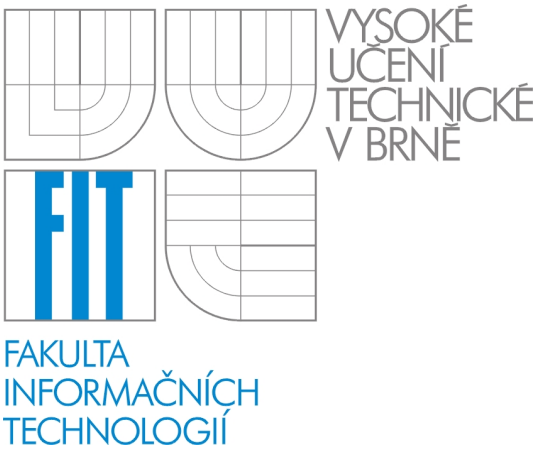


Projekt do předmětu VPD

Sémantický web

2009 / 2010

Ing. Jaroslav Dytrych, xdytry00@stud.fit.vutbr.cz
Fakulta informačních technologií
Vysoké učení technické v Brně

Obsah

1 Úvod

V této práci se budu zabývat sémantickým webem. Původní idea je zatím pouze ve vzdálené budoucnosti a aktuální definice sémantického webu se od ní liší. Práci tedy začnu od počáteční ideje v 2. kapitole, ve 3. kapitole potom uvedu současnou definici a následně se ve 4. kapitole budu zabývat tím, kde je v sémantickém webu sémantika, podle které je tento web pojmenován.

Pro sémantický web jsou velmi důležité ontologie, což jsou rozsáhlé sdílené databáze znalostí. Ontologiemi se budu zabývat ve 5. kapitole.

Šestou kapitolu jsem věnoval popisu nejdůležitějších a nejznámějších technologií současného sémantického webu. Popíši v ní základní technologii sémantického webu, kterou je RDF, dotazovací jazyk nad stromy se sémantickými informacemi, jazyky pro popis ontologií a další technologie potřebné pro tvorbu sémantického webu.

V sedmé kapitole uvedu postup vytvoření jednoduché ontologie v OWL v editoru Protégé, při kterém současně vysvětlím vybrané konstrukce daného jazyka.

2 Idea sémantického webu

Tim Berners-Lee, Jim Hendler a Ora Lassila ve svém článku pro Scientific American prezentovali ideu sémantického webu, ve které by softwaroví agenti na webu automaticky zpracovávali dostupné informace a prováděli za lidi běžné úkony, jako plánování návštěvy u lékaře s ohledem na plánované schůzky, hodnocení lékaře, volné termíny, dopravní situaci v okolí apod. Překlad modelové situace z daného článku lze nalézt v [17].

Pro dosažení daných cílů by bylo nutné, aby měl obsah webu dobře definovanou sémantiku, což umožní počítačům vyhodnotit význam informací a vztahy mezi nimi. Softwarový agent by potom s využitím informací na webu mohl „porozumět“ požadavku uživatele a najít vhodné řešení daného problému. Základem je tedy rozšíření současného webu o sémantické informace.

Problémem je, že sémantický web se zatím vyskytuje především v akademickém a výzkumném prostředí. Dokud většina webů neobsahuje sémantické informace, práce se sémantikou není v širším měřítku možná (agent by měl málo informací pro plnění složitějších úkolů). Pro mnoho firem se potom zdá investice do sémantizace webu neperspektivní. Jedná se o situaci, ze které je jediným východiskem to, že některé z větších firem nabídnou produkty s využitím sémantického webu, jejichž úspěch přinutí konkurenci k tomu, aby posunula vývoj sémantického webu dále. Tato situace však zatím nenastala a idea z výše zmíněného článku je zatím pouhou vizí do vzdálené budoucnosti.

3 Definice sémantického webu

Existuje mnoho různých definic sémantického webu, které odrážejí spíše aktuální stav vývoje, než původní vizi. Uvedu zde stručnou a výstižnou definici z [16]: „Sémantický web je rozšíření současného webu, ve kterém je informacím dán dobře definovaný význam, který umožňuje lépe spolupracovat lidem a počítačům.“

Sémantický web by měl umožnit např. vyhledávání nejenom podle obsahu, ale také podle popisu. Měl by umožnit vyhledávání článků od určitého autora (rozpoznat je od článků s citacemi autora) apod. Základem pro umožnění této funkcionality je možnost anotování informací ve stránkách. Pokud např. anotujeme jméno autora článku příslušnou anotací, může být toto jednoznačně identifikováno, nebude možná záměna s citací a bude možné vyhledávat články od určitého autora. Anotace lze považovat za určitá metadata, která musejí být dobře a jednotně definována. Budeme-li mít definované i vztahy mezi metadaty, bude možné z informací odvozovat další informace na základě metadat a vztahů mezi nimi. Pokud však vztahy nebudou určitým způsobem omezené, odvozování může být příliš náročné, nebo může vlivem velké komplexity a snížení přehlednosti dojít k zavedení nepravdivé (chybné) informace a z té lze potom dle matematické logiky odvodit cokoliv, což způsobí zahlcení databáze s odvozenými informacemi nesmysly. Velmi důležitá je tedy definice metadat, kterou se budu zabývat dále v této práci.

4 Sémantika v sémantickém webu

Sémantiku lze rozdělit na [24]:

1. implicitní,
2. explicitní a neformální
3. explicitní a formální pro zpracování člověkem

4. explicitní a formální pro zpracování strojem

Implicitní sémantika je pouze v myslích lidí. Význam různých termínů je všeobecně známý a pro všechny shodný (např.: adresa). Nevýhodou je, že různé termíny mohou být interpretovány různě dle prostředí, kultury, národních zvyklostí apod. Např. datum „1.2.2010“ může být špatně interpretováno, protože někdo jej interpretuje jako 1. února a někdo jako 2. ledna. Implicitní sémantika je všudypřítomná a pro nezaměnitelné termíny může být využita k hardkódování významu některých výrazů do aplikací.

Explicitní neformální sémantika je neformální textový popis v přirozeném jazyce, který je uveden ve slovnících a v dalších textových popisech (např. v poznámce pod čarou, textové anotaci apod.). Protože komplexita popisu v přirozeném jazyce je příliš velká, i když má stroj k dispozici přidaný textový popis daného termínu a ne pouze jeho kontext, stále má velmi výrazně omezené možnosti popis analyzovat a význam z něj určit. Explicitní neformální sémantika je tedy určena především pro zpracování člověkem.

Pokud je sémantika explicitní a formální, stále může být vhodná pouze pro zpracování člověkem. Popis ve formálním jazyce totiž nemusí být postačující k analýze významu termínu. Příkladem může být sémantika hardkódovaná do aplikace určené ke specifickému účelu. V dané aplikaci je sice význam termínu jasný a dobře popsaný včetně vztahů, ale pokud bychom data přenesli do jiné aplikace, nebude je možné analyzovat.

Explicitní formální sémantika pro zpracování strojem je určená k přímému automatickému zpracování a odvozování. Význam věcí je definován formálním způsobem pomocí vztahů mezi termíny. Např. lze říci, že daný termín je podtřídou třídy zvíře, což je podtřída živočicha, živočich má určité vlastnosti apod. Pokud při definici vycházíme z termínů, které jsou definované v nějaké centrální či všeobecně přístupné databázi, pokud budou data přenesena do jiné aplikace, tato aplikace může odvodit jejich význam bez znalosti prostředí, ze kterého data pocházejí.

V prostředí sémantického webu se vyskytují všechny uvedené druhy sémantiky. Nejdůležitější je však explicitní formální sémantika pro zpracování strojem, kterou je sémantický web specifický a která tvoří základní část sémantického webu.

5 Ontologie

Integrace dat vyžaduje jednotnou definici termínů (např. autor, vydavatel, ...), kategorií (např. lidé, zvířata, ...) a vztahů (např. autor je člověk).

Nejjednodušším řešením je vytvořit slovníky, které přesně definují sémantiku různých termínů. Tyto však často nejsou postačující, protože neobsahují dostatečné definice vztahů. Potřebujeme tedy silnější nástroj, kterým jsou ontologie.

Ontologie definuje koncepty a vztahy využité k popisu a reprezentaci určité oblasti vědění [11]. Ontologie poskytují:

- identifikace objektů,
- konstrukci tříd objektů,
- komplexní klasifikační schémata,
- vyjádření shody a rozdílnosti tříd,
- charakterizace vlastností,
- ...

