallinen vai WWW-resurssi.

Myös päättelyn tekeminen oli paikoin hieman ongelmallista. Edellisessä alaluvussa tehtiin yksinkertaiset päättelysäännöt ominaisuuden *hasChild* päättelemiseksi ominaisuuden *childIn* avulla ja vastaavasti ominaisuuden *hasSpouse* päättelemiseksi ominaisuuden *spouseIn* avulla. Nämä parit ovat toisilleen käänteisiä ominaisuuksia. Protégéssa käänteisyyden pystyy määrittämään ominaisuuden määrityksen yhteydessä, ja normaalisti riittää arvon antaminen toiselle ominaisuudelle, jolloin Protégé asettaa toisenkin automaattisesti. Tätä yritettiin hyödyntää nytkin, mutta toiminto ei onnistunut. Tämä johtui ilmeisestikin siitä, että sukutieto tuotiin Protégéhen valmiina, eikä Protégé enää siinä vaiheessa tarkistanut käänteisiä ominaisuuksia.

Protégéssa pystyy tekemään päättelyä myös suoraan OWL-kieleen pohjautuen. Tällaisen päättelyn avulla voidaan esimerkiksi tarkistaa ontologian yhtenäisyys tai luokitella ontologia uudestaan. Tässä tutkimuksessa kokeiltiin myös tämäntyyppistä päättelyä käyttäen päättelykoneita Pellet [Clark & Parsia, LLC, 2006] ja Racer [Racer Systems GmbH & Co. KG, 2006]. Näiden avulla yritettiin sijoittaa *Individual*-luokan ilmentymiä joihinkin uusiin luokkiin määrittämällä uusille luokille sellaisia rajoitteita, joiden perusteella luokittelu voitaisiin tehdä. Ongelmaksi kuitenkin muodostui se, että mikäli uuden luokan rajoitteen määrityksessä käytti tietotyyppi-ominaisuuksia, esimerkiksi *String*-tyyppiä, niin päättelyä ei saatu toimimaan. Viitteitä tämäntyyppisistä ongelmista on havaittu muissakin tutkimuksissa [Ruttenberg ym., 2005]. Näiden ongelmien vuoksi tässä tutkimuksessa päätettiin lopulta keskittyä SWRL-päättelyn tekemiseen.

Nämä Protégén käyttöön liittyvät ongelmat ratkesivat lähinnä yrityksen ja erehdyksen kautta, sillä niihin liittyen ei löytynyt kunnollista ohjeistusta. Monet mainituista epäselvyyksistä johtuvat varmastikin käyttäjän tottumattomuudesta Protégén käyttöön, ja ohjelmistoon syvällisemmin perehtyneille vakiokäyttäjille vastaavanlaisia ongelmia ei ehkä tulekaan vastaan. Tämä kertoo kuitenkin siitä, että ohjelmistoa pitäisi vielä kehittää paljon, ennen kuin sen laajamittaisempi käyttö olisi mahdollista, sillä ohjelmiston helppo ja selkeä käytettävyys olisi tässä avainasemassa.

5.6 Arviointia

Tutkielman empiirisen osan tavoitteena oli tutkia, kuinka semanttisen webin tekniikat soveltuvat sukutiedon käsittelyyn. Kaiken kaikkiaan voidaan todeta, että käytetyt tekniikat toimivat melko hyvin ja niitä voidaan varmastikin hyödyntää tulevaisuudessa. Arvioidaan seuraavaksi empiirisen osan tuloksia ja johtopäätöksiä hie-

man tarkemmin ja tehdään katsaus samojen menetelmien monipuolisempaan käyttöön tulevaisuudessa.

Nykyisessä GEDCOM-muodossa olevan sukutiedon muuntaminen semanttiseen muotoon onnistui hyvin muunnosohjelmaan tehtyjen pienten muutosten jälkeen. Tutkimuksessa käytetty muunnosohjelma ja siihen liittyvä ontologia oli tarkoitettu nimenomaan kokeilevaan tiedonmuunnokseen, ja niiden sisältämät GEDCOM-standardin tärkeimmät osat riittivätkin hyvin tämän tutkimuksen tarkoituksiin. Kohtuullisilla lisämuutoksilla GEDCOM-ontologian ja muunnosohjelman voisi varmasti täydentää sellaisiksi, että niitä voisi käyttää sukutiedon muuntamiseen laajemmassakin mittakaavassa. Ontologiaa pitäisi tällöin laajentaa kuvaamaan koko GEDCOM-standardi, ja samoin muunnosohjelmaan pitäisi tehdä ontologian laajennettuun osaan liittyvät täydennykset. Lisäksi muunnosohjelma tulisi laajentaa sellaiseksi, että sitä voisi käyttää useampienkin sukututkimusohjelmistojen tuottamien GEDCOM-tiedostojen muuntamiseen. Tosin joidenkin sukututkimusohjelmistojen tuottamat GEDCOM-tiedostot ovat keskenään niin ristiriitaisia, että muunnosohjelman muokkaaminen kaikille erityyppisille GEDCOM-tiedostoille saattaa olla hankalaa.

