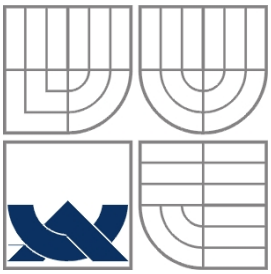
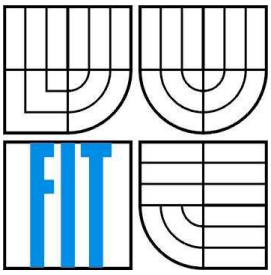


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ONTOLOGIE A SÉMANTICKÝ WEB

ONTOLOGY AND SEMANTIC WEB

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE — Radek Stuchlík
AUTHOR

VEDOUCÍ PRÁCE — Ing. Pavel Očenášek
SUPERVISOR

BRNO 2007

Zadání diplomové práce/5004/2006/xstuch01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Stuchlík Radek**
Obor: Výpočetní technika a informatika
Téma: **Ontologie a Semantický Web**
Kategorie: Web

Pokyny:

1. Seznamte se s možnostmi vývoje SW aplikací v prostředí současného WWW. Nastudujte principy ontologie a tvorby semantického webu.
2. Navrhněte tutorial pro návrh a realizaci jednoduchých aplikací v prostředí sémantického webu. První část tutorialu se bude zabývat problematikou ontologie. Návrh koncipujte tak, aby bylo možné tutorial použít pro výuku ontologie a principů semantického webu v některém z předmětů na FIT VUT.
3. Základní verzi tutorialu implementujte jako interaktivní prezentaci dle pokynů vedoucího práce (do prosince 2006).
4. Doplňte do tutorialu další podrobnější kapitoly týkající se sémantického webu a jeho implementace. Dále tutorial doplňte o demonstrační příklady, detailnější komentáře a odkazy na externí zdroje a literaturu.
5. Diskutujte další možnosti pokračování projektu.

Literatura:

- Dle doporučení vedoucího práce.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Bez požadavků.

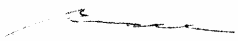
Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Očenášek Pavel, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 22. května 2007

L.S.



doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Radek Stuchlík**
Id studenta: 21173
Bytem: Velkopavlovická 14, 628 00 Brno
Narozen: 14. 11. 1981, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Ontologie a Semantický Web
Vedoucí/školitel VŠKP: Očenášek Pavel, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Diplomová práce s názvem Ontologie a Sémantický Web se snaží o vysvětlení základních principů ontologií, které úzce souvisejí s tzv. novou generací webu: sémantickým webem. Diplomová práce je koncipována jako tutoriál a zaměřuje se tedy jak na teoretické základy, tak i na praktické příklady využití dosud vyvinutých technologií. Účelem tutoriálu je představit základní myšlenky sémantického webu, technologií a datových formátů, které by měly umožnit jeho zavedení do praxe.

Klíčová slova

Sémantický web, sémantika, ontologie, metadata, RDF, OWL, URI.

Abstract

The purpose of the master’s thesis "Ontology and Semantic Web" is to give a description of general principles of ontologies, which are closely associated with so-called new generation of web: semantic web. The thesis is conceived as tutorial and it is focused on both theoretical basics and practical examples of the use of technologies developed recently. The aim of this tutorial is to present main ideas of semantic web, technologies and data formats, which should provide its implementation into standard practice.

Keywords

Semantic web, semantics, ontology, metadata, RDF, OWL, URI.

Citace

Radek Stuchlík: Ontologie a Sémantický Web, diplomová práce, Brno, FIT VUT v Brně, 2007

Ontologie a sémantický web

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radek Stuchlík
V Brně, dne 10.5.2007

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce za podnětné připomínky a nápady pro moji práci.

Obsah

1 Úvod

Tato diplomová práce se zabývá dvěmi oblastmi, které úzce souvisejí se současnou celosvětovou počítačovou sítí Internet. První z nich je oblast, která bývá někdy nazývána ontologickým inženýrstvím a týká se právě ontologií. Oblast ontologického inženýrství je známa již delší dobu, počátky jejího vývoje sahají do 90. let minulého století. Ontologie a ontologické inženýrství poskytlo základ technologie potřebné pro zrealizování myšlenky webu nové generace, jejímž autorem je Tim Barners-Lee, který je ředitelem konsorcia W3C a je také považován za zakladatele současného Internetu. Právě web nové generace, tedy sémantický web, je předmětem druhé části této diplomové práce. Pojem sémantický web spatřil světlo světa již před několika lety, ale ještě stále se nepodařilo uvést do běžné praxe všechny jeho výhody, které byly prezentovány vizionáři této oblasti Internetu. Avšak po dobu vývoje sémantického webu vznikla řada nových technologií a některé z nich jsou již prakticky využívány.

Struktura tohoto dokumentu je poněkud odlišná od standardu jiných diplomových prací. Hlavním důvodem je fakt, že účelem diplomové práce bylo vytvoření tutoriálu, který má být využit jako učební text pro výuku předmětu PIS – Pokročilé Informační Systémy, vyučovaného na fakultě informačních technologií VUT v Brně. Vzhledem k tomu, že formát samotného tutoriálu je odlišný od předepsaného formátu diplomové práce, tato obsahuje pouze shrnutí informací obsažených v tutoriálu. Samotný učební text je potom přiložen jako jedna z příloh diplomové práce, avšak objemem textu odpovídá požadavkům na objem textu diplomové práce.

První část diplomové práce s názvem Tutoriál se tedy věnuje jednotlivým kapitolám přiloženého učebního textu. V druhé části je představena interaktivní verze tutoriálu, která je jakousi alternativou k textovému dokumentu. Tuto verzi tutoriálu je možné spustit v běžném internetovém prohlížeči a to jak lokálně, tak i z webového umístění.

2 Tutoriál

Jak již bylo naznačeno v úvodu, tato kapitola se zabývá jednotlivými částmi učebního textu. Každá následující podkapitola představuje jednu hlavní kapitolu tutoriálu a popisuje hlavní myšlenky a informace o daném tématu, řešeném v rámci konkrétní části tutoriálu.

2.1 Koncepce sémantického webu

Část tutoriálu věnovaná koncepci sémantického webu se snaží přinést odpověď na otázky: Co je to sémantický web? Proč vznikl? Jak má být implementován a pomocí jakých technologií? Dále jsou zde uvedeny základní stavební kameny sémantického webu. Je zde podrobněji popsán problém sémantiky informace. Jakou vlastně známe sémantiku? Vysvětluje se zde, jaký je rozdíl mezi sémantikou implicitní a explicitní, formální a neformální.

Mezi další základní stavební kameny sémantického webu patří metadata. Jaká je definice pojmu metadata? Mimo jiné se zde uvádí, jak lze metadata kategorizovat a je zde také uvedena první zmínka o ontologiích. Následuje objasnění pojmu zdroj. Co vše lze považovat za zdroj a jak jej lze identifikovat? V této části kapitoly je také vysvětlen rozdíl mezi URI (Uniform Resource Identifikator), URL (Uniform Resource Locator) a URN (Uniform Resource Name).

Poslední část kapitoly se zabývá jmennými prostory. Vysvětluje princip deklarace nového jmenného prostoru a také obsahuje tabulku s popisem jmenných prostorů, které budou využity i v dalších částech tutoriálu.

2.2 RDF

Kapitola tutoriálu s názvem RDF (Resource Description Framework), což lze volně přeložit jako: „formát pro popis zdrojů“, představuje nový formát pro reprezentaci webových dat. Přesněji, podle definice konsorcia W3C, které stojí za vývojem tohoto formátu, jde o obecný rámec pro popis, výměnu a znovupoužití metadat.

První část se věnuje syntaxi formátu RDF. Vysvětluje, že RDF nemá definovanou implicitní syntaxi pro reprezentaci dat. Je definována pouze jakási abstraktní syntaxe, která se vztahuje k RDF trojicím. Avšak existují konkrétní syntaxe, které lze pro reprezentaci dat v RDF použít. Jedná se o syntaxe N-Triples, N3 (Notation 3) a pravděpodobně nejčastěji využívanou RDF/XML. Ke každému typu syntaxe je doplněn příklad použití.

Další část se věnuje RDF grafům. RDF graf je definován jako konečná množina RDF trojic. Popisuje také datové typy, které lze v RDF grafech použít, vysvětluje význam pojmu literál a také

objasňuje specifickou část RDF grafů: prázdné uzly. Textový popis je doplněn o spustitelné prezentace ve formátu pps (PowerPoint prezentace), které se snaží přehledně vysvětlit způsob vytvoření RDF grafu a také objasnit význam prázdných uzlů v RDF grafech.

2.3 Ontologie

Tato část tutoriálu se věnuje ontologiím. Snaží se o vymezení pojmu v mezích použitelnosti v prostředí webu a jeho služeb. Jsou zde také uvedeny dvě základní definice od T.Grubera (1993) a W.Borsta (1997), které byly dále rozvíjeny a v současné době existují snad i desítky různě se doplňujících definicí tohoto pojmu.

Dále se kapitola zabývá účelem ontologií a definuje typy ontologií podle různých členění, například dle oborových oblastí na ontologie terminologické, informační a znalostní. Nebo členění podle předmětu formalizace na ontologie generické, doménové, úlohové a aplikační. V poslední části této kapitoly je rozebrána struktura ontologií a jsou zde vysvětleny významy jejích jednotlivých částí, což jsou třídy, instance a individua, relace, omezení slotů, primitivní datové hodnoty a axiomy.

2.4 Ontologické jazyky

Kapitola týkající se ontologických jazyků v úvodu představuje některé významné, dnes již historické, ontologické jazyky. Snaží se objasnit, proč vlastně ontologické jazyky vznikaly a jaký byl postup při jejich vývoji. Častým jevem bylo, že po vzniku jazyka byl jeho vývoj záhy ukončen, ale dosavadní zkušenosti byly použity při vývoji nového jazyka, založeného na principech jazyka předchozího. Podrobněji jsou zde popsány dva významné historické jazyky: SHOE (Simple HTML Ontology Extension), který vznikl v roce 1996 na University of Maryland a jazyk Ontobroker, jež vznikal ve stejné době, ale na evropské univerzitě v Karlsruhe.

Další část této kapitoly se zabývá RDF Schématem (zkráceně RDFS). V tomto případě se nejedná o klasický ontologický jazyk, jde spíše o sémantické rozšíření formátu RDF. Avšak RDFS je významnou součástí dalších ontologických jazyků, zejména dnes nejpoužívanějšího ontologického jazyka OWL, kterému je vyhrazena samostatná kapitola tutoriálu. RDFS totiž doplňuje do struktury RDF základní ontologické konstrukce, třídy a binární relace a umožňuje nad těmito třídami a relacemi definovat hierarchické struktury. Ty pak lze využívat v ostatních ontologických jazycích, včetně zmiňovaného OWL.

Následující část kapitoly se věnuje deskripční logice. Deskripční logika je vlastně rodinou logických formalismů, které se využívají pro reprezentaci znalostí. Jsou zde popsány základy syntaxe a sémantiky deskripční logiky a také konstruktory logických výrazů. Informace o deskripční logice

jsou zde uvedeny převážně kvůli návaznosti na následující popisovaný ontologický jazyk DAML+OIL, jehož základy jsou vystavěny právě na deskripční logice.

Poslední část této kapitoly se tedy věnuje jazyku DAML+OIL, který vznikl spojením dvou rozdílných koncepcí: DAML (DARPA Agent Marku-up Language) a OIL (Ontology Inference Layer).

2.5 OWL

Pro ontologický jazyk OWL (Ontology Web Language) byla vyhrazena samostatná kapitola, protože v současné době je tento jazyk nejpoužívanějším ontologickým jazykem. Za jeho vznikem a vývojem stojí konsorcium W3C. Velký význam na vzniku OWL mají předchozí zkušenosti s jazykem DAML+OIL.

První část kapitoly o jazyce OWL představuje jeho další dvě verze OWL Light a OWL DL, které by měly rozlišit úrovně použitelnosti jazyka ve vztahu k různé úrovni znalostí uživatele (vývojáře) a také ve vztahu k potřebám jednotlivých aplikací. Nejvyšší verzí je potom samotný jazyk OWL, někdy označovaný jako OWL Full.