Jedná se obvykle o rozsáhlé databáze obsahující velké množství objektů, tříd, vlastností a vztahů mezi nimi. Lze je připodobnit ke slovníkům rozšířeným o vztahy mezi termíny a další informace. Obdobně jako slovníky, i ontologie existují obecné nebo se zaměřením na určitou oblast vědění. Propojením různých ontologií lze vytvářet větší celky, které umožní definovat termíny z různých oblastí. Propojování a znovupoužití ontologií je velmi důležité i proto, že tvorba ontologie je extrémně náročná a vyžaduje velmi dobré znalosti z dané oblasti vědění. Jedna menší skupina lidí tedy nikdy nemůže vytvořit ontologii popisující všechno vědění.

Pokud bychom propojili všechny ontologie a doplnili weby o sémantické informace definované pomocí propojených databází ontologií, dosáhneme situace, ve které lze naplnit výše uvedenou ideu sémantického webu.

Příklady existujících ontologií (více příkladů lze nalézt v [26]):

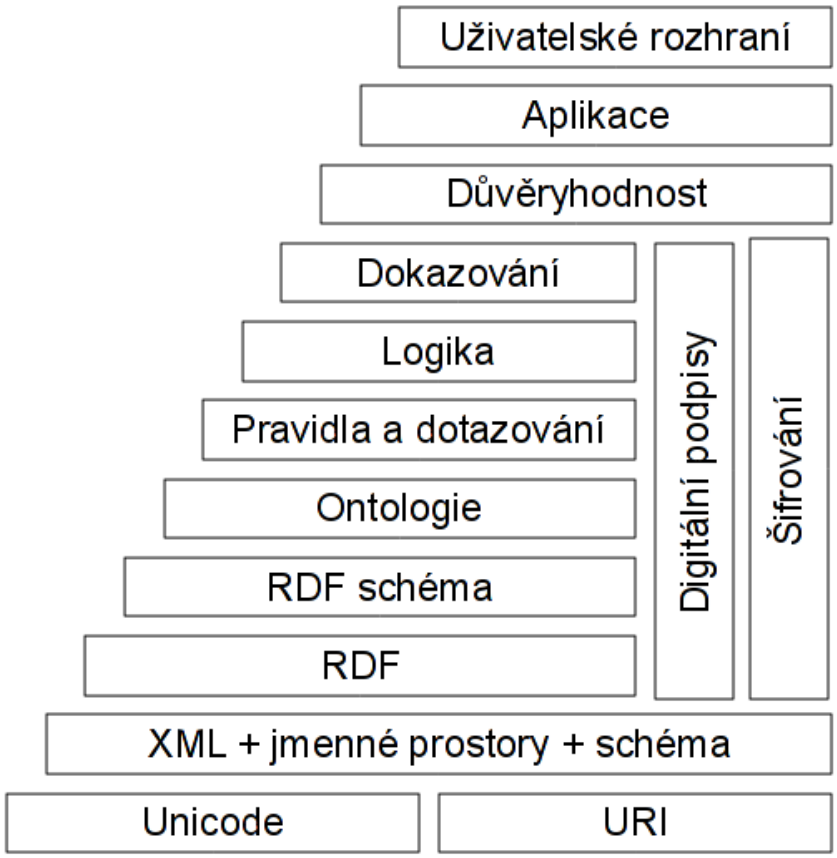
- DBpedia - data z Wikipedie převedená do RDF,
- FOAF (The Friend of a Friend) - lidi, identifikace, kontakty,
- Dublin Core - publikace, muzejní exponáty apod.,
- LOD (Linking Open Data) - propojená síť ontologií.

6 Technologie

Základem sémantického webu jsou technologie umožňující doplnění sémantiky do současného webu. Existuje více různých technologií, které se liší ve způsobu, jakým sémantiku do webu doplňují, ve vyjadřovacích schopnostech a ve složitosti zpracování, která může růst až za hranice možností současné výpočetní techniky.

Další technologie potom slouží k odvozování informací, tvorbě agentů (viz výše) a k dalším účelům. Tyto technologie však dle výše uvedené současné definice sémantického webu slouží spíše k jeho využití, než k jeho tvorbě, a v této práci se jimi nebudu zabývat. Více informací o agentech lze nalézt v [10].

Celkový pohled na vrstvenou architekturu sémantického webu poskytuje diagram, který poprvé prezentoval Tim Berners-Lee a který nyní existuje v mnoha variantách (viz např. [16], [3] a [26]). Tento diagram je vyobrazen na obrázku 1.



Obrázek 1: Diagram vrstvené architektury sémantického webu

Vrstva pro důvěryhodnost (Trust) je horní vrstvou sémantického webu (výše už je pouze aplikace, která jej využívá). Vývoj této vrstvy doposud nepokročil od formulování její vize, kterou je možnost dotazování na důvěryhodnost informací na webu a zajištění pravdivosti informací.

Vrstva logiky slouží k automatickému odvozování informací ze sémantických dat a ontologií. Softwarový agent může s jejím využitím určovat, zda daný zdroj splňuje jeho potřeby apod. V této vrstvě jsou nyní různé konkrétní implementace odvozování, které nejsou obecnou technologií sémantického webu a nebudu je zde tedy popisovat.

Vrstva dokazování slouží k určování, zda jsou odvozené výrazy pravdivé. K tomuto účelu se využívají pravidla z predikátové logiky prvního řádu a dokazovací jazyky [21]. Pro zápis důkazů lze využít i speciální ontologie a syntaxi Notation3.

Digitální podpisy a šifrování jsou potřebné pro zajištění důvěrnosti a autentizace (pokud budou agenti pracovat s citlivými osobními daty uživatelů). Autentizace je důležitá také pro zjištění důvěryhodnosti informací (tvrzení od uživatele, kterému důvěřuji, bude důvěryhodnější).

Následuje podrobnější popis technologií zbývajících vrstev.

6.1 Unicode a URI

Unicode je standard pro kódování znaků všech existujících abeced. Tento standard umožňuje uložení textů v různých jazycích a abecedách v jednotném formátu, což výrazně usnadňuje přenositelnost dat. Více o tomto standardu lze nalézt v [1].

URI (Uniform Resource Identifier) je jednoznačný řetězcový identifikátor prostředku na webu. Jeho syntaxe je aktuálně daná RFC 3986 [6].

6.2 XML

XML (Extensible Markup Language) je obecný flexibilní značkovací jazyk, který umožňuje vytváření značkovacích jazyků pro různé účely a typy dat. Umožňuje popsat strukturu dokumentů, přičemž se nezabývá jejich vzhledem (vizuální prezentací). Byl vyvinut konsorciem W3C jako nástupce dříve využívaného SGML (Standard Generalized Markup Language), přičemž odstranil všechny jeho hlavní nevýhody (např. nedeterminističnost způsobenou možností vynechávání značek).

XML nemá pevný slovník značek, ale je zde možnost definovat značky a strukturu dokumentů pro jednotlivé aplikace. V XML 1.0 se k tomuto účelu využívalo DTD (Document Dype Definition), které se nyní nahrazuje XML schématem. Hlavní výhodou XML schéma oproti DTD je, že metadata jsou popsána v syntaxi XML a ne jiným jazykem.

V sémantickém webu se XML využívá především pro serializaci sémantických informací (viz dále). Jeho hlavními výhodami jsou uniformnost, velká rozšířenost, snadné zpracování (mnoho dostupných knihoven) a možnost definovat různé typy dokumentů. Vzhledem ke striktnější struktuře, než v SGML, je jeho aplikace XHTML (Extensible Hypertext Markup Language) určená pro tvorbu webových stránek lépe strojově zpracovatelná, než HTML (Hypertext Markup Language) a umožňuje tak případné přehlednější a determinističtější přiřazení sémantiky jednotlivým prvkům v dokumentu.

6.3 RDF

Vývoj sémantického webu začal v roce 1998, kdy byla založena pracovní skupina Web Metadata Working Group. Jejím cílem bylo vytvořit framework umožňující sdílení dat a jejich znovupoužití různými aplikacemi. Prvním výsledkem jejího výzkumu bylo vytvoření RDF (Resource Description Framework), který se stal základní technologií sémantického webu.

Hlavní sady doporučení pro RDF následně vydaly 2 pracovní skupiny W3C:

- RDF Core Working Group,
- Web Ontology Working Group.

RDF je datový model ve formě orientovaného grafu, který se skládá z RDF trojic (s,p,o), kde [11]:

s je podmět (subject),

p je přísudek (predicate) či vlastnost (property),

o je předmět (object).