Semanttiseksi muunnetun sukutiedon esittäminen Protégé-editorilla onnistui hyvin edellisessä alaluvussa kuvattuja hankaluuksia lukuunottamatta. Pienillä muutoksilla sukutieto saatiin melko selkeästi näkyviin, ja päättelysääntöjen avulla sukutietoa pystyttiin helposti täydentämään. Nykyisiin sukututkimusohjelmistoihin verrattuna juuri päättelyn tekeminen onkin yksi semanttisen esitystavan eduista. Päättelykoneen avulla sukutiedosta voidaan hakea esimerkiksi juuri sellaisia sukulaisuussuhteita kuin sukututkija haluaa. SWRL:n avulla säännöt pystytään kirjoittamaan nopeasti, ja johtopäätökset voi liittää sukutietoon saman tien. Perinteisissä ohjelmistoissa vastaavanlaista mahdollisuutta ei ole, koska esimerkiksi tiettyjen sukulaisuussuhteiden selville saamiseksi täytyy ohjelmistossa olla valmiina mekanismi niiden selvittämiseksi. Uusien sukulaisuussuhteiden selvittämiseksi tarvitaan muutoksia ohjelmakoodiin, mikä kaupallisen ohjelmiston tapauksessa on yleensä työlästä ja hidasta. Lisäksi, mikäli tarve vaikkapa tietyn sukulaisuussuhteen selvittämiseen kiinnostaa vain harvoja tutkijoita, tällainen ominaisuus jää yleensä toteuttamatta.

Tutkimuksessa tehdyt toimenpiteet osoittavat käytettyjen tekniikoiden ja ohjelmistojen mahdollistavan olemassa olevan sukutiedon esityksen ja käsittelyn semanttisessa muodossa. Tietotekniikkaan perehtyneelle henkilölle toimenpiteet kävivät melko vaivattomasti, mutta tyypilliselle sukututkijalle nämä olisivat kuitenkin olleet teknisesti aivan liian vaikeita. Heitä varten tietämyskannan päälle voi-

taisiin rakentaa erillinen käyttöliittymä, joka mahdollistaisi kuvattujen toimenpiteiden huomattavasti helpomman suorituksen. Protégélla tehtyihin tietämyskantoihin pääsee käsiksi sovellusliittymän (engl. *application programming interface*, API) kautta, mikä mahdollistaa sen, että muut sovellukset voivat käyttää tietämyskannan sisältöä omiin tarkoituksiinsa [Stanford Medical Informatics, 2007a].

Tässä tutkimuksessa kuvatut toimenpiteet ovat vain esimerkki siitä, miten sukutietoa voidaan käsitellä semanttisen webin keinoin. Työn tärkeimpänä saavutuksena voidaan pitää sitä, että sukutieto saatiin muunnettua onnistuneesti semanttiseen muotoon, ja näin ollen kaikki nykyiset ja tulevat semanttisen webin tekniikat, menetelmät ja työkalut ovat myös sukututkijoiden käytettävissä. Tämä voisi helpottaa ja edistää sukututkimuksen tekemistä tulevaisuudessa niin tietojen etsiminen ja käsittelyn kuin tulosten esittämisenkin kannalta. Esimerkiksi sukutiedon monipuolinen haku WWW:stä, erilaisten päättelyjen tekeminen sekä eri tutkimusalojen tietojen yhdistäminen sukutietoon voisivat jatkossa olla mahdollisia semanttisen webin myötä. Lisäksi semanttinen esitysmuoto ja semanttiset tekniikat mahdollistavat tulevaisuudessa erityisesti sukutiedon nykyistä paremman siirrettävyyden.

6 Yhteenveto

Semanttinen web on nykyisen WWW:n laajennos, jonka tarkoituksena on tulevaisuudessa olla ihmisten lisäksi myös koneiden käytettävissä. Tämän mahdollistaa tiedon esittäminen sellaisessa muodossa, että koneet pystyvät ”ymmärtämään” tiedon merkityksiä automaattisesti. Tiedon semanttiseen esittämiseen tarvitaan RDF-tietomallia ja ontologioita. RDF mahdollistaa tiedon ja sen merkityksen esittämisen, ja ontologiat tarjoavat keinon tietoon liittyvien käsitteiden määritykseen. Semantiikan käyttö mahdollistaisi WWW:n nykyistä paljon monimuotoisemman käytön älykkäämpien palveluiden ja sovellusten tullessa mahdollisiksi, ja esimerkiksi tiedonhaku helpottuisi huomattavasti.

WWW:n laajentaminen semanttiseksi on kuitenkin vasta kehitysvaiheessa, ja kestää vielä kauan aikaa, ennen kuin siihen liittyvien tekniikoiden ja sovelluksien kehittäminen on sellaisessa pisteessä, että voidaan todella puhua semanttisesta webistä. Realistisempaa onkin odottaa semanttisten tekniikoiden tulevan ensin käyttöön tietyillä sovellusaloilla, ja mahdollisesti vasta myöhemmin tulevaisuudessa koko WWW:ssä. Tekniikan ja sovelluksien kehittymisen lisäksi suurena haasteena on niiden käyttöönotto. WWW:n käyttäjät ovat tottuneet nykyisiin tekniikoihin, joten heidän motivoimisensa semanttisen webin tekniikoiden käyttöön ei ole aivan helppoa.

Semanttista webiä voidaan soveltaa useilla eri aloilla, ja tässä tutkielmassa perehdyttiin semanttisen webin tekniikoiden soveltamiseen sukututkimuksessa. Nykyiset sukututkimusohjelmistot ovat monipuolisia työkaluja sukututkimustyön tulosten tallentamisessa, muokkaamisessa ja esittämisessä. Haasteellista niissä on kuitenkin tietojen siirrettävyys eri ohjelmistojen välillä, mikä usein olisi tarpeellista. Sukutiedon siirtoon on kehitetty GEDCOM-standardi, mutta sen epäselvyydestä ja monista tulkinnoista johtuen siirrettävyys on kuitenkin käytännössä melko heikkoa.