Zbylá část kapitoly podrobně rozebírá strukturu jazyka OWL. Zaměřuje se na reprezentaci ontologických tříd, pomocí příkladů vysvětluje možné způsoby deklarací tříd a jejich omezení. Jsou zde uvedeny příklady tříd definovaných identifikátorem, výčtem prvků nebo omezením vlastností tříd (sjednocením, průnikem, doplňkem). Stejným způsobem popisuje vlastnosti tvrzení. Vysvětluje rozdíly mezi vlastnostmi objektovými a dato-typovými. Obsahuje výčet konstruktorů, které lze použít k definici vlastností a opět, pomocí příkladů, názorně popisuje jejich použití.

2.6 Editor ontologií Protégé-owl

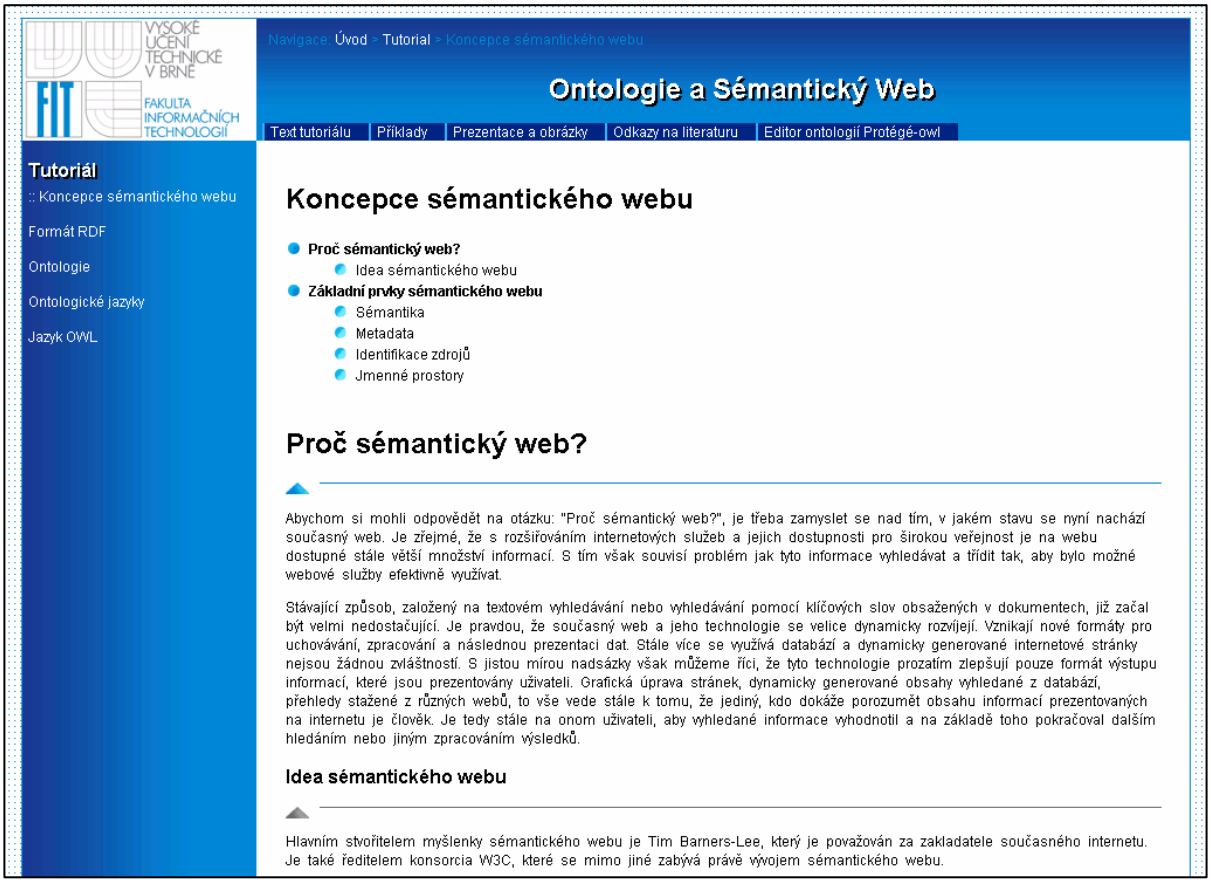
Poslední kapitola tutoriálu je věnována praktickým nástrojům pro vytváření a editaci ontologií, tedy konkrétně jednomu vybranému nástroji Protégé-owl. Jedná se o freeware software, který je možné legálně stáhnout ze sítě Internet a volně používat. Důležitým argumentem pro výběr tohoto nástroje je fakt, že jako jeden z mála volně dostupných editorů je stále vyvíjen a nabízí editační funkce, které jsou běžně dostupné u komerčních editorů.

Kapitola o editoru Protégé-owl obsahuje odkazy, odkud je možné editor stáhnout a popisuje postup instalace. Dále čtenáře stručně seznamuje s uživatelským rozhraním, představuje jeho možnosti a také obsahuje postup při vytváření nové ontologie.

3 Webová verze tutoriálu

Jako doplněk k textové verzi tutoriálu byla v rámci diplomové práce vytvořena webová verze tutoriálu. Hlavní nabídka obsahuje odkazy, které mají čtenáři nabídnout efektivnější výběr kapitol tutoriálu, možnost zobrazit si pouze příklady seřazené podle příslušnosti ke kapitolám tutoriálu. Mimo to nabídka obsahuje seznam všech obrázků a spustitelných prezentací ve formátu pps (PowerPoint). Další položkou nabídky jsou odkazy na literaturu použitou pro vypracování tutoriálu. Odkazované dokumenty mohou čtenáři nabídnout podrobnější informace o daném tématu zpracovaném v tutoriálu.

Webová verze tutoriálu byla implementována pouze pomocí XHTML, CSS a několika Java skriptů. Důvodem k této formě implementace byl požadavek na možnost spuštění tutoriálu jak z webového, tak i z lokálního umístění. Tato verze tutoriálu je k diplomové práci přiložena na CD. Následující obrázek obsahuje pohled na uživatelské rozhraní webového tutoriálu:



4 Závěr

Oblast ontologií a sémantického webu představuje hlavní směr vývoje současného Internetu a jeho služeb. V této diplomové práci, resp. v přiloženém tutoriálu, jsem se snažil zachytit vznik a vývoj technologií potřebných pro realizaci principů a myšlenek sémantického webu a také popsat současnou úroveň poznání těchto technologií vzhledem k možnostem jejich využití pro potřeby sémantického webu. Avšak vzhledem k rychlosti, s jakou se objevují nové poznatky a přibývá vývojářů, kteří experimentují s uvedenými technologiemi, je dost dobře možné, že některé informace v tutoriálu již budou zastaralé a některé služby, které jsou zde uváděny jako vizionářské, již budou prakticky použitelné.

I přes tento fakt se domnívám, že informace obsažené v tutoriálu mohou být i nadále považovány za solidní zdroj základních informací o hlavních stavebních kamenech sémantického webu a ontologického inženýrství. Nepřehlédnutelným přínosem tutoriálu je, že je vypracován kompletně v češtině. Z vlastních zkušeností, kterých jsem nabyl při tvorbě tohoto tutoriálu, vím, že většina relevantních materiálů týkajících se sémantického webu je dostupná převážně v angličtině.

Možnosti dalšího pokračování tohoto, poněkud specifického, projektu jsou vzhledem k předchozím slovům zřejmé. Je třeba dále sledovat vývoj technologií sémantického webu a snažit se zmapovat případné změny týkající se stávajících technologií nebo vzniku nových, například ontologických jazyků.

Seznam příloh

Ontologie a sémantický web

tutoriál

Obsah

1 Úvod

V tomto dokumentu jsem se snažil shrnout některé základní informace o nově vznikající a neustále se vyvíjející oblasti informačních technologií a sice sémantického webu a s ní přímo související problematikou ontologií. Dokument je psán formou tutoriálu, snaží se tedy popsat jak teoretický základ dané problematiky, tak také představit a vysvětlit praktickou část problému pomocí příkladů. První část tutoriálu se zabývá základními principy a součástmi sémantického webu. Druhá část tutoriálu vysvětluje problematiku ontologií a jsou zde podrobněji popsány některé ontologické jazyky.

1.1 Struktura dokumentu

Struktura dokumentu

Dokument je strukturován pomocí číslovaných kapitol, které jej rozdělují na několik logických celků. Formát dokumentu tutoriál rozděluje na textovou část a levý pomocný sloupec, ve kterém jsou umístěna pomocná hesla pro snazší orientaci v textu. Za účelem vyhledávání je na konci dokumentu umístěn rejstřík.

Tutoriál obsahuje také spustitelné prezentace ve formátu MS Office PowerPoint, které jsou v levém sloupci označeny ikonou:

1.2 Předpokládané znalosti

Předpokládané znalosti

Vzhledem k omezenému rozsahu problematiky popisované v tomto tutoriálu se u čtenáře předpokládají některé základní znalosti z oblasti internetových technologií a aplikací, které zde nejsou vysvětleny.

Konkrétně se jedná o problematiku značkovacích jazyků. Naprostou nutností je znalost formátu HTML (HyperText Markup Language). Aby bylo možné pochopit novější datové formáty, které jsou vhodné pro sémantický web, je třeba znát alespoň základní principy formátu XML (eXtended Markup Language) a způsoby zpracování XML dokumentů.

2 Koncepce sémantického webu

2.1 Proč sémantický web?

Proč sémantický web?

Abychom si mohli odpovědět na otázku: „Proč sémantický web?“, je třeba zamyslet se nad tím, v jakém stavu se nyní nachází současný web. Je zřejmé, že s rozšiřováním internetových služeb a jejich dostupnosti pro širokou veřejnost je na webu dostupné stále větší množství informací. S tím však souvisí problém, jak tyto informace vyhledávat a třídit tak, aby bylo možné webové služby efektivně využívat.

Stávající způsob, založený na textovém vyhledávání nebo vyhledávání pomocí klíčových slov obsažených v dokumentech, již začal být velmi nedostačující. Je pravdou, že současný web a jeho technologie se velice dynamicky rozvíjejí. Vznikají nové formáty pro uchovávání, zpracování a následnou prezentaci dat. Stále více se využívá databází a dynamicky generované internetové stránky nejsou žádnou zvláštností. S jistou mírou nadsázky však můžeme říci, že tyto technologie prozatím zlepšují pouze formát výstupu informací, které jsou prezentovány uživateli. Grafická úprava stránek, dynamicky generované obsahy vyhledané z databází, přehledy stažené z různých webů, to vše vede stále k tomu, že jediný, kdo dokáže porozumět obsahu informací prezentovaných na Internetu je člověk. Je tedy stále na onom uživateli, aby vyhledané informace vyhodnotil a na základě toho pokračoval dalším hledáním nebo jiným zpracováním výsledků.

2.1.1 Idea sémantického webu

Idea sémantického webu

Hlavním stvořitelem myšlenky sémantického webu je Tim Barners-Lee, který je považován za zakladatele současného Internetu. Je také ředitelem konsorcia W3C, které se mimo jiné zabývá právě vývojem sémantického webu.

Na rozdíl od současného webu, který je zaměřen na prezentaci informací srozumitelných pro člověka, se sémantický web snaží prezentovat informace takovým způsobem, aby byly srozumitelné jak pro člověka, tak pro stroje. Pod pojmem „informace srozumitelné pro stroje“ je myšlena schopnost speciálních softwarových aplikací vyhledat a zpracovat obsah informace získané z webu. Přeneseně řečeno, takové aplikace by měly “vědět“, co mají s nalezeným obsahem dělat. Na řešení takového problému bychom se mohli podívat ze dvou obecných hledisek.

Porozumění specifikaci sémantiky informace

Prvním pohledem je doplnění informací prezentovaných na webu o sémantické informace, kterým by byly speciální softwarové aplikace schopny porozumět. To by znamenalo vytvoření obecně platné specifikace sémantiky všech informací, které by pak byly strojově čitelné. Taková aplikace by dokázala číst a automaticky zpracovat obsah webu. Díky tomu by bylo možné zpracovávat

mnohem složitější webové úlohy, například kombinované vyhledávání informací s jejich vyhodnocením. Současná úroveň poznání webových technologií však prozatím podobné řešení problému neumožňuje.

Přidání sémantiky do aplikací

Druhou možností zanesení sémantiky do webu je přímé zapracování znalostí do webových aplikací, které pak dokáží vyhledávat informace klasickými metodami a na základě těchto znalostí informace zpracovat. Tento způsob představuje jakousi alternativní možnost k prozatím vizionářské představě sémanticky obohacených informací.