Všechny prvky trojice mohou být zdroje identifikované URI. Prvky, které nejsou zdroje ale pouze datové hodnoty, jsou literály. Vezmeme-li tedy v přirozeném jazyce větu „Praha je město“, lze ji vyjádřit např. RDF trojicí, kde URI XHTML elementu obsahujícího slovo Praha bude předmět, predikát (např. rdf:type) bude vyjadřovat „ je typu“ a předmět bude URI definice města v nějaké databázi ontologií (viz dále). Tímto způsobem lze popsat cokoliv.

RDF lze rozdělit do 3 částí:

- model,
- specifikace syntaxe,
- specifikace schématu.

RDF model udává způsob modelování světa pomocí orientovaného grafu. Jedná se tedy o obecný princip.

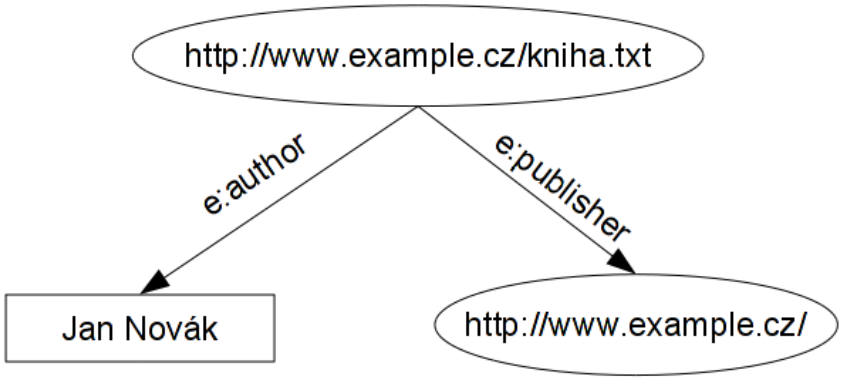
Specifikace syntaxe popisuje syntaxi zápisu RDF trojic. Existuje celá řada různých syntaxí, z nichž nejznámější jsou RDF/XML, RDFa a Turtle (viz dále).

Specifikace schématu definuje využitelné termíny, vztahy a omezení. Jsou zde tedy mimo jiné definovány základní přísudky, které lze využít (např. rdf:type, což je speciální URI zkrácená s využitím jmenných prostorů).

6.3.1 Syntaxe RDF

V této práci uvedu pouze základní principy vybraných nejznámějších syntaxí, detailnější informace o syntaxi RDF lze nalézt v [23] a [25].

RDF graf lze přehledně vyjádřit v grafické podobě. Příklad je na obrázku 2.



Obrázek 2: Příklad RDF grafu

Nejjednodušší z vybraných syntaxí je Turtle, což je přehledný textový formát, vhodný pro čtení člověkem. Detailní popis syntaxe Turtle lze nalézt v [5]. Následuje jednoduchý příklad reprezentace dvou RDF trojic z obrázku 2, kde e je jmenný prostor pro zkrácení zápisu.

```
<http://www.example.cz/kniha.txt>
  e:author     "Jan Novák"@cs ;
  e:publisher <http://www.example.cz/> .
```

Turtle je zjednodušením Notation3, kterou vyvinul Tim Berners-Lee. Alternativním zjednodušením této syntaxe je také N-Triples [21]. Více o Notation3 lze nalézt v [4].

Nejčastěji využívanou syntaxí je RDF/XML. Jedná se o serializaci RDF do formátu XML umožňujícího jednoduché uložení a zpracování. Následuje jednoduchý příklad, pro srovnání opět z obrázku 2.

```
<rdf:Description rdf:about="http://www.example.cz/kniha.txt">
  <e:author xml:lang="cs">Jan Novák</e:auhor>
  <e:publisher rdf:resource="http://www.example.cz/"/>
</rdf:Description>
```

Existují také syntaxe rozšiřující syntaxi současného (X)HTML, které umožňují přidání sémantických informací přímo do současných webových stránek. Nejznámější z těchto syntaxí je RDFa (RDF in Attributes), která značkám XHTML přidává atributy pro reprezentaci grafu RDF. Následuje příklad:

```
<p about="http://www.example.cz/kniha.txt">
Knihu s názvem Příklad napsal
<span property="e:author" lang="cs">Jan Novák</span>
a vydalo nakladatelství
<span rel="e:publisher" resource="http://www.example.cz/">Example</span>.
</p>
```

U syntaxe je důležité zmínit také interní (prázdné) uzly (blank nodes) RDF grafu. Tyto lze vyjádřit ve vech výše uvedených syntaxích. Pokud máme např. větu „Vydavatel je něco, co má název a adresu“, vyjádření slova „něco“ je problematické. Není to zdroj, který by měl adresu ani literál (nejedná se o věc s názvem „něco“). Tento problém lze řešit vytvořením URI pro daný termín, což však vyžaduje generování jednoznačné URI, která je potřeba jenom pro navázání uzlů v RDF a nemá žádný větší význam. Jako alternativní řešení lze tedy využít prázdný uzel v RDF grafu, který bude mít pouze lokální význam pro navázání příslušných uzlů. Následuje příklad v syntaxi RDF/XML:

```
<rdf:Description rdf:about="http://www.example.cz/kniha.txt">
  <e:author xml:lang="cs">Jan Novák</e:auhor>
  <e:publisher>
    <e:pub_name>Example</e>
    <e:address>Brno</e:address>
  </e:publisher>
</rdf:Description>
```

Prázdné uzly mohou být problematické při spojování grafů, kdy dva prázdné uzly (případně i se stejným interním identifikátorem) jsou různé.

6.3.2 RDF Schéma

RDF schéma (RDFS) definuje využitelné termíny, vztahy a omezení. Oficiální název je jazyk pro popis RDF slovníku (RDF Vocabulary Description Language), ale z historických důvodů se stále více využívá název RDF schéma [11].

RDFS definuje:

- zdroje,
- třídy,
- vztahy,
- vlastnosti.

Třídy jsou také zdroje, ale jsou také kolekcemi možných zdrojů [11]. RDFS definuje význam těchto termínů a speciální URI pro jejich označení (obvykle zkracujeme pomocí jmenných prostorů, např. rdfs:Class).

Vztahy jsou definovány mezi zdroji a jsou to:

- typování (typing),
- vytváření podtříd (subclassing).

Typování vyjadřuje, že individuál (zdroj) náleží do nějaké třídy. Nejedná se o obdobu datových typů, protože individuál může náležet do více tříd, ale o vyjádření, že individuál patří do dané skupiny. Vytváření podtříd slouží k vyjádření vztahu, kdy všichni individuálové jedné třídy náleží i do druhé třídy, ale ne naopak. Při získávání RDF trojic z grafu lze pomocí tranzitivity vztahů odvozovat i trojice, které nejsou explicitně uvedeny.

Vlastnosti jsou speciální třídy (rdf:Property) identifikované URI, s jejichž využitím lze definovat typ podmětu a předmětu. Lze specifikovat jejich doménu (typ a rozsah hodnot) a protože jsou současně i zdroji, lze definovat i vlastnosti vlastností. Následuje příklad vlastnosti v syntaxi Turtle převzatý z [11]:

```
:title
  rdf:type     rdf:Property;
  rdfs:domain  :Fikce;
  rdfs:range   rdfs:Literal.
```

Literály mohou mít datový typ dle schématu XML (např. boolean, integer, float apod.) a lze definovat jejich jazyk (cs, en, de, ...), např. (Turtle):

```
:book
  :pages       "150"^^xsd:integer;
  :year        "2010"^^xsd:gYear@cs.
```

Specifikace RDFS obsahuje také řadu předdefinovaných tříd a vlastností, jejichž detailnější popis včetně příkladů lze nalézt v [11]:

- kolekce (collections) (seznamy),
- kontejnery (containers),
- reifikace (zvěcnění - vytvoří předmět z něčeho, co jím není),
- komentář (comment),
- ...

6.4 SPARQL

SPARQL (Simple Protocol And RDF Query Language) [15] je dotazovací jazyk pro dotazování nad RDF grafem. Dotaz se skládá z následujících částí:

- PREFIX - volba jmenného prostoru,
- SELECT nebo CONSTRUCT - definice zobrazovacího formátu,
- WHERE - formulace dotazu,
- SORT BY - třídění výsledků,
- OFFSET - omezení do daného čísla předmětu.

Dotaz tedy může vypadat následovně:

```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

Jednoduchý příklad převzatý z [11]:

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        FILTER(?currency == Kč }
```

Více o SPARQL lze nalézt v [22].