Tutkielman tavoitteena oli selvittää, kuinka olemassa olevaa sukutietoa pystyy muuntamaan semanttiseen muotoon ja käsittelemään semanttisen webin tekniikoiden avulla. Ensin nykyisessä GEDCOM-muodossa oleva sukutieto muunnettiin semanttiseksi muunnosohjelman avulla, minkä jälkeen tarkasteltiin sukutiedon käsiteltävyyttä Protégé-ontologiaeditorilla. Lisäksi ontologiaan tehtiin päättelysääntöjä, joiden avulla sukutietoon pystyttiin lisäämään uutta tietoa. Nämä toimenpiteet onnistuivat joitakin vaikeuksia lukuunottamatta hyvin. Laajempaa käyttöä ajatellen

käytetyt menetelmät eivät kuitenkaan tällaisenaan riitä, vaan ollakseen yleiskäyttöisempiä sekä muunnosohjelmaa että siihen liittyvää ontologiaa tulisi laajentaa. Lisäksi ongelmana on tutkimuksessa käytettyjen menetelmien tekninen haastavuus, ja tällaisenaan tyypillinen sukututkija ei näitä juurikaan pystyisi hyödyntämään.

Tutkimus kuitenkin osoittaa käytettyjen tekniikoiden ja ohjelmistojen mahdollistavan sukutiedon esityksen ja käsittelyn semanttisin keinoin. Nykyisiä sukututkimusohjelmistoja uudet työkalut eivät varmasti pysty korvaamaan vielä vuosiin, mutta esimerkiksi tutkimuksessa tehty sukulaisuussuhteita koskeva päättely osoittaa semanttisella webillä olevan sellaisia mahdollisuuksia, jotka nykyisistä ohjelmistoista puuttuvat.

Tutkimuksen tärkeimpänä saavutuksena voidaan pitää olemassa olevan sukutiedon onnistunutta muunnosta semanttiseen muotoon. Sen ansiosta sukutietoa pystytään käsittelemään lukuisien nykyisten ja tulevien semanttisen webin tekniikoiden, menetelmien ja työkalujen avulla. Nämä mahdollistavat nykyiseen verrattuna paljon monimuotoisemman tiedon etsimisen, muokkaamisen ja esityksen, kun nämä toimenpiteet eivät rajoitu enää yhteen ohjelmistoon tai esityskieleen. Etenkin nykyisin ongelmallisena pidetty sukutiedon siirrettävyys voisi parantua olennaisesti. Vanhaan GEDCOM-standardiin pohjautuvaan siirtotapaan verrattuna ontologioiden ja semanttisen kielen käyttö sukutiedon esityksessä voisi yhtenäistää tiedon esitystapaa ja parantaa täten siirrettävyyttä huomattavasti.

Semanttisen webin tekniikoilla on siis mahdollista helpottaa sukututkimuksen tekemistä jo nyt ja vielä paremmin tulevaisuuudessa. Ongelmana tässä – kuten varmasti muillakin sovellusaloilla – on kuitenkin se, että itse sukututkijat eivät ole tietoisia uusista tekniikoista, eivätkä tyypillisesti edes teknisesti niin taitavia, että pystyisivät niitä tällä hetkellä hyödyntämään. Tulevaisuudessa semanttisen webin hyödyt voitaisiin saada myös sukututkijoiden ulottuville juuri heidän yksilöllisten tarpeidensa mukaisesti kehitettyjen ohjelmistojen muodossa. Kaiken kaikkiaan semanttisen webin voidaan odottaa helpottavan sukututkimusta tulevaisuudessa uusien tekniikoiden yleistyessä.

Lähteet

[Antoniou ym., 2004] Antoniou G. ja van Harmelen F., *A Semantic Web Primer*, The MIT Press, Cambridge, 2004.

[Askren, 2007] Askren J., *The Semantic Web for Family History*, <http://jay.askren.net/Projects/SemWeb/>, viitattu 6.2.2007.

[Bechhofer ym., 2004] Bechhofer S., van Harmelen F., Hendler J. ym., *OWL Web Ontology Language, Reference*, W3C:n suositus, <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>, 10.2.2004.

[Berners-Lee ym., 2001] Berners-Lee T., Hendler J. ja Lassila O., *The semantic web*, Scientific American, 284 (5), 2001, s. 34–43.

[Berners-Lee ym., 2005] Berners-Lee T., Fielding R. ja Masinter L., *Uniform Resource Identifier (URI): Generic Syntax*, Network Working Group, <http://gbiv.com/protocols/uri/rfc/rfc3986.html>, 01/2005.

[Boley, 2006] Boley H. (toim.), *The Rule Markup Initiative*, <http://www.ruleml.org/>, 24.10.2006.

[Bray ym., 2006] Bray T., Paoli J., Sperberg-McQueen C. M. ym. (toim.), *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, W3C:n suositus, <http://www.w3.org/TR/2006/REC-xml-20060816/>, 29.9.2006.

[Brickley ym., 2004] Brickley D. ja Guha R. V. (toim.), *RDF Vocabulary Description Language 1.0: RDF Schema*, W3C:n suositus, <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>, 10.2.2004.