Aplikací, které pracují na takovém principu již existuje spousta. Například webové aplikace, někdy označované jako "obchodní agenti", které dokáží vyhledávat informace z webů na základě některých klíčových slov a najít tak nejlevnější cenu požadovaného výrobku apod.

2.2 Základní prvky sémantického webu

Základní prvky sémantického webu

Abychom mohli lépe pochopit princip sémantického webu, je třeba vysvětlit si některé základní pojmy a popsat základní prvky budoucí generace webu.

2.2.1 Sémantika

Jak již sám název napovídá, nejpodstatnější součástí sémantického webu je právě ona sémantika. Vzhledem k zaměření a rozsahu tohoto dokumentu se zaměříme na objasnění tohoto pojmu pouze v souvislosti se sémantickým webem.

Základním a nejjednodušším výkladem slova sémantika je "význam" (v některých publikacích je sémantický web nazýván významovým webem). Jak již bylo řečeno, významnou součástí sémantického webu budou programy, agenti, kteří budou schopni vykonávat složité úlohy právě na základě porozumění obsahu různých webových zdrojů, budou chápat jejich "význam". Stejně tak bude třeba, aby byly schopny vyměňovat si informace mezi sebou. Jaký druh sémantiky bude tedy třeba, v čem bude obsažena a jak bude možné ji využívat? Obecně bychom mohli náhled na sémantiku ve vztahu k sémantickému webu rozdělit do čtyř skupin:

- Implicitní
- Explicitní (neformální)
- Formální pro zpracování člověkem
- Formální pro strojové zpracování

Implicitní sémantika

Implicitní sémantika

V nejjednodušším případě se můžeme na sémantiku dívat jako na implicitní. V takovém případě je význam založen na společném porozumění lidí danému termínu podle konvenčních znalostí.

Předpokládejme, že existuje webová aplikace na bázi internetového obchodu a je implementována pomocí XML. Je běžnou praxí, že jednotlivé

ceny zboží budou označeny značkami s názvem “cena“. V současné době neexistuje způsob, jak pomocí formátu XML definovat význam značky, ale člověk tomuto termínu porozumí a může tak tuto “znalost“ zapracovat do webové aplikace. Na stejném principu fungují v současné době již zmiňovaní obchodní agenti.

Je zřejmé, že nevýhodou implicitní sémantiky je právě nejednoznačnost významu. Nejsou určena žádná pravidla pro zachycení sémantiky a tak i zdánlivě jednoznačné termíny, jako například zmiňovaná “cena“, mohou mít v různých implementacích různé významy (různá měna, obsažení daně apod.).

Explicitní (neformální) sémantika

Explicitní sémantika

Dalším případem neformální sémantiky je explicitní sémantika. Na rozdíl od implicitní sémantiky je význam explicitně vyjádřen, ale problém zůstává v neformálnosti tohoto vyjádření.

Představme si například slovník cizích slov. Pro všechna hesla ve slovníku je zde neformálně (přirozeným jazykem) explicitně vyjádřen význam, ale vzhledem ke složitosti přirozeného jazyka je v současné době téměř nemožné zpracovat tyto sémantické informace strojově. Je zřejmé, že explicitní sémantika je určena převážně pro člověka.

Formální sémantika pro zpracování člověkem

Formální sémantika

Podobně jako u předchozího typu sémantiky je formální sémantika explicitní. Avšak, jak už plyne z názvu, jde o sémantiku určenou pro zpracování člověkem. Můžeme si ji představit jako dokumentaci psanou formálním jazykem nebo jako formální specifikaci významu.

Formální sémantika pro strojové zpracování

Formálně a explicitně vyjádřená sémantika tak, aby byla plně automaticky zpracovatelná, je prozatím vizí. Předpokladem je, že aplikace (softwarový agent) bude schopna automaticky zpracovat i zcela nové pojmy, jejichž sémantiku by dokázala odvodit na základě jiných znalostí. Zde však nastává problém, jak toho lze dosáhnout.

Pomineme-li výše zmiňovaný způsob zapracování znalostí přímo do aplikací, je další možností pouze formální deklarace sémantiky, která by umožňovala dynamicky odhalovat význam obsahu a možnosti jeho zpracování. To však nemůže fungovat zcela obecně, ale je třeba znát syntaktická pravidla daného jazyka, rozumět jeho symbolům. Existuje několik předpokladů, které je třeba brát v úvahu pro úspěšné zavedení formální sémantiky do praxe:

- Stejný jazyk reprezentace
  - různé ontologické jazyky mají různou vyjadřovací sílu, různou míru formální podpory pro zachycení sémantiky
- Logicky kompatibilní konceptualizace
  - stejný jazyk nezaručuje “dorozumění“ dvou stran

- Veřejně deklarované koncepty
  - ani při dodržení předpokladů stejného jazyka a kompatibilní konceptualizace nelze zaručit, že dva lidé budou používat pro stejnou doménu stejnou ontologii
  - dva různé pojmy mohou mít stejný význam a zároveň stejný pojem může mít dva různé významy

2.2.2 Metadata

Metadata

Pojem metadata začal být předmětem zájmu odborné veřejnosti již v 90. letech minulého století. Vznik a rozvoj digitálních knihoven zapříčinil, že bylo třeba nalézt způsob jak efektivněji popisovat datové záznamy. Absolutní rozvoj této tématiky však zaznamenal až vznik a vývoj Internetu a jeho služeb.

Samotné slovo metadata vychází z řeckého slova meta, které lze přeložit jako "mezi" a latinského slova data, tedy "to co je dáno". Velmi často bývá obecně interpretováno jako data o datech, nebo také informace o informacích. V současné době je však tento popis naprosto nedostačující a nevyjadřuje podstatu významu metadat ve vztahu k sémantickému webu.

Definice

Jedním z prvních tvrzení, které může být považováno za definici metadat současné doby je:

Definice pojmu metadata

„Metadata jsou stroji srozumitelné informace o webových zdrojích nebo dalších věcech"

‚jejíž autorem je Tim Barners-Lee. Podle něj existuje na metadata několik pohledů. Hlavním předpokladem je, že metadata jsou stále data, z čehož vyplývá, že mohou být součástí nějakého zdroje (mohou v něm být uloženy). Mohou tedy obsahovat informace o sobě samém nebo také o dalších zdrojích. Tyto informace mohou být různého charakteru. Mohou vypovídat o obsahu dat, o způsobu uložení dat a jejich spravování. Na druhou stranu mohou obsahovat informace, které přímo nesouvisí s obsahem dat, ale například datum vytvoření dokumentu, umístění apod.

Použití metadat se sebou mimo jiné přináší nespornou výhodu, kterou je redukce informačního zahlcení a to převážně díky dvěma výhodám:

- Umožňují abstrahovat důležité informace (formát nebo organizace dat) a zachycují informační obsah nezávisle na původní formě dat.
- Umožňují reprezentovat doménové znalosti popisem informační oblasti, k níž přísluší výchozí data (je tedy možné ověřit si relevanci dat bez nutnosti přistupovat k datům samotným).

Kategorie metadat

Podle úrovně abstrakce, s níž metadata popisují obsah, lze metadata rozdělit od několika kategorií:

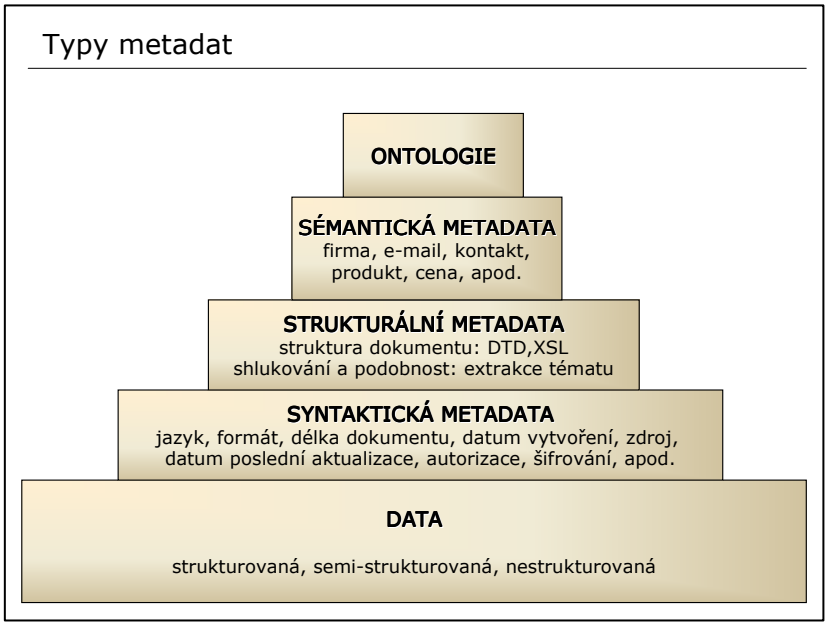
- **Syntaktická metadata** obsahují podrobnosti o zdroji dat (dokumentu). Tato skupina metadat slouží převážně ke katalogizaci nebo kategorizaci.

- **Strukturální metadata** se zaměřují na strukturu dokumentu. Tyto informace lze využít při ukládání, zpracování nebo prezentaci dokumentů. Zjednodušují vyhledávání.

- **Sémantická metadata** popisují kontextově relevantní informace vzhledem k určité doméně. Užitím sémantických metadat lze získat smysluplnou interpretaci dat při zachování možnosti efektivní spolupráce systému na vyšší úrovni.

- **Ontologie** představují nejvyšší formu metadat a současně jsou klíčovým principem sémantického webu. Více viz samostatná kapitola.

Typy metadat



2.2.3 Identifikace zdrojů

Identifikace zdrojů

Současný web je založen na zdrojích a odkazech na ně. Za zdroj v tomto případě obvykle považujeme například dokument, obrázek apod. Tyto odkazy jsou v současném webu zpravidla tvořeny pomocí takzvaných URL (Uniform Resource Locator).

URL

URL

URL můžeme definovat jako řetězec znaků se stanovenou syntaxí [Barners-Lee94], který slouží ke specifikaci umístění požadovaného zdroje na Internetu. URL definuje doménovou adresu serveru, umístění zdroje na serveru a protokol, pomocí kterého je možné zdroj zpřístupnit (viz následující příklad).

Příklad 2.1: URL

```
URL:
http://www.server.cz/dokumenty/doc1.html

protokol:   http
server:     www.server.cz
zdroj:      /dokumenty/doc1.html
```

Příklad 2.1 URL

Přeneseně řečeno, pomocí URL lze identifikovat pouze zdroje, které jsou umístěny na Internetu. Avšak idea sémantického webu se neomezuje pouze na takovéto zdroje, ale požaduje, aby bylo možné identifikovat jakékoliv zdroje z reálného světa (cokoliv abstraktního i skutečného) a k tomuto účelu slouží URI.

URN

URN

Dalším způsobem jak identifikovat zdroj je pomocí URN (Uniform Resource Name), tedy jedinečného názvu zdroje. Proti URL, které slouží k lokalizaci zdroje na Internetu, URN označuje název zdroje, pomocí něhož lze zdroj identifikovat nezávisle na umístění zdroje. Přeneseně řečeno, URN nám říká "co" hledáme a URL "kde" to nalezneme.

Na URN jsou kladeny určité existenční podmínky, jejichž splnění zajišťuje funkčnost URN. Mezi nejdůležitější patří:

- Zachování významu názvu v globálním měřítku – název nesmí představovat konkrétní umístění
- Jedinečnost – stejné URN nesmí být přiřazeno dvěma různým zdrojům
- Stálá životnost – předpokládá se, že význam URN bude trvalý a navždy jedinečný. Je také třeba brát ohled na životnost zdroje.