6.5 OWL

OWL (Ontology Web Language) je jazyk pro popis ontologií. OWL je vystavěn na RDFS, ale poskytuje mnohem větší vyjadřovací schopnosti. Umožňuje vyjádřit:

- ekvivalenci a odlišnost individuálů,
- ekvivalenci a odlišnost tříd,
- ekvivalenci a odlišnost vlastností,
- charakter a chování vlastností (symetrie, tranzitivita, inverze, ...),
- identitu (klíče),
- konstrukci tříd průnikem, spojením, doplňkem, ...
- dědičnost,
- datové typy,
- omezení vlastností,
- omezení kardinality,
- ...

Vyjadřovací schopnosti OWL jsou tak velké, že odvozování je příliš složité. S využitím současné výpočetní techniky není možné implementovat odvozování nad komplexními ontologiemi v OWL. Z tohoto důvodu byly vytvořeny profily, které obsahují pouze vybrané vlastnosti OWL a zavádějí různá omezení:

- OWL Full
- OWL DL
- OWL EL
- OWL QL
- OWL RL

OWL Full obsahuje všechny konstrukce OWL bez omezení. Může tak obsahovat mimo jiné i nekonzistence (např. A obsahuje vše z B a zároveň B je doplňkem A), které jsou při zpracování velkým problémem. Dalším problémem je uzavření světa (kontradikce [29]), kdy nevhodně omezíme kardinalitu na 1 (firma má 1 ředitele a pokud je někdo ředitelem, už jím nikdy nemůže být nikdo jiný). Odvozování v OWL Full je nerozhodnutelným problémem a proto se využívá především při teoretickém výzkumu prováděném člověkem a v aplikacích, které potřebují vybrané konstrukce, které jsou pouze v OWL Full (aplikace potom implementuje různá omezení a nepracuje tedy s celou definicí daného jazyka).

OWL DL (OWL Description Logic) definuje určitá omezení:

- termíny RDFS a OWL jsou rezervované a nelze nad nimi konstruovat výrazy,
- vlastnosti uživatelského objektu mohou být pouze individuálové,
- není k dispozici charakterizace datových typů,
- využití stejného symbolu pro individuála a třídu neznačí identitu,
- stejná vlastnost nemůže být využita současně jako objekt i datový typ,
- ...

Velkou výhodou OWL DL však je, že jsou pro něj k dispozici odvozovací algoritmy. Jedná se o speciální typ strukturované logiky prvního řádu a lze pro něj využívat příslušné kompaktní matematické notace. I přes daná omezení lze vybudovat velké ontologie z mnoha oblastí vědění (medicína, energetika, finančnictví apod.). Jedná se tedy o jeden z nejvyužívanějších profilů OWL.

OWL EL je profil určený pro zpracování v polynomiálním čase, který nevyžaduje zpracování komplexních výrazů. Oproti DL zavádí další omezení:

- nelze omezit kardinality,
- méně omezení vlastností,
- nejsou k dispozici některé vlastnosti (inverzní, reflexivní, symetrické, ...),
- není možné vyjádřit rozdílnost tříd,
- ...

OWL QL (OWL Query Language) je určený pro využití s relačními databázemi a umožňuje snadné dotazování s dotazy využívajícími spojování tabulek. Je využitelný pro jednoduché ontologie s velkým množstvím dat. Oproti DL zavádí následující omezení:

- nelze omezit kardinality,
- méně omezení vlastností,
- nejsou k dispozici některé vlastnosti (inverzní, reflexivní, symetrické, ...),
- není možné vyjádřit rozdílnost tříd,
- není k dispozici tranzitivita a řetězení vlastností,
- ...

OWL RL (OWL Rule Language) umožňuje polynomiální rozhodování a zpracování pomocí pravidel (rule engine). Zavádí následující omezení:

- méně omezení kardinality (obvykle pouze 0/1),
- méně omezení vlastností,
- omezení výrazů nad třídami (sjednocení, průnik, ...),
- není k dispozici omezení datových typů,
- ...

Stále však zůstává celá řada vlastností:

- identita tříd, individuálů a vlastností,
- podtřídy, podvlastnosti,
- domény, rozsahy,
- omezené sjednocení a průnik tříd,
- charakterizace a řetězení vlastností,
- klíče,
- ...

Protože OWL RL využívá především principy RDFS a lze jej zpracovat pomocí jednoduchých pravidel, implementace je jednoduchá a málo výpočetně náročná. Je tedy možné zpracovávat i velká množství dat.

6.5.1 Syntaxe OWL

Nejvyužívanější syntaxí OWL je RDF/XML, což je současně oficiální syntaxe pro výměnu informací. Tato však může být velmi upovídaná (verbose). Existují proto i alternativní syntaxe, jako např. OWL/XML, funkcionální nebo Manchester. Následuje zjednodušený příklad syntaxe Manchester převzatý z [16]:

```
Individual: my:Kč
Individual: my:$
Class: my:Currency EquivalentTo {my:Kč my:$}
Class: my:Listed_Price that p:currency only my:Currency
```

Pro srovnání uvedu stejný příklad v Turtle:

```
my:Kč rdf:type owl:Thing.
my:$ rdf:type owl:Thing.
my:Currency rdf:type owl:Class;
  owl:oneOf (my:Kč my:$).
my:Listed_Price rdf:type owl:Class;
  rdfs:subClassOf [
    rdf:type           owl:Restriction;
    owl:onProperty     p:currency;
    owl:allValuesFrom  my:Currency
  ].
```

Více informací o OWL lze nalézt v [18].

6.6 RIF

RIF (Rule Interchange Format) je formát pro popis pravidel. Některé skutečnosti lze pomocí OWL popsat jenom velmi složitě, zatímco pomocí pravidel stačí několik málo pravidel. Pravidla tedy mohou být využita jako alternativa k OWL. Pravidla mají dlouhou historii v jazyce Prolog, čehož lze s výhodou využít při tvorbě ontologií.

RIF Core je jádro jazyka RIF. Obsahuje definice základních direktiv (např. pro práci s URI), sekvence implikací apod. Jsou zde zabudované datové typy, predikáty, notace lokálních jmen (obdoba prázdných uzlů), bezpečnostní omezení proti zacyklení (forward chaining) apod. Pravidla nejsou omezena na vyjadřování RDF trojic a vztah v nich může zahrnovat i více než 2 entity.

RDF strom lze vyjádřit i v RIF. Trojice (s,p,o) se vyjádří jako s[p->o], „s rdf:type a“ se vyjádří pravidlem A # B apod. Více informací lze nalézt v [8].

6.7 DAML + OIL

DAML (DARPA Agent Markup Language) je jazyk pro ontologie a odvozování vyvinutý agenturou DARPA. Je založen na RDF a rozšiřuje RDFS. Umožňuje vyjádřit:

- inverze,
- jednoznačné a unikátní vlastností,
- seznamy,
- omezení,
- párované seznamy rozdílů,
- datové typy,
- ...

Jednoznačná vlastnost říká, že pokud předměty vlastností jsou stejné, podměty jsou ekvivalentní, např. [21]:

```
foaf:mbox rdf:type daml:UnambiguousProperty .
:x foaf:mbox  .
:y foaf:mbox  .
```

implikuje:

```
:x daml:equivalentTo :y .
```

Při odvozování je v DAML důležité vyjádření ekvivalence. Příklad:

```
:Auto cs:výkon "200kW" .
```

Máme-li výše uvedenou třídu Auto s českým popisem parametrů, mezi kterými je i uvedený výkon, můžeme pomocí DAML českým výrazům přiřadit ekvivalentní anglické:

```
cs:výkon daml:equivalentTo en:power .
```

Agent potom může odvodit anglické parametry:

```
:Auto en:power "200kW" .
```

Kdybychom překlad realizovali pomocí obyčejného slovníku, nemusel by zvolit správný výraz a překlad by mohl být i nesmyslný. Využití DAML (a obecně sémantických informací) zajistí, že budou využity správné významově ekvivalentní výrazy.

Více informací o DAML lze nalézt v [20].

Ontologie OIL (Ontology Inference Layer) se skládá z:

- definice slotů (slot-def),
- definice tříd (class-def).

Definice slotu popisuje binární relaci mezi dvěma entitami. Definice tříd asociuje název třídy s jejím popisem a skládá se z následujících částí:

- typ definice (definition type):
  - defined – třída je kompletně specifikována,
  - primitive – jsou uvedeny pouze omezující podmínky, které nedostačují k plnému definování třídy (určení členství).
- omezení slotu (slot constraint) – omezuje možné hodnoty, které může mít slot, když se aplikuje na instanci třídy (skládá se z názvu, typu hodnoty a hodnoty),
- podtřída (subclass-of) – relace definované třídy s jedním nebo více výrazy nad třídami (název třídy, omezení slotu nebo jejich booleovská kombinace).