[Chandrasekaran ym., 1999] Chandrasekaran B., Josephson J. R. ja Benjamins V. R., *What Are Ontologies, and Why Do We Need Them?*, IEEE Intelligent Systems, 14 (1), 1999, s. 20–26.

[CHISEL, 2007] Computer-Human Interaction and Software Engineering Lab (CHISEL), *Jambalaya*, ohjelmiston kotisivu, <http://www.thechiselgroup.org/jambalaya>, viitattu 6.2.2007.

[Citius Solutions Oy, 2007] Citius Solutions Oy, *Genus Senior Sukututkimusohjelma*, ohjelmiston kotisivu, <http://www.genus.fi>, viitattu 12.3.2007.

[Clark & Parsia, LLC, 2006] Clark & Parsia, LLC, *Pellet: An OWL DL Reasoner*, ohjelmiston kotisivu, <http://pellet.owldl.com/>, 7.11.2006.

[Clerkin ym., 2001] Clerkin P., Cunningham P. ja Hayes C., *Ontology Discovery for the Semantic Web Using Hierarchical Clustering*, Proceedings of the Semantic Web Mining Workshop at the 12th European Conference on Machine Learning (ECML'01) and the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01), Freiburg, Saksa, 2001, s. 27-38.

[Consegur Oy, 2007] Consegur Oy, *Juuret 2.1 sukututkimusohjelmisto*, ohjelmiston kotisivu, <http://juuret.fi>, 15.02.2007.

[Corcho ym., 2001] Corcho O., Fernández-López M. ja Gómez Pérez A., *OntoWeb, Technical Roadmap v1.0*, Universidad Politécnica de Madrid, <http://ontoweb.org/About/Deliverables/Deliverable111.pdf>, 30.11.2001.

[Cover, 2005] Cover R. (toim.), *Genealogical Data and XML*, tekninen raportti, <http://xml.coverpages.org/genealogy.html>, 30.12.2005.

[Cycorp, Inc., 2007] Cycorp, Inc., *Cyc-tietämyskanta*, <http://www.cyc.com/>, viitattu 12.2.2007.

[DCMI, 2006] Dublin Core Metadata Initiative (DCMI), *DCMI Metadata Terms*, <http://dublincore.org/documents/dcmi-terms/>, 18.12.2006.

[Dean, 2002] Dean M. (toim.), *Genealogical Data Communication (GEDCOM) represented in DAML+OIL*, <http://www.daml.org/2001/01/gedcom/gedcom/>, 6.9.2002.

[Dean, 2003] Dean M. (toim.), *GEDCOM to DAML*, <http://www.daml.org/2001/01/gedcom/>, 4.8.2003.

[DeRose ym., 2001] DeRose S., Maler E. ja Orchard D. (toim.), *XML Linking Language (XLink) Version 1.0*, W3C:n suositus, <http://www.w3.org/TR/2001/REC-xlink-20010627/>, 27.6.2001.

[Eriksson ym., 1999] Eriksson H., Fergerson R. W., Shahar Y. ym., *Automatic Generation of Ontology Editors*, Proceedings of the 12th International Workshop on Knowledge Acquisition, Modelling and Management (KAW'99), Banff, Kanada, 1999.

[Fensel, 2000] Fensel D. (toim.), *The semantic Web and its languages*, IEEE Intelligent Systems, 15 (6), 2000, s. 67–73.

[Fensel, 2001] Fensel D., *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*, Springer-Verlag, Berlin, 2001.

[FOAF, 2007] *The Friend of a Friend (FOAF) project*, <http://www.foaf-project.org/>, viitattu 12.2.2007.

[Friedman-Hill, 2006] Friedman-Hill E., *Jess, the Rule Engine for the Java Platform*, ohjelmiston kotisivu, Sandia National Laboratories, <http://www.jessrules.com/>, 21.12.2006.

[Gruber, 1993] Gruber T. R., *A Translation Approach to Portable Ontology Specifications*, Knowledge Acquisition, 5 (2), 1993, s. 199–220.

[Guarino, 1997] Guarino N., *Understanding, building and using ontologies*, International Journal of Human-Computer Studies, 46 (2-3), 1997, s. 293–310.

[Hawke ym., 2005] Hawke S., Tabet S. ja de Sainte Marie C., *Rule Language Standardization, Report from the W3C Workshop on Rule Languages for Interoperability*, tekninen raportti, W3C, <http://www.w3.org/2004/12/rules-ws/report/>, 8.6.2005.

[Helsingin yliopiston kirjasto, 2002] Helsingin yliopiston kirjasto – Suomen kansalliskirjasto, *Dublin Core metadataformaatin suomalainen versio*, <http://www.lib.helsinki.fi/dublin_core/dc-sfs.html>, 9.10.2002.

[Hetemäki, 1999] Hetemäki I. (toim.), *Filosofian sanakirja*, WSOY, Juva, 1999.

[Hillmann, 2005] Hillmann D., *Using Dublin Core*, Dublin Core Metadata Initiative (DCMI), <http://dublincore.org/documents/usageguide/>, 7.11.2005.

[Holt ym., 2002] Holt R. C., Schürr A., Sim S. E. ym., *Graph eXchange Language (GXL)*, <http://www.gupro.de/GXL/>, 17.7.2002.