Syntaxe URN je následující:

```
urn: <NID> : <NSS>
```

Kde <*NID*> představuje *Namespace Identificator*, tedy identifikátor jmenného prostoru a <*NSS*> představuje *Namespace Specific String*, tedy specifický řetězec jmenného prostoru. Některé identifikátory jmenných prostorů jsou již rezervované a jsou často využívány. Například *isbn*, který souvisí se systémem ISBN (International Standard Book Number) pro jedinečnou identifikaci knih pomocí čísel. Následující řádek ukazuje, jak by vypadalo URN například pro knihu The Godfather od Maria Puza:

```
urn:isbn:0399103422
```

URI

URI

Zkratka URI (Uniform Resource Identificator) označuje jedinečný identifikátor zdroje. URI je také řetězec znaků, který odpovídá určité syntaxi [Barners-Lee05], ale v identifikaci zdroje se neomezuje pouze na jeho umístění nebo název. URL a URN jsou vlastně určitým typem URI nebo lépe řečeno, URL a URN tvoří jakousi podmnožinu všech URI. Společnou vlastností všech URI je, že mohou být vytvořeny různými společnostmi naprosto nezávisle a mohou být používány k identifikaci zdrojů dosažitelných pomocí Internetu, ale stejně tak k identifikaci zdrojů, které prostřednictvím Internetu dosažitelné nejsou. Za identifikovatelný zdroj můžeme považovat konkrétní osobu, společnost, ale i abstraktní pojmy, které nemají fyzickou podobu.

URI reference

URI reference

S pojmem URI úzce souvisí pojem URI reference, který označuje běžné použití identifikátoru zdroje. Je to další typ řetězce, kterým lze reprezentovat URI, nebo lépe řečeno, reprezentovat zdroj identifikovaný pomocí URI. Při běžném neformálním použití se rozdíl mezi URI a URI referencí příliš neprojevuje. Je však třeba říct, že URI reference mohou představovat jak plnou URI, v takovém případě hovoříme o absolutní URI referenci, tak i jen její část specifickou pro dané schéma, tedy jde o relativní URI referenci.

Absolutní URI reference identifikuje zdroj bez ohledu na kontext, ve kterém se URI reference nachází. Naproti tomu relativní URI reference je zkrácenou formou absolutní URI reference, kde část (prefix) může chybět. URI reference může navíc k absolutní části URI obsahovat takzvaný identifikátor fragmentu, který se od hlavní části URI odděluje znakem #. Tento identifikátor fragmentu má význam pro zpřesnění identifikace zdroje, nebo některé jeho části. Následující příklad názorně ukazuje rozdíly mezi absolutními a relativními URI referencemi.

Příklad 2.2: URI reference

```
Absolutní URI reference:
http://www.server.cz/dokumenty/doc1.html#odstavec1
http://www.server.cz/dokumenty/doc1.html

Relativní URI reference:
/dokumenty/doc1.html
#odstavec1
```

Příklad 2.2 URI a URI reference

2.2.4 Jmenné prostory

Jmenné prostory

Dále je třeba vysvětlit pojem, který úzce souvisí s datovými formáty na bázi XML. Vzhledem k tomu, že tyto formáty dovolují vytvářet si vlastní značky, bylo třeba nějakým způsobem zabezpečit jedinečnost významu jednotlivých jmen značek a právě k tomuto účelu slouží jmenné prostory. Jmenné prostory jsou jakési množiny jmen, které jsou si nějak významově blízké. Každý jmenný prostor je identifikován vlastním URI. Vzhledem k faktu, že některá

URI mohou být velmi dlouhá, používá se takzvaných prefixů, které jsou danému URI přiřazeny (deklarace jmenného prostoru). V dokumentu potom stačí před názvem značky uvést daný prefix jmenného prostoru, do kterého název značky spadá. K deklaraci jmenného prostoru slouží atribut značky *xmlns*. Syntaxe použití atributu je následující:

```
xmlns:označení_prefixu_jmenného_prostoru = "URI
obsahující jmenný prostor"
```

Existují jmenné prostory, jejichž prefixy jsou již rezervovány a nelze je deklarovat. Vlastně i samotný atribut *xmlns* je sám prefixem jmenného prostoru, který je identifikován pomocí URI "http://www.w3.org/2000/xmlns" a slouží k definici prefixů jmenných prostorů. Následující tabulka zobrazuje některé další předdefinované prefixy jmenných prostorů a jejich význam.

Prefix	**URI**	**Význam**
xml	"http://www.w3.org /XML/1998/namespace"	jmenný prostor pro značky, používané k obecnému popisu XML dokumentu nebo jeho částí
rdf	"http://www.w3.org /1999/02/22-rdf-syntax-ns#"	jmenný prostor pro značky formátu RDF
rdfs	"http://www.w3.org /2000/01/rdf-schema#"	jmenný prostor pro značky formátu RDFS

Tabulka 2.1 Rezervované prefixy jmenných prostorů

Deklarovaný prefix a s ním spojený jmenný prostor lze použít již ve značce, která jej deklaruje a také ve všech potomcích této značky ve struktuře dokumentu. Následující příklad ukazuje deklaraci a použití smyšleného jmenného prostoru *mjp* (můj jmenný prostor). V příkladu je vidět již zmiňovaný fakt, že prefix jmenného prostoru lze použít již ve značce, ve které je teprve deklarován.

Příklad 2.3: Deklarace jmenného prostoru

```
<mjp:hlavni
xmlns:mjp="http://radek.stuchlik/namespaces/mjp">
    <mjp:vedlejsi>
                ...
 </mjp:vedlejsi>
 ...
</mjp:html>
```

Příklad 2.3 Deklarace jmenného prostoru a jeho použití

3 RDF

Standard RDF (Resource Description Framework), tedy volně přeloženo jako formát pro popis zdrojů. Podle definice konsorcia W3C je to: *„obecný rámec pro popis, výměnu a znovupoužití metadat“*.

Hlavním účelem vzniku formátu RDF, byla potřeba vytvořit obecný jazyk, kterým by bylo možné reprezentovat informace o webových zdrojích, tedy přeneseně řečeno o metadatech webových zdrojů.

3.1 Syntaxe RDF

Syntaxe RDF

V prvé řadě je třeba zdůraznit, že formát RDF nemá přímo definovanou syntaxi jako ostatní logické jazyky. RDF disponuje jakousi abstraktní syntaxí, která předpokládá, že každý webový zdroj má určité vlastnosti, které můžeme nějakým způsobem popsat, a na základě kterých můžeme o zdrojích utvářet jednoduchá tvrzení ve formě takzvaných trojic. Každá taková trojice (tvrzení) se skládá ze subjektu, predikátu a objektu.

- Subjekt představuje věc, kterou chceme pomocí RDF popsat. Může jí být jakýkoliv zdroj, kterému lze přiřadit URI a může tedy být identifikován pomocí URI reference.
- Predikát se někdy označuje jako vlastnost tvrzení a popisuje vztah mezi subjektem a objektem.
- Objekt potom představuje hodnotu dané vlastnosti tvrzení.

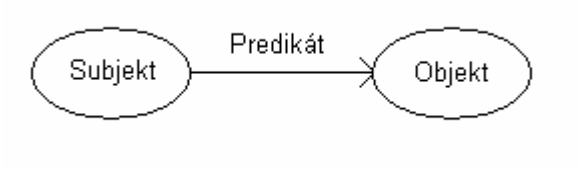
Existuje více možných způsobů, jak lze reprezentovat tuto abstraktní syntaxi RDF. Vzhledem k potřebám sémantického webu je již ve specifikaci formátu RDF doporučená syntaxe pomocí značek XML (více viz kapitola 3.3.3).

3.2 RDF grafy

RDF grafy

Základem RDF jsou tedy uspořádané trojice: <subjekt, predikát, objekt>. Množina takových trojic se nazývá RDF grafem. Jinými slovy RDF graf je definován jako konečná množina RDF trojic. RDF graf se zobrazuje pomocí orientovaného grafu propojených uzlů.

- Každá trojice je reprezentována dvojicí uzlů (nebo dvojicí uzel a list) a orientovanou spojnicí (hranou).
- Hrana je vždy orientována směrem od subjektu k objektu a vyjadřuje vztah mezi subjektem a objektem.
- Objekty tvrzení mohou vystupovat jako subjekty jiného tvrzení nebo mohou být listem grafu.
- Každou trojici RDF můžeme spojit s jinou trojicí, aniž by se změnil její význam, bez ohledu na celkovou složitost grafu.



Obrázek 3.1 RDF trojice

Každá část RDF trojice může být v grafu reprezentována pomocí URI reference. Ale subjekt a objekt, které tvoří uzly grafu, mají ještě jiné možnosti reprezentace.

Subjekt může být reprezentován také *prázdným uzlem*. Objekt může být kromě URI reference reprezentován prázdným uzlem, nebo řetězcem *literálů*.

3.2.1 Datové typy

Datové typy

RDF používá datové typy pro reprezentaci hodnot, například celých čísel, čísel s plovoucí desetinnou čárkou a dat. Datové typy se skládají z lexikálního prostoru, prostoru hodnot a lexikálně-hodnotového přiřazení.

Lexikální prostor datového typu je množina řetězců kódovaných pomocí Unicode.

Lexikálně-hodnotové přiřazení datového typu je množina uspořádaných dvojic, kde první část dvojice patří do lexikálního prostoru datového typu a druhá část dvojice patří do prostoru hodnot datového typu. Dále platí:

- Každý prvek lexikálního prostoru může být přiřazen pouze k jednomu prvku z prostoru hodnot.
- Každý prvek z prostoru hodnot může být přiřazen k jakémukoliv počtu prvků (i žádnému) z lexikálního prostoru (lexikální reprezentace hodnoty).

Například v lexikálně-hodnotovém přiřazení datového typu XML Schématu *xsd:boolean* má každý prvek prostoru hodnot dvě lexikální reprezentace:

- Prostor hodnot:
  {T,F}
- Lexikální prostor:
  {“0”,”1”,”true”,”false”}
- Lexikálně-hodnotové přiřazení:
  {(“false”,F),(“0”,F),(“true”,T),(“1“,T)}

RDF nemá definovaný žádný vnitřní koncept pro reprezentaci čísel a dalších běžných hodnot, ale využívá datové typy definované odděleně, které jsou identifikovány pomocí URI reference. Běžně se využívá datových typů předdefinovaných pro XML Schéma. Některé datové typy používané v XML Schématu však nejsou vhodné pro použití v RDF.

Uvnitř RDF neexistuje ani žádný standardní postup pro definování nových datových typů, ale díky XML Schématu je možné rozšiřovat stávající

datové typy. RDF má předdefinovaný pouze jediný datový typ *rdf:XMLLiteral*, který se používá ke vkládání značek XML do RDF.

3.2.2 Literály

Literály

Literály slouží k identifikování hodnot (čísel, dat) prostřednictvím lexikální reprezentace. Cokoli reprezentovatelného pomocí literálu může být reprezentováno pomocí URI reference. Literál může v RDF tvrzení vystupovat pouze jako objekt, nikoli jako subjekt nebo predikát. Rozlišujeme dva typy literálů: jednoduchý literál a typovaný literál.

- Jednoduchý literál je řetězec znaků Unicode, ke kterému může být přidána značka konkrétního jazyka.
- Typovaný literál je také řetězec znaků Unicode, za kterým však musí být povinně uvedena URI reference datového typu. Označuje prvek identifikovaného prostoru hodnot datového typu získaného pomocí lexikálně-hodnotového přiřazení k řetězci literálu.

Typované literály, které mohou být definovány pomocí datového typu *xsd:boolean* XML Schématu jsou uvedeny v následující tabulce:

Typovaný literál	**Lexikálně-hodnotové přiřazení**	**Hodnota**
<xsd:boolean, "0">	<"0", F>	F
<xsd:boolean, "false">	<"false", F>	F
<xsd:boolean, "1">	<"1", T>	T
<xsd:boolean, "true">	<"true", T>	T

Tabulka 3.1

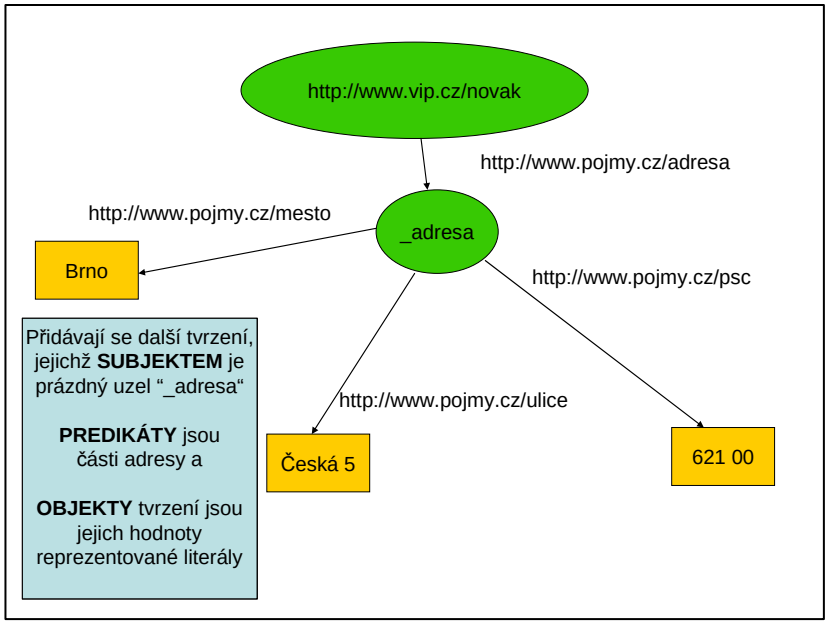
3.2.3 Prázdné uzly

Prázdné uzly

Prázdný uzel je speciální variantou uzlu v RDF grafu. Hlavním rozdílem proti uzlům, které představují objekty a subjekty, je to, že k těmto uzlům není přiřazena žádná URI reference. Vnitřní struktura RDF neupřesňuje, jak mají být prázdné uzly označeny. Mohou být zcela neoznačené, ale obecně se doporučuje nějak tyto uzly v grafu označovat, aby byly identifikovatelné.