Následuje zjednodušený příklad z [9] (definice slotů jsou z prostorových důvodů vynechány):

```
class-def animal  % zvířata jsou třída
class-def plant  % rostliny jsou třída
  subclass-of NOT animal  % rostliny jsou disjunktní od zvířat
class-def tree
  subclass-of plant  % stromy jsou podtřída rostlin
class-def branch
  slot-constraint is-part-of  % větve jsou součástí stromů
    has-value tree
class-def defined carnivore
  subclass-of animal  % masožravci jsou zvířata
  slot-constraint eats  % která jí pouze jiná zvířata
    value-type animal
class-def defined herbivore % býložravci jsou zvířata
  subclass-of animal
  slot-constraint eats  % která jí pouze rostliny nebo části rostlin
    value-type plant
    OR (slot-constraint is-part-of has-value plant)
```

DAML+OIL vznikl jako spojení jazyků DAML-ONT, který je součástí DAML, a OIL [13]. Je navržen k popisu struktury domény a využívá objektově orientovaný přístup. DAML+OIL je ekvivalentem deskriptivní logiky. Ontologie sestává ze sady axiomů, které jsou předpoklady. Třídy mohou být reprezentovány nejenom URI, ale také výrazy, pro jejichž konstrukci je k dispozici celá řada konstruktorů. Vyjadřovací síla jazyka je dána podporovanými konstruktory a druhy axiomů. Konstruktory tříd DAML+OIL jsou:

- intersectionOf,
- unionOf,
- complementOf,
- oneOf – definice výčtem členů,
- toClass – pro omezení slotu,
- hasClass – pro omezení slotu,
- hasValue – kombinace oneOf a hasClass,
- minCardinalityQ,
- maxCardinalityQ,
- cardinalityQ.

Axiomy DAML+OIL jsou:

- subClassOf,
- sameClassAs,
- subPropertyOf,
- samePropertyAs,
- disjointWith,
- sameIndividualAs,
- differentIndividualFrom,
- inverseOf,
- transitiveProperty,

- uniqueProperty,
- unambiguousProperty.

Protože subClassOf a sameClassAs mohou být aplikovány na libovolný výraz, vyjadřovací schopnosti DAML+OIL jsou větší, než u jiných obbdobných jazyků, které na levé straně takových axiomů povolují pouze atomické názvy (omezení cyklů).

Více informací o DAML+OIL lze nalézt v [13].

6.8 Mikroformáty

Nechceme-li do webu doplňovat sémantiku přímo pomocí RDFa, můžeme využít mikroformát a přímo konvertovat obsah webu do RDF. Tato technologie nebyla vyvinuta přímo pro sémantický web, ale vyvíjí ji samostatná komunita. Hlavním principem je znovupoužití (X)HTML elementů a atributů pro doplnění metainformací (typicky např. atributy title a class). Pokud se například dohodne, že class se využije pro názvy, title pro hodnoty apod, je možné sémantické informace uložit přímo do (X)HTML bez přidávání dalších atributů či elementů. K transformaci do RDF lze potom využít transformaci XSLT [27].

Nevýhodou je, že transformace na RDF je specifická pro každý mikroformát (dle konvence, kam se u něj uloží jaké informace). Znovupoužití atributů přináší také další nevýhody spojené s tím, že využíváme atributy původně určené k jinému účelu (title se zobrazí při najetí myší na objekt, class je spojen se styly apod.).

6.9 GRDDL

GRDDL (Gleaning Resource Descriptions from Dialects of Languages) poskytuje jednotný model pro transformaci XHTML do RDF. Název je možné přeložit do češtiny jako „sbírání zdrojových popisů z dialektů jazyků“. Je to obdoba mikroformátů, ale umožňuje transformaci libovolných XML dat do RDF přidáním několika elementů a atributů umožňujících nalezení správné transformace:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head profile="http://www.w3.org/2002/12/cal/cardcaletc
                 http://purl.org/NET/erdf/profile">
    ...
    <link rel="transformation" href="http://www.w3.org/2002/12/cal/glean-hcal.xsl"/>
    ...
  </head>
  ...
```

Více o GRDDL lze nalézt v [7].

6.10 SKOS

SKOS (Simple Knowledge Organization System) slouží k reprezentaci a sdílení klasifikací, slovníků, tezaurů apod. Definuje třídy a vlastnosti pro přidání těchto struktur do RDF stromu, které lze rozdělit do následujících skupin [11]:

- základní popis (Concept, ConceptScheme, ...),
- popisky (prefLabel, altLabel, ...),
- dokumentace (definition, historyNote, ...),
- sémantické relace (broader, narrower, related, ...),
- kolekce (Collection, OrderedCollection, ...),
- mapování konceptů (broadMatch, narrowMatch, ...).

Následuje zkrácený příklad ze [11]:

```
<http://id.loc.gov/authorities/sh85061165#concept>
  a               skos:Concept;
  skos:prefLabel  "Historical Fiction"@en;
  skos:broader    <http://id.loc.gov/authorities/sh85048050#concept>;
  ...
```

SKOS poskytuje mezičlánek mezi tištěnými slovníky a sémantickým webem. Více informací lze nalézt v [14].

6.11 POWDER

POWDER (Protocol for Web Description Resources) umožňuje definovat popisné zdroje. Sady zdrojů jsou definovány omezeními URI (např. shoda začátku). K tomuto účelu je v něm obsaženo rozšíření RDF o shodu a neshodu s regulárním výrazem. Každý zdroj musí mít uvedené také atributy (attribution), aby byl připravený pro využití vestavěné autentizace. Následuje jednoduchý příklad s jedním popisným zdrojem (značka <dr>) převzatý z [2]:

```
<powder xmlns="http://www.w3.org/2007/05/powder#"
        xmlns:ex="http://example.org/vocab#">
  <attribution>
    <issuedby src="http://authority.example.org/company.rdf#me" />
    <issued>2007-12-14T00:00:00</issued>
  </attribution>
  <dr>
    <iriset>
      <includehosts>example.com</includehosts>
    </iriset>
    <descriptorset>
      <ex:color>red</ex:color>
      <ex:shape>square</ex:shape>
      <displaytext>Cokoliv na example.com je červené a čtveraté</displaytext>
      <displayicon src="http://authority.example.org/icon.png" />
    </descriptorset>
  </dr>
</powder>
```

POWDER se obvykle využívá pro licencování, označení přístupnosti či jiné značení obsahu. Více informací lze nalézt v [2].

7 Příklad vytvoření jednoduché ontologie

V této kapitole uvedu příklad vytvoření jednoduché ontologie naší fakulty. Jednoduchosti příkladu využiji k přehlednému vysvětlení postupu tvorby ontologie a základních konstrukcí v OWL. Ontologii budu vytvářet v editoru Protégé [19], jehož funkcionalitu průběžně popíšu.

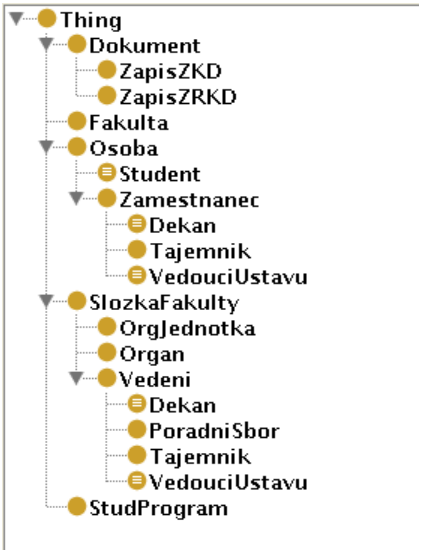
7.1 Určení základních tříd

Prvním krokem při tvorbě ontologie je určení základních tříd (Classes) objektů domény. Protože při objektově orientovaném přístupu jsou vše třídy a každá třída je potomkem jiné třídy, musí existovat třída na vrcholu hierarchie. Touto třídou je třída `Thing`, ze které dědí základní třídy. Na naší fakultě lze vyjmenovat např. následující základní třídy:

- Fakulta (abychom do ontologie mohli zahrnout i samotnou fakultu),
- Složka fakulty,
- Osoba,
- Studijní program,
- Dokument.

Složky fakulty můžeme dále rozdělit na podtřídy:

- Vedení,
- Organizační jednotka,
- Orgán.



Obrázek 3: Výsledná hierarchie tříd

Dále můžeme rozčlenit i vedení, osoby můžeme rozdělit na studenty a zaměstnance apod. Vztah mezi třídou a nadtřídou ve výsledném RDF grafu je „je“ (is-a). Výsledná hierarchie tříd je na obrázku 3.

OWL je založen na předpokladu otevřeného světa (Open World Assumption), což znamená, že pokud něco není explicitně omezeno či vyloučeno, je to možné. Pokud tedy např. neuvedeme, že organizační jednotka není to stejné jako orgán, připouštíme, že se tyto složky fakulty mohou shodovat, což není pravda. Rozdílnost se vyjadřuje pomocí seznamů rozdílných tříd (Disjoint classes). V editoru Protégé se tento seznam zadává u některé z rozdílných tříd, ale vygeneruje se z něj obecný axiom, takže tento seznam ovlivní i další uvedené třídy. Pokud např. chceme vyjádřit, že poradní sbor nemůže být současně jinou složkou vedení (není to osoba, ale skupina lidí), uvedeme u něj tři seznamy, přičemž v každém bude jedna ze zbývajících složek vedení.

Rozdílnost tříd specifikujeme i u dalších tříd, ale ne u zaměstnance a studenta, protože student doktorského studia patří do obou těchto tříd současně.

Pokud chceme uvést, že třída má dvě nadtřídy (např. děkan je podtřída vedení a současně zaměstnanec), uvedeme je do seznamu nadtříd u dané třídy a to buď jako součást výrazu nad třídami (Class expression) v rámci již existující položky, nebo do samostatné položky.

Výraz nad třídami v seznamu nadtříd tvoří prázdný uzel v RDF grafu, tedy třídu, která je specifikovaná pouze svými nadtřídami a vlastnostmi, ale nemá žádné označení. Ve výrazu nad třídami můžeme využít logické operátory a vyjádřit, že třída je podtřídou obou nadtříd současně (Vedení and Zaměstnanec), nebo pouze jedné z nich (Vedení or Zaměstnanec). Můžeme také využít vlastností (viz dále) a třídu specifikovat pomocí vztahů s jinými třídami.

7.2 Definice vlastností

Nyní přejdeme k definici objektových vlastností (Object properties), tedy vztahů mezi dvěma třídami či individuály. Kromě těchto existují ještě datové vlastnosti, kterými se budeme zabývat později. Obdobně jako máme hierarchii tříd, máme i hierarchii objektových vlastností. Na vrcholu této hierarchie je vlastnost `topObjectProperty`. Nejprve tedy určíme základní vlastnosti, které budou jejími podvlastnostmi (specifičtějšími vlastnostmi). Pro výše uvedené třídy to jsou:

- je součástí,
- je členem,
- je vedoucím,
- studuje,
- spolupracuje,
- ví o (něčem).

K některým vlastnostem můžeme definovat také inverzní vlastnosti, tedy např.:

- má součást,
- má člena,
- má vedoucího.

Inverzní vlastnosti definujeme tak, že je nejprve definujeme jako běžné vlastnosti, a potom je uvedeme do seznamů inverzních vlastností u daných vlastností. Definujeme tedy např. vlastnost „má člena“ a potom u vlastnosti „ je členem“ přidáme položku do seznamu inverzních vlastností. Editor Protégé nám automaticky doplní odpovídající informaci ke druhé z daných vlastnosti.

Vlastnost „ je vedoucí“ je tranzitivní, tedy pokud řekneme, že vedoucím pana X je pan Y a vedoucím Y je Z, pak platí, že vedoucím X je nepřímo i Z. Vlastnost tedy v Protégé označíme příslušným zaškrtávacím políčkem.

Jako specifičtější podtřídy vlastnosti „ je vedoucím“ můžeme přidat:

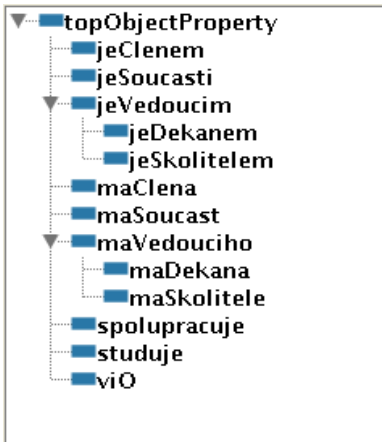
- je děkanem,
- je školitelem.

K uvedeným vlastnostem definujeme i inverzní vlastnosti jako podtřídy „má vedoucího“. Následně můžeme říci, že fakulta může mít v dané chvíli pouze jednoho děkana a „má děkana“ je tedy funkcionální vlastnost. Pro funkcionální vlastnost platí, že v daném vztahu (mezi třídami) mohou být pouze dva individuálové (viz dále), tedy např. zendulka a FIT. Vlastnost „ je děkanem“ potom bude nejenom inverzní, ale i inverzní funkcionální, protože nemůže být děkanem dvou fakult současně.

Vlastnosti mohou být také symetrické (např. pokud vedení spolupracuje s orgány fakulty, tento vztah platí i ve druhém směru) a reflexivní, což je vlastnost, kterou má individuál i sám se sebou (např. „ví o“, kdy zaměstnanec ví o zápisu z kolegia děkana i sám o sobě). Asymetrická vlastnost neplatí ve druhém směru (pokud je zaměstnanec děkanem fakulty, nemůže být fakulta děkanem zaměstnance) a ireflexivní vlastnost individuál nemůže mít sám se sebou (zaměstnanec nemůže být děkanem sám sebe, ale pouze fakulty).

Pokud potřebujeme vyjádřit, že vztah může být pouze mezi individuály určitého typu, můžeme využít domény (domains) a rozsahy (ranges). Každá vlastnost spojuje individuála z určité domény s individuálem z určitého rozsahu. Např. vlastnost „ je děkanem“ tedy bude mít jako doménu třídu Zaměstnanec a jako rozsah třídu Fakulta.

Výsledná hierarchie vlastností je na obrázku 4.



Obrázek 4: Výsledná hierarchie vlastností

7.3 Vztahy mezi třídami

Když máme definované vlastnosti, můžeme je využít k definování vztahů mezi třídami i k definování samotných tříd. Nejprve je však nutné vysvětlit pojem kardinalita. Každý vztah může mít určitou kardinalitu, což je informace, která udává, kolik individuálů do daného vztahu vstupuje. Možné hodnoty jsou:

- existuje (some) - např. pokud existuje organizační jednotka, které je zaměstnanec vedoucím, jedná se o vedoucího ústavu (`jeVedoucim some OrgJednotka`),
- pro každý (only) - např. každý člen orgánu fakulty je osoba (`maClena only Osoba`),
- minimálně (min) - např. zaměstnanec ví minimálně o 1 zápisu z KD,
- exaktně (exactly) - např. student studuje právě jeden studijní program,
- maximálně (max) - např. student má maximálně 1 školitele.