[Horrocks ym., 2004] Horrocks I., Patel-Schneider P. F., Boley H. ym., *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, tekninen raportti, W3C, <http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/>, 21.5.2004.

[Hyvönen, 2002] Hyvönen E., *The Semantic Web – The New Internet of Meanings*, kirjassa *Semantic Web Kick-Off in Finland: Vision, Technologies, Research, and Applications*, Hyvönen E. (toim.), HIIT Publications 2002-01, Helsinki Institute for Information Technology, 2002, s. 3–25.

[Kaila, 2007] Kaila K., *Sukuohjelmisto*, ohjelmiston kotisivu, <http://www.sukuohjelmisto.fi/>, viitattu 12.3.2007.

[Kielikone Oy, 2006] Kielikone Oy, *MOT Tietotekniikan liiton ATK-sanakirja 4.0*, <http://mot.kielikone.fi/mot/jyu/netmot.exe>, viitattu 21.3.2007.

[Klyne ym., 2004] Klyne G. ja Carroll J. J. (toim.), *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C:n suositus, <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>, 10.2.2004.

[Knublauch ym., 2004] Knublauch H., Fergerson R. W., Noy N. F. ym., *The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications*, Proceedings of the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japani, 2004, s. 229-243.

[LDS Church, 1996] The Church of Jesus Christ of Latter-day Saints (LDS Church), Family History Department, *The GEDCOM Standard Release 5.5*, tekninen raportti, <http://www.familysearch.org/gedcom/gedcom55.exe>, 2.1.1996.

[LDS Church, 2001] The Church of Jesus Christ of Latter-day Saints (LDS Church), Family History Department, *GEDCOM XML Specification Release 6.0 Draft*, tekninen raportti, <http://www.familysearch.org/gedcom/GedXML60.pdf>, 28.12.2001.

[LDS Church, 2006] The Church of Jesus Christ of Latter-day Saints (LDS Church), Family History Department, *Personal Ancestral File*, ohjelmiston kotisivu, <http://www.familysearch.org/paf>, viitattu 27.3.2006.

[Luhtasaari, 2007] Luhtasaari S., *SukuJutut*, ohjelmiston kotisivu, <http://www.sukujutut.fi/sukujut/index.htm>, viitattu 12.3.2007.

[Luttinen ym., 1988] Luttinen R. ja Hyppönen M., *Sukututkimuksen käsikirja*, WSOY, Porvoo, 1988.

[Maedche ym., 2001] Maedche A., Staab S., *Ontology Learning for the Semantic Web*, IEEE Intelligent Systems, 16 (2), 2001, s. 72–79.

[Manola ym., 2004] Manola F. ja Miller E. (toim.), *RDF Primer*, W3C:n suositus, <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>, 10.2.2004.

[Martinsons, 1995] Martinsons M.G., *Knowledge-based systems leverage human resource management expertise*, International Journal of Manpower, 16 (2), 1995, s. 17-34.

[McGuinness ym., 2004] McGuinness D. L. ja van Harmelen F. (toim.), *OWL Web Ontology Language, Overview*, W3C:n suositus, <http://www.w3.org/TR/2004/REC-owl-features-20040210/>, 10.2.2004.

[Miller ym., 2007] Miller G. A., Fellbaum C., Tengi R. ym., *WordNet – a lexical database for the English language*, Cognitive Science Laboratory, Princeton University, <http://wordnet.princeton.edu/>, viitattu 12.2.2007.

[Noy, 2005] Noy N. (toim.), *Representing Classes As Property Values on the Semantic Web*, tekninen raportti, W3C, <http://www.w3.org/TR/swbp-classes-as-values/>, 5.4.2005.

[Noy ym., 2001] Noy N. F. ja McGuinness D. L., *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford University, Medical Informatics, tekninen raportti, 2001, <http://protege.stanford.edu/publications/ontology_development/ontology101.pdf>.

[O'Connor, 2007] O'Connor M., *SWRLTab*, Stanford University, <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab>, 4.2.2007.

[O'Connor ym., 2005] O'Connor M., Knublauch H. ym., *Supporting Rule System Interoperability on the Semantic Web with SWRL*, Proceedings of the 4th International Semantic Web Conference (ISWC2005), Galway, Irlanti, 2005, s. 974-986.

[OMG, 2005] Object Management Group (OMG), *MOF 2.0/XMI Mapping Specification, v2.1*, <http://www.omg.org/docs/formal/05-09-01.pdf>, 1.9.2005.

[Palander, 1999] Palander S., *Sukututkimusta tietokoneella*, KotiPC, 6, 1999, s. 30–34.

[Pepper ym., 2001] Pepper S. ja Moore G. (toim.), *XML Topic Maps (XTM) 1.0*, <http://www.topicmaps.org/xtm/>, 6.8.2001.

[PhpGedView, 2007] *PhpGedView*, <http://www.phpgedview.net/>, 14.3.2007.

[Racer Systems GmbH & Co. KG, 2006] Racer Systems GmbH & Co. KG, *Racer*, ohjelmiston kotisivu, <http://www.racer-systems.com/>, 11.12.2006.

[Raggett ym., 1999] Raggett D., Le Hors A. ja Jacobs I. (toim.), *HTML 4.01 Specification*, W3C:n suositus, <http://www.w3.org/TR/1999/REC-html401-19991224/>, 24.12.1999.