Příkladem vhodného použití prázdného uzlu může být RDF graf, zachycující informaci o adrese osoby. Pro zjištění potřebných informací není důležitý pojem “adresa“ jako taková. Je třeba znát ulici, město a poštovní směrovací číslo. Na následujícím obrázku je pomocí RDF grafu reprezentováno tvrzení v přirozeném jazyce: „Pan Novák bydlí v Brně na ulici Česká 10 a jeho PSČ je 62100“.

RDF graf
Prázdný uzel



Obrázek 3.2 RDF graf s prázdným uzlem

3.3 Konkrétní syntaxe RDF

Konkrétní syntaxe RDF

Jak již bylo uvedeno, RDF nemá implicitně definovanou konkrétní syntaxi, ale pouze jakousi abstraktní syntaxi. V současné době existuje několik konkrétních syntaxí používaných k zápisu RDF tvrzení.

3.3.1 N-Triples

N-Triples

Jednou z možností zápisu RDF trojic je syntaxe N-Triples. Jejím základem je řádkový textový formát. Na každém řádku je zapsána jedna trojice ve formátu:

```
<subjekt> <predikát> <objekt> .
```

Pokud je objekt tvořen literálem, zapisuje se tento do uvozovek místo mezi závorky < >. K literálu lze přiřadit také typ, který se od názvu literálu odděluje pomocí znaků ^^. Pokud bychom chtěli zapsat jako objekt číslo 15 a označit, že se jedná o číslo typu integer, vypadalo by to následovně:

```
<subjekt> <predikát> "15"^^xsd:integer .
```

Nevýhodou této syntaxe je obvykle velký objem dat a také menší přehlednost pro uživatele. Každý subjekt a predikát je identifikován pomocí URI reference, což mohou být dlouhé řetězce. Subjekt tvrzení musí být uveden na každém řádku, ve kterém vystupuje, i když je subjektem pro více tvrzení. Výhodou pak může být jednoduché strojové zpracování a generování.

Následující příklad obsahuje RDF trojice z obrázku 3.2 zapsané pomocí syntaxe N-Triples.

Příklad 3.1: Syntaxe N-Triples

```
<http://www.vip.cz/novak>
    <http://www.pojmy.cz/adresa> <_adresa> .
<_adresa> <http://www.pojmy.cz/mesto> “Brno“ .
<_adresa> <http://www.pojmy.cz/ulice> “Česká 5“ .
<_adresa> <http://www.pojmy.cz/psc> “621 00“ .
```

Příklad 3.1 Syntaxe N-Triples

3.3.2 N3 (Notation 3)

N3 (Notation 3)

Další syntaxí, kterou lze použít pro reprezentování RDF trojic je tzv. notace 3. Tato syntaxe je na první pohled podobná výše zmiňované syntaxi N-Triples. Každá trojice je opět zaznamenávána textově ve formátu:

<subjekt> <predikát> <objekt> .

Zde se však mohou používat prefixy jmenných prostorů s relativními URI referencemi (viz kapitola 2.2.3). Prefix lze k URI referenci přiřadit pomocí klíčového slova: *@prefix*. Za klíčové slovo se obecně považuje každé slovo, začínající znakem @ (zavináč). Relativní URI reference potom zapisujeme ve formátu: *prefix*:*URI reference*. Je třeba zdůraznit, že v případě použití zkráceného zápisu části trojice pomocí prefixu se již tyto části neuzavírají do závorek < >.

Prázdné uzly se označují podobně, jako relativní URI reference, ale namísto prefixu se uvádí pouze znak _ (podtržítko). Tedy formát zápisu prázdného uzlu je: _:*název uzlu*.

Dalším důležitým rozdílem N3 syntaxe od N-Triples je možnost zapisovat predikáty a objekty patřící k jednomu subjektu tak, že se pouze jednou uvede subjekt, následuje seznam uspořádaných dvojic: predikát objekt, oddělených středníkem. Podobně je možné zkráceně zapsat tvrzení, ve kterých vystupuje více objektů pro stejný subjekt a predikát. V tomto případě se objekty od sebe oddělují čárkami.

Následující příklad opět zachycuje zápis RDF trojic z obrázku 3.2, tentokrát pomocí syntaxe N3. V první části jsou deklarovány jmenné prostory a jejich prefixy. V druhé části jsou zaznamenány samotné trojice.

Příklad 3.2: Syntaxe N3

```
@prefix vip: <http://www.vip.cz#>
@prefix poj: <http://www.pojmy.cz#>

vip:novak poj:adresa _:adresa .
_:adresa poj:mesto “Brno” ;
poj:ulice “Česká 5” ; poj:psc “621 00” .
```

Příklad 3.2 Syntaxe N3

3.3.3 RDF/XML syntaxe

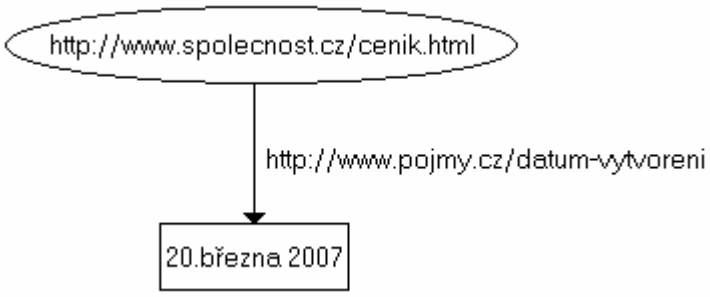
RDF/XML syntaxe

Pro potřeby sémantického webu je velmi vhodnou reprezentací tzv. XML syntaxe RDF, někdy označovaná jako RDF/XML. K zapsání tvrzení reprezentovaných pomocí RDF grafu se využívá jak elementů a jejich obsahů tak atributů a jejich hodnot.

RDF graf si můžeme představit jako konečnou množinu cest od kořene k listům. Tuto množinu je třeba zapsat pomocí XML jako vnořené elementy, které vytvoří stejnou strukturu. (Dokument ve formátu XML lze jednoduše transformovat na uspořádaný strom). Hlavní uzel, *subjekt* tvrzení, se stane hlavním elementem XML dokumentu. Jeho *predikátová* hrana bude prvním vnořeným elementem jehož hodnota může být *objekt* tohoto tvrzení. Tento *objekt* se stane *subjektem* dalšího tvrzení a stejným způsobem lze pokračovat dále.

Základní princip

Nejdříve vysvětlíme základní převedení jedné trojice RDF grafu do syntaxe RDF/XML. Jako tvrzení použijeme větu, která může být v přirozeném jazyce zapsána: *„Dokument http://www.spolecnost.cz/cenik.html byl vytvořen 20.března 2007“*. Následující obrázek zachycuje RDF graf tohoto tvrzení:



Obrázek 3.3 RDF graf – datum vytvoření dokumentu

Příklad 3.3: RDF/XML syntaxe

Pomocí syntaxe RDF/XML by stejné tvrzení vypadalo následovně:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
         xmlns:poj="http://www.pojmy.cz/">

<rdf:Description
rdf:about="http://www.spolecnost.cz/cenik.html">
<poj:datum-vytvoreni>20.Března 2007</poj:datum-
vytvoreni>
</rdf:Description>
</rdf:RDF>
```

Příklad 3.3 RDF/XML syntaxe

SPUSTIT

- Na prvním řádku je uvedeno povinné označení XML dokumentu a jeho verze.
- Na dalším řádku začíná obsah elementu <*rdf:RDF*>, který říká, že následující obsah XML dokumentu se bude týkat RDF. Prvním z atributů tohoto elementu je deklarace jmenného prostoru *xmlns:rdf* označující, že všechny odkazy doplněné o prefix *rdf* jsou součástí jmenného prostoru definovaného URI referencí, která je hodnotou tohoto atributu. Následuje deklarace smyšleného jmenného prostoru *xmlns:poj*.
- Další element <*rdf:Description*> je základní element, který označuje popis nějakého zdroje definovaného pomocí atributu *rdf:about*. Hodnota tohoto atributu představuje *subjekt* tvrzení.
- Následující vnořený element <*poj:datum-vytvoreni*> představuje vlastnost (*predikát*) tvrzení a jeho hodnota "20.března 2007" představuje *objekt* tvrzení.

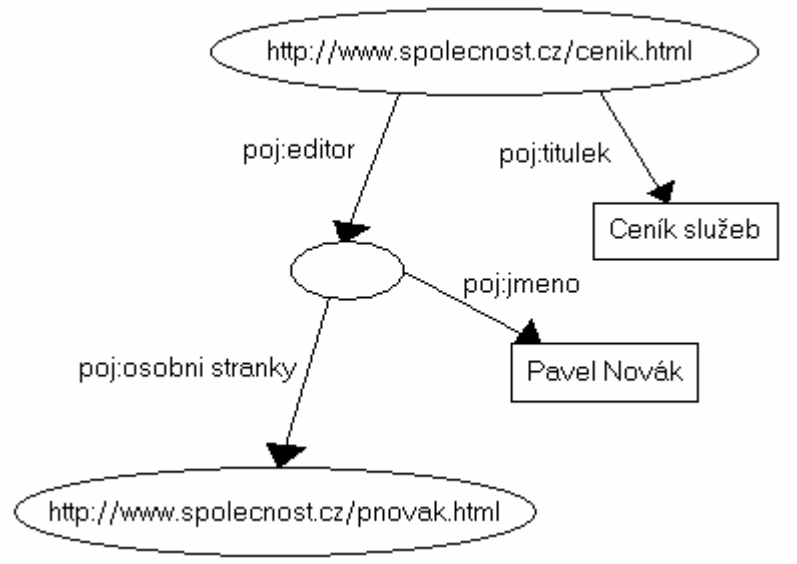
Pro doplnění dodejme, že při použití syntaxe N3 bychom takové tvrzení zapsali takto:

```
<http://www.spolecnost.cz/cenik.html>
  poj:datum-vytvoreni "20.března 2007" .
```

RDF graf s prázdným uzlem

Složitější RDF graf s prázdným uzlem

V dalším příkladu popíšeme RDF/XML syntaxi o něco složitějšího RDF grafu, který obsahuje také prázdný uzel. Jak již bylo řečeno v úvodu této kapitoly, je třeba vyhledat všechny cesty v grafu od hlavního uzlu (kořene) ke všem listům.



Obrázek 3.4 RDF graf

Obrázek 3.4 obsahuje RDF graf zachycující několik tvrzení o zdroji, dokumentu, reprezentovaném pomocí URI reference *“http://www.spolecnost.cz/cenik.html“*. Tvrzení zaznamenávají, že dokument je opatřen titulkem, a že byl editován editorem, který je identifikován URI referencí na osobní webové stránky a jménem. Osoba editora je v grafu reprezentována prázdným uzlem.

Pro úplnost tato tvrzení zapíšeme nejdříve pomocí N3 (využíváme zkráceného zápisu pomocí prefixů jmenných prostorů, definovaných na začátku příkladu:

Příklad 3.4 N3 syntaxe RDF grafu z obrázku 3.4