Nyní můžeme vyjádřit, že vedoucí ústavu je vedoucím nějaké organizační jednotky. Tuto informaci vložíme jako nadtřídu třídy `VedouciUstavu`, kterou vytvoříme jako omezení objektu (object restriction), což je v podstatě výraz nad třídami, který tvoří prázdný uzel v RDF grafu. Tímto jsme vyjádřili nutnou podmínku pro to, abychom mohli říci, že individuál je vedoucím ústavu. Tato podmínka však není postačující, protože je rovněž nutné, aby vedoucí ústavu byl zaměstnanec. I když uvedeme obě nadtřídy, stále však není jisté, zda se jedná o postačující podmínky (předpoklad otevřeného světa, viz výše). Je tedy nutné explicitně vyjádřit, že podmínky jsou postačující. Toto lze provést tak, že vytvoříme výraz nad třídami, který umístíme do seznamu ekvivalentních tříd u třídy `VedouciUstavu`, nebo lze využít funkce editoru Protégé a konvertovat třídu na definovanou (Defined), což není nic jiného, než převedení seznamu nadtříd (podmínek) do výrazu v ekvivalentních třídách. Výsledek konverze je vyobrazen na obrázku 5.

Obrázek 5: Konverze primitivní třídy na definovanou

Definovaná (defined) třída je taková, u které lze na základě uvedených informací určit, kteří individuálové jsou jejími členy. Opakem je třída **primitivní** (primitive), u které jsou uvedeny pouze nutné podmínky, nikoliv však postačující.

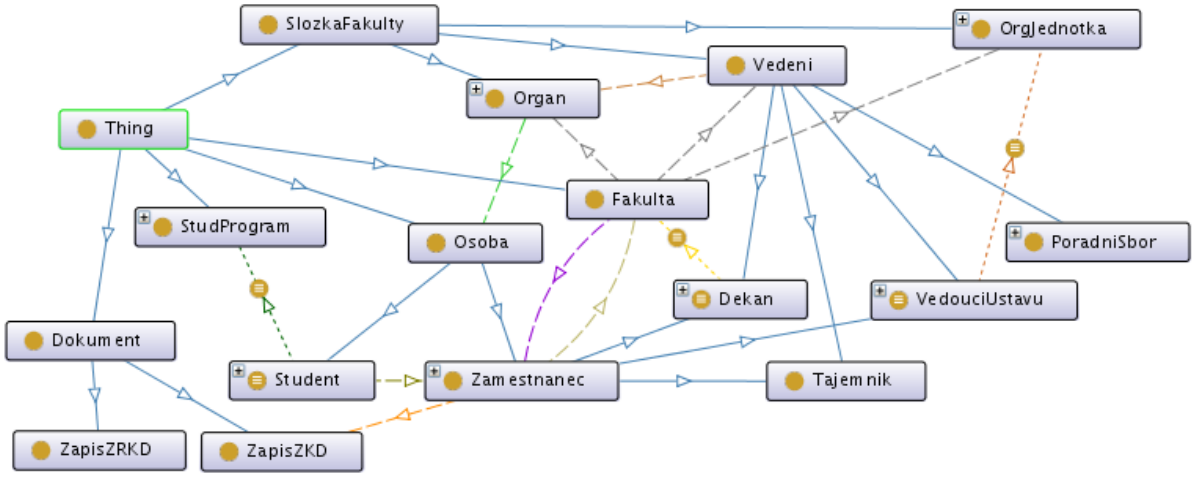
Mezi oběma variantami lze třídu v Protégé automaticky konvertovat, je však nutné dát pozor na ztrátu informací. Např. pokud uvedeme minimální nutné podmínky pro vedoucího ústavu a následně třídu konvertujeme na definovanou, neznamená to, že nemůžeme definovat další vlastnosti, např. to, že vedoucí ústavu je podtřídou vedení. Pokud potom provedeme zpětnou konverzi, mezi nutnými podmínkami stále máme, že vedoucí ústavu je podtřídou vedení. Další konverze potom z této podmínky udělá součást nutných a postačujících podmínek. Před posledními dvěma konverzemi tedy bylo možné odvodit, že každý zaměstnanec, který je vedoucím organizační jednotky, je vedoucím ústavu. Po konverzích se však tato informace ztratila, protože zaměstnanec by ještě musel být podtřídou vedení, což je sice nutná podmínka, ale nikde není explicitně uvedena a nyní již vedoucí ústavu nelze automaticky odvodit. Z uvedeného příkladu je rovněž vidět, že pro odvozování informací je výhodné, aby nutné a postačující podmínky byly současně minimální.

Další důležitou konstrukcí je tzv. axiom pokrytí (covering axiom). Máme-li např. vedení, které má podtřídy reprezentující jeho jednotlivé složky, nevíme, zda neexistuje ještě další složka vedení, která není uvedena. Pokud chceme vyjádřit, že všechny složky jsou uvedeny a vedení nic jiného neobsahuje, využijeme axiom pokrytí, což je nadtřída tvořená výrazem, ve kterém jsou jednotlivé podtřídy s logickými spojkami „nebo“ (tedy individuál patří do podtřídy A nebo B nebo ...). Zajímavé na této konstrukci je, že nadtřídou je třída tvořená podtřídami (cyklus v RDF grafu).

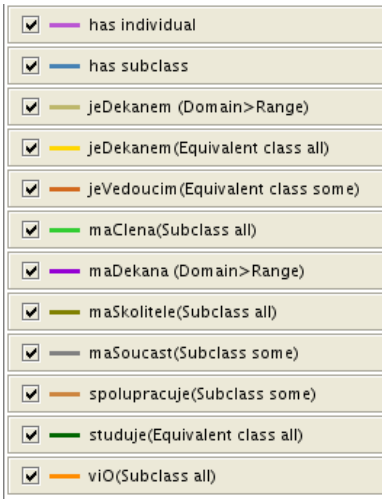
Dalším axiomem je axiom uzavření (closing), který umožňuje vyjádřit, že ve vztahu nemůže být žádný jiný individuál, než individuálové z uvedené třídy (např. členem orgánu fakulty může být pouze osoba). Při využívání tohoto

axiomu je nutná obezřetnost, abychom nezavedli příliš velké omezení, které by mohlo způsobit problémy (mohl by být členem někdo, kdo není osoba, třeba robot?).

Na obrázku 6 je vyobrazen graf tříd výsledné ontologie a vztahů mezi nimi a na obrázku 7 je legenda k tomuto grafu.



Obrázek 6: Graf výsledné ontologie ze zásuvného modulu `OntoGraf`



Obrázek 7: Legenda ke grafu výsledné ontologie z obrázku 6

7.4 Datové vlastnosti

Datové vlastnosti (data properties) spojují třídu či individuála s nějakým literálem (hodnotou datového typu ze schématu XML). Můžeme pomocí nich vyjádřit datové omezení třídy (pokud je individuál členem třídy, má vlastnost tohoto typu) a to, že individuál má konkrétní hodnotu dané vlastnosti.

Datové vlastnosti se definují samostatně a mají vlastní hierarchii, na jejímž vrcholu je `topDataTypeProperty`. Obdobně jako u objektových vlastností u nich můžeme definovat rozdílnost, domény a rozsahy. Rozsahy se zde definují jako datové typy ze schématu XML nebo jako výraz nad těmito typy.

V uvedeném příkladu jsem definoval pouze několik vlastností pro přehlednou ukázku. Pokud bychom měli definovat např. všechny vlastnosti osob, které se na fakultě využívají, překročilo by to rozsah této práce. Na obrázku 8 je uvedena vytvořená hierarchie datových vlastností.

Obrázek 8: Výsledná hierarchie datových vlastností

Nyní můžeme vyjádřit např. to, že každá osoba musí mít datum narození. Toto provedeme tak, že osobě přidáme nadtřídu s datovým omezením (Data type restriction) s vlastností `datumNarozeni` a exaktní kardinalitou 1.

7.5 Individuálové

Individuálové jsou konkrétními instancemi tříd (např. ve třídě `Fakulta` bude individuál FIT). Individuály označujeme jejich URI, které končí názvem (označením) individuála, který dle konvence začíná malým písmenem.

Do našeho příkladu přidáme individuály pro studijní programy, poradní sbory, orgány, organizační jednotky a děkana. Studijním programům potom můžeme přiřadit datovou vlastnost s celkovým počtem kreditů. Protože jsme zadali, že počet kreditů je z rozsahu `int`, musíme při zadávání hodnoty zadat i datový typ (jinak by vznikla nekonzistence).

Následně přidáme individuály pro vedoucí ústavů. Vedoucímu UPSY explicitně přidáme nadtřídu `VedouciUstavu`, vedoucím ostatních ústavů pouze přidáme vlastnost (object property assertion) `jeVedoucim`, kterou je přiřadíme k jednotlivým individuálům organizačních jednotek. U vedoucího UIFS tuto vlastnost vynecháme a místo ní využijeme inverzní vlastnost `maVedouciho` pro ústav.

Pokud nyní v Protégé spustíme klasifikaci, dojde k odvození vztahů a informací, které nejsou explicitně uvedeny. Ve třídě `VedouciUstavu` tedy budou všichni vedoucí ústavů, i když jsme tuto informaci explicitně neuvedli.

Informace, které jsme explicitně zadali, se nazývají předpokládané (asserted) a informace, které byly automaticky odvozeny, se nazývají odvozené (inferred). Protégé tyto informace rozlišuje uvedením do samostatného seznamu nebo barvou. Příklad výsledku odvozování ve třídě `VedouciUstavu` je na obrázku 9.

Obrázek 9: Předpokládané a odvozené informace o třídě `VedouciUstavu`

Z odvozování je vidět strojové zpracování informací, které je jedním ze základních principů sémantického webu. Pokud bychom vytvořili webovou stránku a doplnili ji o sémantické informace, do těchto informací můžeme zahrnout, že daná informace je např. o vedoucím ústavu. Strojově lze potom odvodit, že tato informace je informací o nějakém zaměstnanci. Pokud uvedeme i ústav, lze odvodit konkrétního zaměstnance (individuála). Naopak pokud uvedeme, že nějakou informaci napsal daný zaměstnanec, lze odvodit, že se jedná o informaci od vedoucího ústavu.