[RosettaNet, 2007] *RosettaNet*, RosettaNet-yhteisön kotisivu, <http://www.rosettanet.org/>, viitattu 12.2.2007.

[Ruttenberg ym., 2005] Ruttenberg A., Rees J. A. ja Luciano J. S., *Experience Using OWL DL for the Exchange of Biological Pathway Information*, Proceedings of the OWL Experiences and Directions Workshop, Galway, Irlanti, 2005.

[Salminen, 2005] Salminen A., *Jakso 1: XML pähkinänkuoressa*, kurssin XML-kieli (Extensible Markup Language) materiaalia, Jyväskylän yliopisto, Tietojenkäsittelytieteiden laitos, 2005, <http://www.cs.jyu.fi/~airi/opetus/xml/xml-kieli/xml-jakso1-100205.pdf>, viitattu 20.6.2005.

[Silvonen ym., 2002] Silvonen P. ja Hyvönen E., *Semantic Web Tools*, kirjassa *Semantic Web Kick-Off in Finland: Vision, Technologies, Research, and Applications*, Hyvönen E. (toim.), HIIT Publications 2002-01, Helsinki Institute for Information Technology, 2002, s. 137–152.

[Sintek, 2006] Sintek M., *OntoViz*, Stanford University, <http://protege.cim3.net/cgi-bin/wiki.pl?OntoViz>, 27.2.2007.

[Sippu, 2000] Sippu S., *Tietokoneavusteinen sukututkimus*, Genos, 3, 2000, s. 127–144.

[Stanford Medical Informatics, 2007a] Stanford Medical Informatics, *Protégé*, ohjelmiston kotisivu, <http://protege.stanford.edu>, viitattu 9.3.2007.

[Stanford Medical Informatics, 2007b] Stanford Medical Informatics, *Protégé-2000 User's Guide*, <http://protege.stanford.edu/publications/UserGuide.pdf>, viitattu 9.3.2007.

[Stanford Medical Informatics, 2007c] Stanford Medical Informatics, *Protégé-OWL FAQ*, <http://protege.stanford.edu/doc/owl-faq.html>, viitattu 9.3.2007.

[Sugimoto, 2006] Sugimoto M. L., *Genealogy and the Semantic Web: A Guide for Family Historians and Amateur Genealogists*, tekninen raportti, <http://polaris.gseis.ucla.edu/mleahey/genealogyAndSemanticWebXHTML.htm>, 15.6.2006.

[Suomen Sukututkimusseura, 1999] Suomen Sukututkimusseura, *Sukututkimus askel askeleelta*, Gummerus Kirjapaino Oy, Jyväskylä, 1999.

[Suomen Sukututkimusseura, 2005] Suomen Sukututkimusseura, *Sukututkimusohjelmat*, <http://www.genealogia.fi/sukuo>, viitattu 10.6.2005.

[Varjonen, 2006] Varjonen V., *Viittaaminen sähköisiin dokumentteihin*, Oulun yliopiston kirjasto, <http://herkules.oulu.fi/vili/viittaus/>, 28.2.2006.

[Walker, 2004] Walker T., *Automating the Extraction of Domain-Specific Information from the Web – A Case Study for the Genealogical Domain*, Master's Thesis, Department of Computer Science, Brigham Young University, Yhdysvallat, marraskuu 2004, <http://www.deg.byu.edu/papers/thesis.walker.final.pdf>.

[Wang ym., 2006] Wang T. D., Parsia B. ja Hendler J., *A Survey of the Web Ontology Landscape*, Proceedings of the 5th International Semantic Web Conference (ISWC2006), Athens, Georgia, Yhdysvallat, 2006, s. 682-694.

[Zandhuis, 2005] Zandhuis I., *Towards a Genealogical Ontology for the Semantic Web*, Proceedings of the 16th International Conference of the Association for History and Computing, Amsterdam, Alankomaat, 2005, s. 296-300.