```
@prefix poj: <http://www.pojmy.cz/>
@prefix sp: <http://www.spolecnost.cz/>

sp:cenik.html poj:editor _:uzel .
sp:cenik.html poj:titulek “Ceník služeb” .
_uzel poj:osobni_stranky sp:pnovak.html .
_uzel poj:jmeno “Pavel Novák” .
```

Příklad 3.4 N3 syntaxe RDF grafu z obrázku 3.4

Při průchodu grafem za účelem jeho zapsání pomocí syntaxe RDF/XML zjistíme, že grafem vedou celkem tři cesty od kořene k listům. Pokud bychom tedy chtěli zapsat všechny tyto cesty, byla by struktura výsledného dokumentu následující:

Příklad 3.5 RDF/XML Všechny cesty RDF grafu z obrázku 3.4

```
<rdf:Description rdf:about=”...cenik.html”>
  <poj:editor>
    <rdf:Description>
      <poj:osobni_stranky>
        <rdf:Description rdf:about=”...novak.html”>
        </rdf:Description>
      </poj:osobni_stranky>
    </rdf:Description>
  </poj:editor>
</rdf:Description>

<rdf:Description rdf:about=”...cenik.html”>
  <poj:editor>
    <rdf:Description>
      <poj:jmeno>Pavel Novák</poj:jmeno>
    </rdf:Description>
  </poj:editor>
</rdf:Description>

<rdf:Description rdf:about=”...cenik.html”>
  <poj:titulek>Ceník služeb</poj:titulek>
</rdf:Description>
```

Příklad 3.5 RDF/XML syntaxe RDF grafu z obrázku 3.4

SPUSTIT

Ale tím bychom vůbec nevyužili toho, že prázdný uzel od určitého místa spojuje dvě z těchto cest. V RDF grafech je běžné, že z jednoho subjektu vychází více predikátů. RDF/XML proto poskytuje možnost spojení takových vlastností do jednoho elementu. V našem případě to znamená, že tělo výsledného dokumentu bude obsahovat dva elementy *<rdf:Description>*. První z nich bude popisovat *subjekt* představující RDF grafem popisovaný dokument:

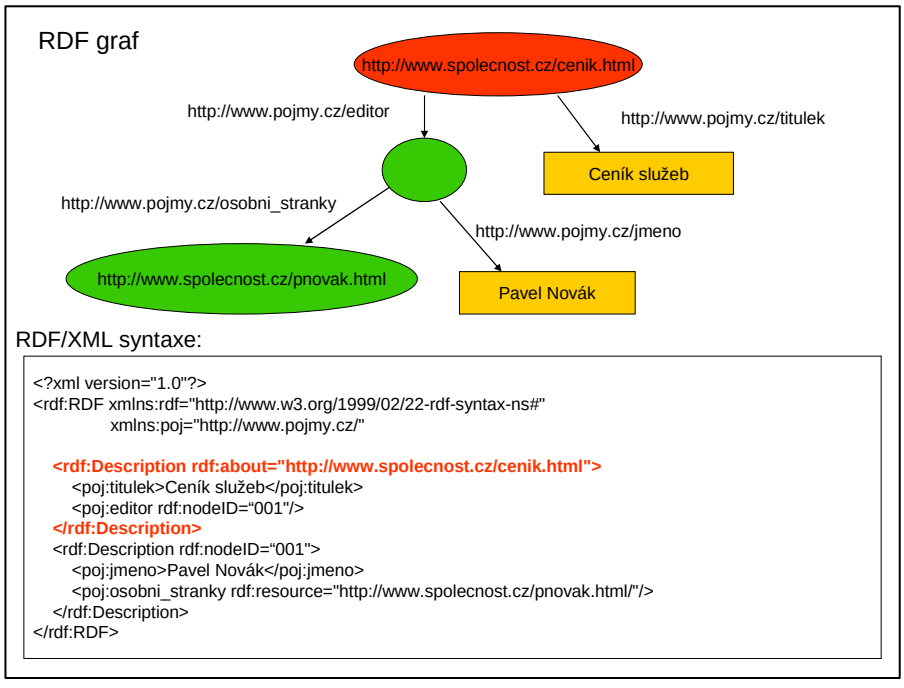
```
<rdf:Description
rdf:about="http://www.spolecnost.cz/cenik.html">
        <poj:titulek>Ceník služeb</poj:titulek>
        <poj:editor rdf:nodeID=“001"/>
</rdf:Description>
```

a bude v sobě obsahovat dva elementy (tedy znázorňuje dvě trojice). Element *<poj:titulek >* je *predikátem* a jeho hodnota *objektem*. Element *<poj:editor>* je také *predikátem*, ale hodnotu, *objekt*, představuje hodnota jeho atributu *rdf:nodeID* a znamená označení ID prázdného uzlu.

Podobně i druhý element *<rdf:Description>* bude obsahovat dva vnořené elementy, jejichž význam je zřejmý:

```
<rdf:Description rdf:nodeID=“001">
     <poj:jmeno>Pavel Novák</poj:jmeno>
     <poj:osobni_stranky
rdf:resource="http://www.spolecnost.cz/pnovak.html/">
     </poj:osobni_stranky>
</rdf:Description>
```

RDF graf RDF/XML syntaxe



4 Ontologie

4.1 Vymezení pojmu

Pojem Ontologie je ve filosofii znám již velmi dlouhou dobu. Lze jej vyložit jako nauku o bytí, nebo také Univerzální soustavu znalostí popisující jevy, zákonitosti a objekty světa. V oblasti informačních technologií se tento pojem začal objevovat až na začátku 90. let. Význam slova Ontologie se v této oblasti vědy rozšiřuje nejen na souhrny znalostí, ale také na metody získávání těchto znalostí.

4.1.1 Definice

Definice pojmu ontologie

Existuje velké množství definic ontologií, ale jednou z nejzákladnějších formulací je:

„*Ontologie je explicitní specifikace konceptualizace"* (T. Gruber, 1993)

Tuto definici v roce 1997 upravil W. Borst na :

„*Ontologie je formální, explicitní specifikace sdílené konceptualizace*".

- *Konceptualizací* je myšlen systém pojmů, kterými lze popsat (modelovat) určitou část reálného světa.
- V obou definicích vystupuje termín *explicitní*, je tedy třeba jednoznačně definovat typ konceptu a podmínky jeho použití.
- Druhá definice klade důraz na *formální* specifikaci. Je tedy třeba užití nějakého konkrétního jazyka s definovanou syntaxí, aby bylo možné strojové zpracování.
- Další podmínkou je, aby konceptualizace byla *sdílená*. Nejde tedy o znalosti jedince, ale znalosti určité zájmové skupiny lidí.

4.1.2 Účel ontologií

Účel ontologií

Mezi základní způsoby využití ontologií patří:

- podpora porozumění mezi lidmi, například mezi experty a znalostními inženýry
- podpora komunikace mezi počítačovými systémy (interoperabilita)
- usnadnění návrhu aplikací orientovaných na znalosti (inteligentní výukové systémy, pojmové vyhledávání, zpracování přirozeného jazyka, apod.)

4.2 Členění ontologií

4.2.1 Členění podle oborových oblastí

Členění ontologií podle oborových oblastí

Ontologie lze dělit podle různých hledisek. V návaznosti na historický význam ontologií a jejich vlivu na současné obory využití ontologií můžeme ontologie rozdělit na terminologické, informační a znalostní.

Terminologické ontologie

Terminologické, nebo také lexikální, ontologie lze přirovnat k tezaurům, používaných převážně v knihovnictví, ale i v dalších oborech orientovaných na textové zdroje. Hlavní částí takových ontologií jsou termíny, které již nejsou dále formálně definovány.

Informační ontologie

Informační ontologie jsou rozvinutím databázových konceptuálních schémat. Představují nadstavbu nad primárními zdroji, pro které zajišťují konceptuální abstrakci a vyšší úroveň kontroly integrity, než je tomu u běžných nástrojů. Konceptuální abstrakce (abstraktní zachycení pojmů modelujících část reálného světa) je nezbytnou podmínkou například pro tzv. pojmové vyhledávání informací.

Znalostní ontologie

Znalostní ontologie navazují na výzkum reprezentace znalostí v rámci umělé inteligence. Ontologie v tomto případě znamenají spíše logické teorie a jejich vazba na reálné objekty světa je oproti informačním ontologiím relativně volná. Koncepty takových ontologií jsou definovány pomocí formálního jazyka. Právě znalostní ontologie mají zásadní význam pro sémantický web a jeho technologie a proto se budeme v dalším výkladu ontologií zaměřovat právě na tento typ ontologií.

4.2.2 Členění podle předmětu formalizace

Členění ontologií podle předmětu formalizace

Pravděpodobně nejběžnějším způsobem kategorizování ontologií je právě členění podle předmětu formalizace (zdroje konceptualizace). Podle různých autorů opět existuje více variant, ale obecně patří mezi základní typy: generické, doménové, úlohové a aplikační ontologie.

Generické ontologie

Generické ontologie bývají někdy označovány jako ontologie vyššího řádu. Tyto ontologie zachycují obecné zákonitosti, které nejsou vázány na konkrétní věcnou oblast. Příkladem může být zachycení zákonitostí času, vzájemné pozice objektů apod. Tím, že generické ontologie zachycují nejobecnější pojmy a vztahy, bývají považovány za základ struktury dalších typů ontologií.

Doménové ontologie

Doménové ontologie jsou asi nejvíce užívaným typem ontologií. Vždy se soustředí na popis pojmů, vztahujících se ke konkrétní věcné oblasti, kterou lze libovolně ohraničit. Mohou existovat doménové ontologie popisující celou lékařskou vědu, nebo naopak ontologie, jejíž doménou budou například pouze kožní onemocnění apod.

Úlohové ontologie

Úlohové ontologie jsou někdy označovány jako reprezentační ontologie, nebo také meta-ontologie. Hlavním rozdílem proti předchozím typům ontologií je, že úlohové ontologie nezachycují znalosti z reálného světa tak jak jsou, ale snaží se o jejich odvozování.

Využitím tohoto typu ontologií jsou modely znalostních úloh a metod jejich řešení. Mezi úlohy typické pro zpracování pomocí takových znalostních modelů patří například diagnostika, zhodnocení nebo plánování.

Aplikační ontologie

Aplikační ontologie jsou velmi specifickým typem ontologií. Obecně sice zahrnují jak doménovou, tak i úlohovou část, ale tyto modely jsou upraveny pro potřeby konkrétní aplikace.

4.3 Struktura ontologií

Struktura ontologií

Přestože existuje více typů ontologií, základní struktura znalostních ontologií je ve všech projektech, jazycích a nástrojích obdobná. Avšak používaná terminologie může být někdy nejednoznačná. V následujících podkapitolách popíšeme základní součásti, které tvoří strukturu ontologií a pokusíme se vymezit používané pojmy. Konkrétní příklady a detaily struktury ontologií budou vysvětleny v kapitole 5.

4.3.1 Třídy, koncepty

Třídy, koncepty

Základním stavebním kamenem znalostních ontologií jsou třídy, které můžeme definovat jako hierarchicky uspořádané množiny konkrétních objektů. Termín třída bývá v některých zdrojích zaměňován termínem *koncept* (tedy pojem) nebo *kategorie*.

Na rozdíl od tříd, které známe například z objektově orientovaných programovacích jazyků, nezahrnují ontologické třídy procedurální metody. Interpretace třídy se opírá o pojem relace, tedy jinak řečeno: *„třída odpovídá unární relaci na určité doméně objektů“*.

4.3.2 Instance, individua

Instance, individua

Pojem *individuum* představuje konkrétní objekt reálného světa. Je tedy jedním z množiny objektů, které mohou být označeny pomocí *třídy*. Záměrně říkáme "mohou", protože individuum může být do ontologie vloženo i bez příslušnosti k nějaké třídě (avšak tato možnost není některými ontologickými jazyky podporována). Pojem *instance* je chápán jako ekvivalent individua, ale na rozdíl od něj instance přímo asociuje příslušnost k nějaké třídě. Instance třídy vlastně označuje "člena" třídy.

4.3.3 Relace, sloty

Relace, sloty

Relace v ontologiích vyjadřují vztah mezi objekty (individui). Tyto objekty potom vytváří n-tice, které nazýváme instancemi relací. V tradičních jazycích mohou být pojmenované relace specifikovány pomocí libovolných logických podmínek. V některých jazycích jim však lze přiřadit pouze předdefinovaná omezení.

Existují také relace pouze mezi dvěma objekty (binární relace), které se nazývají *vlastnosti* nebo *sloty*. Takové relace nejsou nijak svázány s konkrétními třídami, definují pouze vztahy mezi jednotlivými objekty.

Zvláštním typem relace jsou takzvané *funkce*. Poněkud formálněji můžeme funkční sloty definovat jako relace, u nichž je hodnota n-tého argumentu jednoznačně určena předchozími n-1 argumenty. Někdy se funkční slot označuje jako *atribut*. Platí, že atribut je definován pro všechny instance třídy.

Stejně jako u tříd existují hierarchická uspořádání také u relací. Hierarchie pak spočívá v tom, že argumenty podřízené relace tvoří podmnožinu argumentů nadřízené relace.

4.3.4 Omezení slotů, meta-sloty

Omezení slotů, meta-sloty

Relacím (slotům) je možné přiřazovat vlastnosti, někdy označované jako takzvané *meta-sloty*. Typickým příkladem meta-slotu může být vyjádření hierarchického vztahu (podřízenosti a nadřízenosti) slotů. Dalšími vlastnostmi slotů jsou mimo jiné *definiční obor* a *obor hodnot*, které jsou vymezeny konkrétními třídami. Takovéto vlastnosti slotů označujeme jako *globální omezení*, protože se vztahují ke slotu bez ohledu na jeho použití.

Často je však třeba omezit hodnoty (např. z konkrétního definičního oboru) slotu aplikovaného na konkrétní třídu. Zejména se jedná o omezení *kardinality* a oboru hodnot slotu. V takovém případě mluvíme o *lokálním omezení* slotů.

4.3.5 Primitivní hodnoty a datové typy

Primitivní hodnoty a datové typy

Jak již bylo řečeno, argumenty relací (slotů) mohou být reprezentovány buď pomocí objektů, v takovém případě hovoříme o *objektových slotech*, ale také pomocí *primitivních hodnot*, které žádnému objektu neodpovídají. V tom případě se jedná o takzvané *dato-typové sloty*.

Obor hodnot takového slotu může být definován pomocí některého základního datového typu (string, integer,…).

4.3.6 Axiomy, pravidla

Axiomy, pravidla

V ontologiích se nacházejí výrazy, které explicitně vymezují příslušnost k některým třídám a relacím. Avšak do ontologií lze zařazovat také logické, výrokové formule, které mohou vyjadřovat například ekvivalenci nebo disjunkci tříd, rozklad třídy na podtřídy apod. Tyto formule jsou nazývány *axiomy*, či *pravidly*. V závislosti na dané interpretaci jazyka (viz kapitola 5) mohou být axiomy součástmi tříd, nebo mohou vystupovat zcela samostatně.

5 Ontologické jazyky

5.1 Historické ontologické jazyky

Historické ontologické jazyky

Potřeba formálně reprezentovat ontologie vyústila ve vznik mnoha ontologických jazyků v poměrně krátkém období. Jedním z prvních takových projektů byl Cyc (název je odvozen od slova enCYClopedia) a byl zahájen již v roce 1984. Cílem bylo shromažďovat všeobecné znalosti, které by tvořily doplněk k expertním znalostem ve znalostních systémech.

Dalším jazykem, který vznikl na počátku 90. let, byl jazyk nazvaný Ontolingua. Cílem bylo vytvoření dostatečně přehledného jazyka, který by umožňoval sdílení ontologií mezi různými znalostními systémy. Hlavním záměrem tedy nebyla sama reprezentace ontologií, ale způsob jejich výměny a z toho vyplývá omezená možnost odvozování v tomto jazyce. Díky tomu začaly vznikat další jazyky, které měly umožnit, při zachování podstaty ontologií, přímou podporu programových aplikací bez nutnosti překládání do jiného jazyka.

Takovým jazykem byl například OCML (Operational Conceptual Modelling Language). Ten byl vybaven vlastním interpretem implementovaným v prostředí CommonLISP. Deklarativní část OCML byla prakticky stejná jako u jazyka Ontolingua, ale kromě toho byla podporována řada konstruktorů převzatých z procedurálních jazyků (podmínky, cykly,…).

Pro potřeby sémantického webu však bylo třeba vytvořit jazyky, které by splňovaly základní podmínky pro reprezentaci sémanticky psaných webových stránek.

5.1.1 SHOE

SHOE

Prvním z takových jazyků byl SHOE (Simple HTML Ontology Extension), který vznikl v roce 1996 na University of Maryland. Jazyk SHOE umožňuje vkládat přímo do kódu HTML stránek metadata o objektech, kterých se tyto stránky týkají a také přímo ontologie, které definují sémantiku těchto metadat. Oproti předchozím ontologický jazykům je jazyk SHOE mnohem jednodušší.

Deklarace ontologií se umisťovala na začátek kódu HTML stránky ve formátu:

```
<ONTOLOGY ID="identifikátor ontologie"
          VERSION="verze">
</ONTOLOGY>
```

Do těchto značek se vkládaly definice kategorií (tj. tříd):

```
<DEF-CATEGORY NAME="název kategorie">
```

a definice relací:

```
<DEF-RELATION NAME="název relace">
```

Anotace samotné stránky se potom vkládala do elementu s názvem instance. Samotná instance byla identifikována pomocí URL stránky:

```
<INSTANCE KEY="URL stránky">
```

Vnořené elementy pak označovaly používanou ontologii:

```
<USE-ONTOLOGY ID="identifikátor ontologie"
              URL="URL ontologie"
              VERSION="verze ontologie">
```

a také metadata o vlastníkovi stránky, například to, že je instancí nějaké kategorie (třídy):

```
<CATEGORY NAME="název kategorie">
```

5.1.2 Ontobroker

Ontobroker

Přibližně ve stejné době, kdy vznikal jazyk SHOE, se na univerzitě v Karlsruhe začal vyvíjet koncept jazyka Onotobroker. Stejně jako u jazyka SHOE byla hlavní myšlenkou možnost přidat k webovým stránkám sémantiku, avšak zde se zamýšlela poněkud jiná architektura, která měla být důsledně centralizovaná.

Hlavním předpokladem byla existence ontologického serveru, který by spravoval všechny ontologie a k němuž by měli přístup pouze kvalifikovaní uživatelé. Obecná koncepce také nepředpokládá, že data pro potřeby ontologií budou získávána náhodným přístupem k webovým stránkám, ale spíše cíleným navštěvováním stránek registrovaných poskytovatelů informací patřících k nějaké odborné komunitě.

V důsledku toho Ontobroker nepoužívá jednotný jazyk pro zachycení anotací a ontologií. Zatímco jazyk anotací představuje jednoduché přidání atributů přímo do HTML kódu, jazykem ontologií je propracovaný formalismus (tzv. F-logic). Následuje ukázka záznamu ontologie v tomto formátu:

```
Person :: Object.
Researcher :: Person.
Person [lastName =>> STRING;
        publication =>> Publication].
```

Jde o záznam hierarchie tříd. Pro třídu *Person* jsou zde definovány sloty *lastName* (dato-typový) a *publication* (objektový). Odpovídající znalostní anotace se potom vkládá přímo do HTML kódu v podobě atributů značek. Například k elementu označujícímu odkaz *<a>* se připojí atribut *onto*, který v následujícím příkladu označuje, že URL stránka (zastoupena symbolem *page*) je instancí třídy *Researcher*:

```
<a onto="page:Researcher">URL odkazu</a>
```

5.2 RDF Schéma

RDF Schéma

RDF schéma (zkráceně RDFS) není klasickým ontologickým jazykem. Výstižnějším označením by mohlo být: sémantické rozšíření formátu RDF. RDFS bylo vytvořeno v roce 1999 společností W3C. RDFS doplňuje do struktury RDF základní ontologické konstrukce, tedy třídy a binární relace s možností stanovit jejich definiční obor a obor hodnot. Kromě toho může být nad třídami i nad sloty definována hierarchie.

V následujících kapitolách (a také v kapitole 6.) budou postupně popsány jednotlivé části struktury ontologií. Ve všech příkladech budeme využívat pojmů z existující ontologie, která se zaměřuje na víno. Pojmy jsou definovány v angličtině a v zájmu zachování zpětné kompatibility s ontologií nebudou překládány. Pro přehlednost těchto příkladů si hned v úvodu deklarujeme jmenný prostor *wont* (wine ontology), definovaný URI referencí:
`http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#`

5.2.1 Třídy

Třídy RDFS

RDFS tedy k popisování vlastností zdrojů používá systém tříd (tabulka 5.1) a jejich vlastností. V následujícím textu se budeme věnovat popisu pouze hlavních RDFS tříd a vlastností, kompletní seznam těchto tříd a vlastností lze najít v [Brickley04].

Jednotlivé zdroje mohou být rozděleny do skupin, které nazýváme třídami. Členové těchto tříd jsou pak instancemi třídy. Třída sama o sobě může být také zdrojem, který lze identifikovat pomocí URI reference a lze jej popsat RDF vlastností.

Každá třída má svoji množinu instancí, která se nazývá rozšíření třídy. Platí tvrzení:

- dvě různé třídy mohou mít stejné množiny instancí, ale stále budou různými třídami.

Rozdíl je v tom, že instance obou tříd mohou být popsány pomocí různých vlastností. Pro názornost si lze dvě různé třídy představit jako dvě různé databáze se stejnými prvky, ale s různým popisem.

Fakt, že zdroj je instancí nějaké třídy se zapisuje pomocí RDF vlastnosti *rdf:type*. Je třeba zdůraznit, že k reprezentaci RDF a RDFS vlastností zdrojů se využívá atributů ze jmenného prostoru RDF a RDFS (viz kapitola 2.2.4).

Skupina zdrojů, které jsou třídami RDFS, je sama třída pojmenovaná *rdfs:Class*. Pokud tedy chceme definovat RDFS třídu, vytvoříme tvrzení o nějakém zdroji, který bude tuto třídu reprezentovat a přiřadíme mu vlastnost *rdf:type* jejíž hodnotou bude zdroj *rdfs:Class*.

V následujícím příkladu je uvedeno RDF tvrzení, které definuje zdroj *Wine* jako třídu RDFS (k reprezentaci je použita notace N3):

Příklad 5.1 Deklarace třídy RDFS

```
wont:Wine rdf:type rdfs:Class .
```

Příklad 5.1 Deklarace třídy RDFS

Podtřídy

Hierarchická struktura RDFS umožňuje definovat kromě tříd také takzvané podtřídy. Vlastnost, která označuje, že třída je podtřídou jiné třídy se zapisuje pomocí *rdfs:subClassOf*. Tato vlastnost je transitivní, tedy platí, že pokud je třída T1 podtřídou třídy T, potom všechny instance třídy T1 budou zároveň instancemi třídy T.

V následujícím příkladu je definována třída *RedWine* jako podtřída třídy *Wine* a další třída *DryRedWine* definovaná jako podtřída třídy *RedWine*. Tedy přeneseně řečeno, suchá červená vína patří do třídy červených vín a zároveň také do třídy všech vín.

Příklad 5.2 Deklarace podtříd RDFS

```
wont:RedWine rdfs:subClassOf wont:Wine .
wont:DryRedWine rdfs:subClassOf wont:RedWine .
```

Příklad 5.2 Deklarace podtříd

Třída	Popis
rdfs:Class	Třída všech zdrojů, které jsou RDFS třídami. Třída rdfs:Class je instancí sebe sama.
rdfs:Resource	Cokoliv popsaného pomocí RDF se nazývá zdrojem a je instancí této třídy. Tato třída obsahuje vše, jakákoliv další třída je podtřídou této třídy. Třída rdfs:Resource je instancí třídy rdfs:Class.
rdfs:Literal	Třída literálů, jejichž hodnoty jsou řetězce nebo čísla. Tato třída je instancí třídy rdfs:Class, ale je podtřídou třídy rdfs:Resource.
rdfs:Datatype	Třída datových typů. Je instancí i podtřídou třídy rdfs:Class, avšak každá instance této třídy je podtřídou třídy rdfs:Literal.
rdf:XMLLiteral	Třída speciálních datových typů nazvaných XML literál. Tato třída je instancí třídy rdfs:Datatype a podtřídou třídy rdfs:Literal.
rdf:Property	Třída RDF vlastností, je instancí třídy rdfs:Class.

Tabulka 5.1 RDFS třídy

5.2.2 Vlastnosti

Specifikace RDFS definuje RDF vlastnost jako relaci mezi zdrojem subjektu a zdrojem objektu. Takové vlastnosti jsou popisovány pomocí třídy *rdf:Property* a RDFS vlastností *rdfs:range*, *rdfs:domain* a *rdfs:subPropertyOf*. Všechny vlastnosti jsou v RDF instancemi třídy *rdf:Property*. Pokud tedy vytváříme novou vlastnost, použijeme k tomu již dříve zmiňovanou vlastnost *rdf:type*, pomocí které přiřadíme zdroj ke třídě *rdf:Property* tak, že vytvoříme RDF tvrzení ve tvaru:

Příklad 5.3 Deklarace vlastnosti RDFS