Pokud bychom měli mnohem podrobněji zpracovanou ontologii, bylo by možné odvozovat velké množství informací. Napojením takové ontologie na velkou síť propojených ontologií bychom získali další možnosti odvozování a je možné, že bychom mohli vytvořit i agenta, který by byl schopen plnit jednoduché úkoly jako např. nalézt literaturu k předmětům, které učí pan děkan, což žádný běžný vyhledávač přímo nedokáže. Jednalo by se však o práci pro celý tým lidí, která by mnohonásobně přesáhla rozsah této práce. Více informací o tvorbě ontologií v editoru Protégé lze nalézt v [28] a [12].

8 Závěr

Sémantický web je rozšířením klasického webu, které je stále ve vývoji. Jeho současná definice se prozatím liší od původní vize, ve které softwaroví agenti na webu dělají za člověka běžné úkony spojené s vyhledáváním a zpracováním informací. Sémantický web je nyní webem, ve kterém jsou obsaženy sémantické informace pro zpracování stroji. Některé technologie využívané pro tvorbu sémantického webu dosud nebyly standardizovány, některé jsou v intenzivním vývoji a některé jsou stále pouze ve formě vizí do budoucnosti. V současné době se proto využívá především v akademické a výzkumné oblasti.

Základní technologií sémantického webu je RDF, což je grafová struktura pro reprezentaci vztahů mezi třídami objektů reálného světa i jednotlivými individuály. Existují různé syntaxe a rozšíření tohoto jazyka, která zvyšují jeho vyjadřovací schopnosti.

Aby bylo možné definovat různé třídy objektů pomocí vztahů s jinými třídami, je potřeba mít slovníky pojmů (tříd) s přesnými definicemi jejich významů a se vztahy mezi jednotlivými pojmy. K tomuto účelu slouží ontologie, které jsou velmi důležitou součástí sémantického webu.

Aby mohl být sémantický web běžně využíván, je třeba, aby byly do současného webu doplněny sémantické informace. Až budou standardizovány všechny potřebné technologie, velké firmy začnou využívat jejich možností a doplní sémantické informace do svých webů, konkurenční boj posune vývoj sémantického webu vpřed. Po doplnění sémantických informací se dostaneme do stádia, ve kterém bude možné zahájit vývoj agentů naplňujících původní vizi sémantického webu tak, jak ji vymyslel Tim Berners-Lee.

Literatura

[1] Alvestrand, H.; aj.: About the Unicode Standard. In *The Unicode Consortium*, The Unicode Consortium, 2009, [Online; navštíveno 14.4.2010].
URL http://www.unicode.org/standard/standard.html

[2] Archer, P.; Smith, K.; Perego, A.: Protocol for Web Description Resources (POWDER): Description Resources. In *W3C Recommendation*, W3C, 2009, [Online; navštíveno 14.4.2010].
URL http://www.w3.org/TR/powder-dr/

[3] Bartoš, P.: GlobalSemantic.Net. 2008, [Online; navštíveno 14. 4. 2010].
URL http://sites.google.com/a/globalsemantic.net/gsn/Home

[4] Berners-Lee, T.: Notation 3. In *An readable language for data on the Web.*, W3C, 2006, [Online; navštíveno 14.4.2010].
URL http://www.w3.org/DesignIssues/Notation3

[5] Berners-Lee, T.; Beckett, D.: Turtle - Terse RDF Triple Language. In *W3C Team Submission*, W3C, 2008, [Online; navštíveno 14.4.2010].
URL http://www.w3.org/TeamSubmission/turtle/

[6] Berners-Lee, T.; aj.: Uniform Resource Identifier (URI): Generic Syntax. In *IETF Tools*, IETF, 2005, [Online; navštíveno 14.4.2010].
URL http://tools.ietf.org/html/rfc3986

[7] Connolly, D.: Gleaning Resource Descriptions from Dialects of Languages (GRDDL). In *W3C Recommendation*, W3C, 2007, [Online; navštíveno 14.4.2010].
URL http://www.w3.org/TR/grddl/

[8] Dean, M.; aj.: RIF. In *RIF Working Group*, W3C, 2010, [Online; navštíveno 14.4.2010].
URL http://www.w3.org/2005/rules/wiki/RIF_Working_Group

[9] Decker, S.; Melnik, S.; van Harmelen, F.; aj.: THE SEMANTIC WEB: The Roles of XML and RDF. In *Knowledge Networking*, VU University Amsterdam, 2000, [Online; navštíveno 14.4.2010].
URL http://www.few.vu.nl/~frankh/postscript/IEEE-IC00.pdf

[10] Hendler, J.: Agents and the Semantic Web. In *IEEE Intelligent Systems Journal*, University of Maryland, Březen 2001, [Online; navštíveno 14. 4. 2010].
URL http://www.cs.umd.edu/~hendler/AgentWeb.html

[11] Herman, I.: Tutorial on the Semantic Web. W3C, Duben 2010, [Online; navštíveno 14. 4. 2010].
URL http://www.w3.org/People/Ivan/CorePresentations/SWTutorial/Slides.pdf

[12] Horridge, M.; Moulton, G.; Stevens, R.; aj.: *A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools*. The University Of Manchester, první vydání, Březen 2010, [Online; navštíveno 18. 5. 2010].
URL http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_2.pdf

[13] Horrocks, I.: DAML+OIL: a Description Logic for the Semantic Web. In *Department of Computer Science*, University of Manchester, 2002, [Online; navštíveno 14.4.2010].
URL http://sites.computer.org/debull/A02mar/hmain-a.ps

[14] Isaac, A.; aj.: SKOS Simple Knowledge Organization System - Home Page. In *W3C Semantic Web Activity*, W3C, 2009, [Online; navštíveno 14.4.2010].
URL http://www.w3.org/2004/02/skos/

[15] Martin, S.: Sémantický web: Google se učí kombinovat. In *Chip.cz online*, Burda Praha, spol. s.r.o., 2009, [Online; navštíveno 14.4.2010].
URL http://old.chip.cz/cs/clanky/semanticky-web-google-se-uci-kombinovat.html

[16] Matthews, B.: Semantic Web Technologies. In *CCLRC Rutherford Appleton Laboratory*, JISC, 2005, [Online; navštíveno 14.4.2010].
URL http://www.jisc.ac.uk/uploaded_documents/jisctsw_05_02bpdf.pdf

[17] Matulík, P.; Pitner, T.: Sémantický web a jeho technologie. 2004, [Online; navštíveno 14. 4. 2010].
URL http://www.ics.muni.cz/zpravodaj/clanky_tisk/296.pdf

[18] McGuinness, D.; van Harmelen, F.: OWL Web Ontology Language. In *W3C Recommendation*, W3C, 2004, [Online; navštíveno 14.4.2010].
URL http://www.w3.org/TR/owl-features/

[19] Musen, M.; aj.: The Protégé Ontology Editor and Knowledge Acquisition System. 2010, [Online; navštíveno 18. 5. 2010].
URL http://protege.stanford.edu/

[20] Pagels, M.; aj.: The DARPA Agent Markup Language Homepage. 2006, [Online; navštíveno 14. 4. 2010].
URL http://www.daml.org/

[21] Palmer, S.: The Semantic Web: An Introduction. 2009, [Online; navštíveno 14. 4. 2010].
URL http://infomesh.net/2001/swintro/

[22] Prud'hommeaux, E.; Seaborne, A.: SPARQL Query Language for RDF. In *W3C Recommendation*, W3C, 2008, [Online; navštíveno 14.4.2010].
URL http://www.w3.org/TR/rdf-sparql-query/

[23] Semantic Web. 2009, [Online; navštíveno 14. 4. 2010].
URL http://semanticweb.org/

[24] Uschold, M.: Where are the Semantics in the Semantic Web? In *AI Magazine*, The Boeing Company, 2000, [Online; navštíveno 14.4.2010].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.8164\&rep=rep1\&type=pdf

[25] Semantic Web. W3C, 2010, [Online; navštíveno 14. 4. 2010].
URL http://www.w3.org/standards/semanticweb/

[26] Semantic Web. In *Wikipedia – The Free Encyclopedia*, nadace WIKIMEDIA, 2010, [Online; navštíveno 14.4.2010].
URL http://en.wikipedia.org/wiki/Semantic_Web

[27] XSLT. In *Wikipedie – Otevřená encyklopedie*, nadace WIKIMEDIA, 2010, [Online; navštíveno 14.4.2010].
URL http://cs.wikipedia.org/wiki/XSLT

[28] Čerba, O.: Ontologie. 2010, [Online; navštíveno 18. 5. 2010].
URL http://www.slideshare.net/Buthyl/ontologie

[29] Černoš, O.: Sémantický web – principy, prostředky, aplikace. FEL ČVUT, 2009, [Online; navštíveno 14. 4. 2010].
URL http://uisk.ff.cuni.cz/dwn/1003/10522cs_CZ_semanticky%20web.pdf