A GEDCOM-ontologia

```
<!-- Created with Protege (with OWL Plugin 3.2, Build 355)  http://protege.stanford.edu -->

<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:j.0="http://www.w3.org/2003/11/swrl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:p1="http://www.cc.jyu.fi/~jennim/gradu/gedcom#"
  xml:base="http://www.cc.jyu.fi/~jennim/gradu/gedcom">

  <owl:Ontology rdf:about="">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      Genealogical Data Communication (GEDCOM) represented in DAML+OIL</rdfs:comment>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      $Id: gedcom.daml,v 1.15 2002/09/06 15:38:18 mdean Exp $</owl:versionInfo>
  </owl:Ontology>

  <owl:Class rdf:ID="Birth">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="IndividualEvent"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Death">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#IndividualEvent"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Individual">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      An individual named in a GEDCOM file.
    </rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
```

```
      <owl:ObjectProperty rdf:ID="hasCousin"/>
    </owl:onProperty>
    <owl:allValuesFrom rdf:resource="#Individual"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="title"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="surname"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="sex"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <owl:onProperty rdf:resource="#title"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="childIn"/>
```

```
    </owl:onProperty>
    <owl:allValuesFrom>
      <owl:Class rdf:ID="Family"/>
    </owl:allValuesFrom>
  </owl:Restriction>
</rdfs:subClassOf>
<owl:sameAs rdf:resource="http://www.daml.org/2001/03/daml+oil-ex#Person"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="spouseIn"/>
    </owl:onProperty>
    <owl:allValuesFrom>
      <owl:Class rdf:about="#Family"/>
    </owl:allValuesFrom>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasSibling"/>
    </owl:onProperty>
    <owl:allValuesFrom rdf:resource="#Individual"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="#Birth"/>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="birth"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#childIn"/>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
```

```
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="name"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="#Individual"/>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasUncle"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="death"/>
    </owl:onProperty>
    <owl:allValuesFrom rdf:resource="#Death"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    <owl:onProperty rdf:resource="#sex"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="#Individual"/>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasParent"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="givenName"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    <owl:onProperty rdf:resource="#name"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#surname"/>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="#Individual"/>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasAunt"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="#death"/>
    </owl:onProperty>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    <owl:onProperty>
```

```
        <owl:ObjectProperty rdf:about="#birth"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      <owl:onProperty rdf:resource="#givenName"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="FamilyEvent">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Event"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#IndividualEvent">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Event"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Marriage">
  <rdfs:subClassOf rdf:resource="#FamilyEvent"/>
</owl:Class>
<owl:Class rdf:ID="Divorce">
  <rdfs:subClassOf rdf:resource="#FamilyEvent"/>
</owl:Class>
<owl:Class rdf:about="#Event">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="date"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
```

```
      <owl:onProperty rdf:resource="#date"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="place"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#place"/>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Family">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Marriage"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="marriage"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="divorce"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Divorce"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Individual"/>
      <owl:onProperty>
```

```
        <owl:ObjectProperty rdf:ID="hasChild"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Individual"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasSpouse"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    A family named in a GEDCOM file.
  </rdfs:comment>
</owl:Class>
<owl:ObjectProperty rdf:ID="familyEventProperty">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="eventProperty"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#birth">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="individualEventProperty"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#marriage">
  <rdfs:subPropertyOf rdf:resource="#familyEventProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#individualEventProperty">
  <rdfs:subPropertyOf rdf:resource="#eventProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#divorce">
  <rdfs:subPropertyOf rdf:resource="#familyEventProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#death">
  <rdfs:subPropertyOf rdf:resource="#individualEventProperty"/>
</owl:ObjectProperty>
<j.0:Imp rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#Rule-5">
  <j.0:body>
    <j.0:AtomList>
```

```
    <rdf:rest>
      <j.0:AtomList>
        <rdf:first>
          <j.0:IndividualPropertyAtom>
            <j.0:argument2>
              <j.0:Variable rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
            </j.0:argument2>
            <j.0:propertyPredicate rdf:resource="#hasSibling"/>
            <j.0:argument1>
              <j.0:Variable rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
            </j.0:argument1>
          </j.0:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest>
          <j.0:AtomList>
            <rdf:first>
              <j.0:DatavaluedPropertyAtom>
                <j.0:argument1 rdf:resource=
                  "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
                <j.0:propertyPredicate rdf:resource="#sex"/>
                <j.0:argument2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >M</j.0:argument2>
              </j.0:DatavaluedPropertyAtom>
            </rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          </j.0:AtomList>
        </rdf:rest>
      </j.0:AtomList>
    </rdf:rest>
    <rdf:first>
      <j.0:IndividualPropertyAtom>
        <j.0:propertyPredicate rdf:resource="#hasParent"/>
        <j.0:argument1>
          <j.0:Variable rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
        </j.0:argument1>
        <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
      </j.0:IndividualPropertyAtom>
    </rdf:first>
  </j.0:AtomList>
</j.0:body>
<j.0:head>
```

```
    <j.0:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
          <j.0:propertyPredicate rdf:resource="#hasUncle"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
    </j.0:AtomList>
  </j.0:head>
</j.0:Imp>
<j.0:Imp rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#Rule-7">
  <j.0:head>
    <j.0:AtomList>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:propertyPredicate rdf:resource="#hasCousin"/>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
          <j.0:argument2>
            <j.0:Variable rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#a"/>
          </j.0:argument2>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </j.0:AtomList>
  </j.0:head>
  <j.0:body>
    <j.0:AtomList>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:propertyPredicate rdf:resource="#hasParent"/>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest>
        <j.0:AtomList>
          <rdf:rest>
            <j.0:AtomList>
              <rdf:first>
```

```
                <j.0:IndividualPropertyAtom>
                  <j.0:propertyPredicate rdf:resource="#hasParent"/>
                  <j.0:argument1 rdf:resource=
                    "http://www.owl-ontologies.com/Ontology1168341945.owl#a"/>
                  <j.0:argument2 rdf:resource=
                    "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
                </j.0:IndividualPropertyAtom>
              </rdf:first>
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            </j.0:AtomList>
          </rdf:rest>
          <rdf:first>
            <j.0:IndividualPropertyAtom>
              <j.0:argument2 rdf:resource=
                "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
              <j.0:argument1 rdf:resource=
                "http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
              <j.0:propertyPredicate rdf:resource="#hasSibling"/>
            </j.0:IndividualPropertyAtom>
          </rdf:first>
        </j.0:AtomList>
      </rdf:rest>
    </j.0:AtomList>
  </j.0:body>
</j.0:Imp>
<j.0:Imp rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#Rule-2">
  <j.0:body>
    <j.0:AtomList>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:propertyPredicate rdf:resource="#spouseIn"/>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </j.0:AtomList>
  </j.0:body>
  <j.0:head>
    <j.0:AtomList>
      <rdf:first>
```

```
        <j.0:IndividualPropertyAtom>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
          <j.0:propertyPredicate rdf:resource="#hasSpouse"/>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </j.0:AtomList>
  </j.0:head>
</j.0:Imp>
<j.0:Imp rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#Rule-1">
  <j.0:head>
    <j.0:AtomList>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
          <j.0:propertyPredicate rdf:resource="#hasChild"/>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </j.0:AtomList>
  </j.0:head>
  <j.0:body>
    <j.0:AtomList>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:propertyPredicate rdf:resource="#childIn"/>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </j.0:AtomList>
  </j.0:body>
</j.0:Imp>
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1166534248.owl">
  <owl:imports rdf:resource="http://www.w3.org/2003/11/swrl"/>
  <owl:imports rdf:resource="http://www.w3.org/2003/11/swrlb"/>
</rdf:Description>
<j.0:Imp rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#Rule-6">
```

```
<j.0:head>
  <j.0:AtomList>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first>
      <j.0:IndividualPropertyAtom>
        <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
        <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
        <j.0:propertyPredicate rdf:resource="#hasAunt"/>
      </j.0:IndividualPropertyAtom>
    </rdf:first>
  </j.0:AtomList>
</j.0:head>
<j.0:body>
  <j.0:AtomList>
    <rdf:rest>
      <j.0:AtomList>
        <rdf:rest>
          <j.0:AtomList>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            <rdf:first>
              <j.0:DatavaluedPropertyAtom>
                <j.0:argument2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >F</j.0:argument2>
                <j.0:propertyPredicate rdf:resource="#sex"/>
                <j.0:argument1 rdf:resource=
                  "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
              </j.0:DatavaluedPropertyAtom>
            </rdf:first>
          </j.0:AtomList>
        </rdf:rest>
        <rdf:first>
          <j.0:IndividualPropertyAtom>
            <j.0:propertyPredicate rdf:resource="#hasSibling"/>
            <j.0:argument2 rdf:resource=
              "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
            <j.0:argument1 rdf:resource=
              "http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
          </j.0:IndividualPropertyAtom>
        </rdf:first>
      </j.0:AtomList>
    </rdf:rest>
```

```
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
          <j.0:propertyPredicate rdf:resource="#hasParent"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
    </j.0:AtomList>
  </j.0:body>
</j.0:Imp>
<j.0:Imp rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#Rule-3">
  <j.0:body>
    <j.0:AtomList>
      <rdf:rest>
        <j.0:AtomList>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          <rdf:first>
            <j.0:IndividualPropertyAtom>
              <j.0:propertyPredicate rdf:resource="#hasSpouse"/>
              <j.0:argument2 rdf:resource=
                "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
              <j.0:argument1 rdf:resource=
                "http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
            </j.0:IndividualPropertyAtom>
          </rdf:first>
        </j.0:AtomList>
      </rdf:rest>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:propertyPredicate rdf:resource="#childIn"/>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
    </j.0:AtomList>
  </j.0:body>
  <j.0:head>
    <j.0:AtomList>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
```

```
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
          <j.0:propertyPredicate rdf:resource="#hasParent"/>
        </j.0:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </j.0:AtomList>
  </j.0:head>
</j.0:Imp>
<j.0:Imp rdf:about="http://www.owl-ontologies.com/Ontology1168341945.owl#Rule-4">
  <j.0:body>
    <j.0:AtomList>
      <rdf:rest>
        <j.0:AtomList>
          <rdf:first>
            <j.0:IndividualPropertyAtom>
              <j.0:argument1 rdf:resource=
                "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
              <j.0:argument2 rdf:resource=
                "http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
              <j.0:propertyPredicate rdf:resource="#hasParent"/>
            </j.0:IndividualPropertyAtom>
          </rdf:first>
          <rdf:rest>
            <j.0:AtomList>
              <rdf:first>
                <j.0:DifferentIndividualsAtom>
                  <j.0:argument2 rdf:resource=
                    "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
                  <j.0:argument1 rdf:resource=
                    "http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
                </j.0:DifferentIndividualsAtom>
              </rdf:first>
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            </j.0:AtomList>
          </rdf:rest>
        </j.0:AtomList>
      </rdf:rest>
      <rdf:first>
        <j.0:IndividualPropertyAtom>
          <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#y"/>
          <j.0:propertyPredicate rdf:resource="#hasParent"/>
```

```
            <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
          </j.0:IndividualPropertyAtom>
        </rdf:first>
      </j.0:AtomList>
    </j.0:body>
    <j.0:head>
      <j.0:AtomList>
        <rdf:first>
          <j.0:IndividualPropertyAtom>
            <j.0:argument2 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
            <j.0:propertyPredicate rdf:resource="#hasSibling"/>
            <j.0:argument1 rdf:resource="http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
          </j.0:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest>
          <j.0:AtomList>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            <rdf:first>
              <j.0:IndividualPropertyAtom>
                <j.0:argument2 rdf:resource=
                  "http://www.owl-ontologies.com/Ontology1168341945.owl#x"/>
                <j.0:propertyPredicate rdf:resource="#hasSibling"/>
                <j.0:argument1 rdf:resource=
                  "http://www.owl-ontologies.com/Ontology1168341945.owl#z"/>
              </j.0:IndividualPropertyAtom>
            </rdf:first>
          </j.0:AtomList>
        </rdf:rest>
      </j.0:AtomList>
    </j.0:head>
  </j.0:Imp>
</rdf:RDF>
```