```
wont:hasMaker rdf:type rdf:Property .
```

Příklad 5.3 Deklarace vlastnosti RDFS

Vlastnost rdfs:range

Jednou z velice důležitých RDFS vlastností, které slouží k popisu vlastností specifických pro danou aplikaci, je vlastnost *rdfs:range*. Ta označuje, že hodnota (nebo hodnoty) deklarované vlastnosti budou zároveň instancemi třídy, která vystupuje jako objekt ve tvrzení, ve kterém je jako RDF vlastnost použita právě vlastnost deklarovaná pomocí *rdfs:range*.

Následující příklad obsahuje tvrzení, která definují třídu *WineColor* a vlastnost *hasColor*. Poslední řádek potom říká, že pokud bude v RDF tvrzení použita vlastnost *hasColor*, objekty tohoto tvrzení budou instancemi třídy *WineColor* (nebo dalších definovaných tříd).

Příklad 5.4 Použití vlastnosti rdfs:range

```
wont:WineColor rdf:type rdfs:Class .
wont:hasColor rdf:type rdf:Property .
wont:hasColor rdfs:range wont:WineColor .
```

Příklad 5.4 Použití vlastnosti rdfs:range

Vlastnost rdfs:domain

Další, již zmíněnou, velmi důležitou RDFS vlastností je *rdfs:domain*. Tato vlastnost je definována obdobně, jako předchozí vlastnost *rdfs:range*, nevztahuje se však k objektům tvrzení, ale k subjektům. Říká, že každý zdroj, kterému je dána vlastnost, definovaná pomocí *rdfs:domain*, je instancí jedné nebo více tříd.

Následující příklad obsahuje tvrzení, která definují třídu *Wine* a vlastnost *hasWineDescriptor*. Poslední řádek potom říká, že pokud bude v RDF tvrzení použita vlastnost *hasWineDescriptor*, subjektem tohoto tvrzení bude vždy instance třídy *Wine* (nebo dalších definovaných tříd).

Příklad 5.5 Použití vlastnosti rdfs:domain

```
wont:Wine rdf:type rdfs:Class .
wont:hasWineDescriptor rdf:type rdf:Property .
wont:hasWineDescriptor rdfs:domain wont:Wine .
```

Příklad 5.5 Použití vlastnosti rdfs:domain

Vlastnost rdfs:subPropertyOf

RDFS umožňuje definovat vztahy jak mezi třídami, tak také mezi vlastnostmi. K tomu slouží poslední zmiňovaná RDFS vlastnost: *rdfs:subPropertyOf*. Tato vlastnost určuje, že jedna vlastnost je speciálním případem jiné obecnější vlastnosti. V následujícím příkladu je vlastnost *hasSugar* definována jako pod-vlastnost vlastnosti *hasWineDescriptor*. Potom všechny instance třídy definované vlastností *hasSugar* budou zároveň instancemi tříd definovaných vlastností *hasWineDescriptor*.

Příklad 5.6 Deklarace pod-vlastnosti

```
wont:hasWineDescriptor rdf:type rdf:Property .
wont:hasSugar rdf:type rdf:Property .
wont:hasWineDescriptor rdfs:subPropertyOf
wont:hasSugar .
```

Příklad 5.6 Použití vlastnosti rdfs:subPropertyOf

5.3 Deskripční logika

Deskripční logika

Deskripční logiky jsou rodinou logických formalismů reprezentujících znalosti. Jejich cílem je zachycení struktury tříd a relací obdobně, jako je tomu u ontologií. Od tradičních ontologických jazyků se liší zejména tím, že nepředpokládají předem stanovený vztah třídy a podtřídy. Vztahy mezi třídami jsou vyhodnocovány na základě jejich popisu (deskripce), který je reprezentován pomocí logických výrazů. Operace nad těmito výrazy se potom sestávají převážně z:

- zjišťování vztahů (podřízenosti, nadřízenosti) mezi třídami
- testování splnitelnosti formulí vymezujících třídy
- zjišťování příslušnosti instancí ke třídám

5.3.1 Syntaxe

Syntaxe deskripční logiky

Terminologie deskripční logiky odpovídá konečné množině axiomů. Axiomy jsou konstruovány pomocí konceptů, rolí (binárních relací) a atributů. Pojmenované koncepty, role a atributy jsou považovány za atomické a pomocí nich lze tvořit další výrazy.

- Koncepty odpovídají třídám a jsou interpretovány jako množiny objektů.
- Role odpovídají relacím a jsou interpretovány jako binární relace na objektech.
- Výrazy jsou potom tvořeny za pomoci konstruktorů. Některé z nich jsou uvedeny v tabulce 5.2 (písmena *C, D* označují koncepty, *A* atribut a *R* název role)

Konstruktor	**Syntaxe**
Konjunkce	$C \sqcap D$
Disjunkce	$C \sqcup D$
Negace	$\neg C$
Existenční omezení	$\exists R.C$
Univerzální omezení	$\forall R.C$
Omezení počtu (kardinalita)	$(\leq n\ R)$, $(\geq n\ R)$
Existenční omezení atributu	$\exists A.C$
Univerzální omezení atributu	$\forall A.C$
Mapování hodnot atributů	$A = B$, $A \neq B$

Tabulka 5.2

5.3.2 Sémantika

Sémantika deskripční logiky

Výrazy mají množinovou sémantiku nad universem instancí Δ^I. Interpretační funkce $\cdot^I$ mapuje každý konceptuální výraz C na podmnožinu C^I universa Δ^I, každý relační výraz R na funkci s množinovými hodnotami (resp. binární

relaci) R^I nad universem Δ^I, a každý funkční výraz A na parciální funkci A^I s definičním oborem $dom(A^I) \subseteq \Delta^I$. Sémantika konstruktorů uvedených v předchozí tabulce je zapsána v tabulce 5.3.

Konstruktor	**Sémantika**
Konjunkce	$C^I \cap D^I$
Disjunkce	$C^I \cup D^I$
Negace	$\Delta^I \setminus C^I$
Existenční omezení	množina instancí $x \in \Delta^I$, pro které platí $R^I(x) \cap C^I \neq \emptyset$
Univerzální omezení	množina instancí $x \in \Delta^I$, pro které platí $R^I(x) \subseteq C^I$
Omezení počtu (kardinalita)	množina instancí $x \in \Delta^I$, pro které platí $\|R^I(x)\| \geq n$, nebo $\|R^I(x)\| \leq n$
Existenční omezení atributu	množina instancí $x \in dom(A^I)$, pro které platí $A^I(x) \in C^I$
Univerzální omezení atributu	množina instancí $x \in \Delta^I$, pro které platí $x \in dom(A^I) \Rightarrow A^I(x) \in C^I$
Mapování hodnot atributů	množina instancí $x \in \Delta^I$, pro které platí $A^I(x) = A^I(x)$, nebo $A^I(x) \neq A^I(x)$

Tabulka 5.3

5.4 DAML+OIL

DAML+OIL

Jak již napovídá název, jazyk DAML+OIL vznikl spojením dvou odlišných ontologických jazyků. Prvním z nich je DAML (DARPA Agent Mark-up Language), sponzorovaný vojenskou institucí DARPA. Cílem bylo vytvořit sémantický jazyk pro formát RDF, který by měl větší vyjadřovací sílu než RDFS. Ukázalo se, že je třeba postavit nové ontologické jazyky na nějakém propracovanějším logickém kalkulu, který by umožňoval konstrukci složitějších podmínek při zachování výhodných vlastností pro výpočty. Tímto logickým kalkulem se stala, výše zmiňovaná, deskripční logika. Na jejím principu byl vytvořen jazyk OIL (Ontology Inference Layer) a jeho sloučením s jazykem DAML vznikl nový ontologický jazyk DAML+OIL.

5.4.1 Třídy a axiomy

Třídy a axiomy

Základem jazyka DAML+OIL jsou třídy. Ty mohou být reprezentovány přímo pomocí URI, nebo pomocí logických výrazů. Logické výrazy nad třídami lze tvořit pomocí konstruktorů, které lze libovolně kombinovat. To umožňuje vytvářet i relativně složité výrazy. Seznam konstruktorů je uveden v tabulce 5.4. Vlastní obsah ontologií tvoří axiomy, vytvořené nad logickými výrazy reprezentujícími třídy. Seznam axiomů je uveden v tabulce 5.5.

Konstruktor	Popis
intersectionOf unionOf complementOf	Booleovské výrazy
oneOf	Množina primitivních hodnot
toClass hasClass	Existenční nebo univerzální omezení na slot (třídy, které jsou hodnotami slotu)
hasValue	Omezení na konkrétní hodnotu
minCardinalityQ masCardinalityQ CardinalityQ	Omezení na kardinalitu slotu

Tabulka 5.4 Konstruktory logických výrazů

Axiom	Popis
subClassOf	Vztah nadřazené a podřízené třídy
sameClassAs	Ekvivalence tříd
disjointWith	Prázdný průnik tříd
subPropertyOf	Vztah nadřazeného a podřízeného slotu
samePropertyAs	Ekvivalence vlastností
inverseOf	Vzájemně inverzní vlastnosti
transitiveProperty	Tranzitivita vlastnosti
uniqueProperty	Vlastnost je funkcí
unambiguousProperty	Inverzní vlastnost k dané vlastnosti je funkcí
sameIndividualAs	Totožnost individuí
differentIndividualFrom	Odlišnost individuí

Tabulka 5.5 Axiomy jazyka DAML+OIL

5.4.2 Struktura jazyka

Struktura jazyka DAML+OIL

Struktura ontologie reprezentované v jazyce DAML+OIL se skládá z hlavičky a dalších elementů tříd, elementů vlastností a instancí.

Hlavička

Hlavička DAML+OIL dokumentu se zapisuje ve formátu:

```
<daml:Ontology rdf:about="">
  <daml:versionInfo>Info o verzi
                    ontologie</daml:versionInfo>
  <daml:imports rdf:resource="URI importovaného
                              zdroje"/>
</daml:Ontology>
```

- Prázdný atribut *rdf:about=""* označuje, že popisovaným zdrojem bude aktuální dokument
- Element <daml:versionInfo> má pouze informativní charakter
- Element <daml:imports> slouží k importování dalšího zdroje, převážně dalších ontologií nebo jmenných prostorů

Třídy

Třída se označuje pomocí elementu <daml:Class> a může obsahovat další elementy (konstruktory tříd), které ji budou definovat (viz. tabulka 5.3). V následujícím příkladu je zobrazena definice třídy *Port* (Portské):

Příklad 5.7 Definice třídy v DAML+OIL

```
<daml:Class rdf:ID=”Port”>
  <rdfs:subClassOf rdf:resource=”RedWine”/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource=”hasSugar”/>
      <daml:hasValue rdf:resource=”Sweet”/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

Příklad 5.7 Definice třídy v DAML+OIL

- Třída Port je definována jako podtřída třídy RedWine pomocí elementu *rdfs:subClassOf*
- Dále také jako podtřída všech objektů, které vyhovují danému omezení umístěného mezi elementy *daml:Restriction*. To říká, že všechny objekty, které jsou hodnotami vlastnosti *hasSugar* musí pro tuto třídu nabývat hodnoty *Sweet*. Přeneseně řečeno: všechna Portská vína musí být sladká.

6 OWL

Na základě předchozích zkušeností s vývojem jazyka DAML + OIL vznikl nový projekt, jehož výsledkem bylo vytvoření nového ontologického jazyka, nazvaného OWL (Ontology Web Language). Za vznikem a vývojem toho jazyka opět stojí konsorcium W3C. První pracovní verze jazyka spatřily světlo světa už v roce 2002.

6.1 Verze jazyka OWL - Light, DL, Full

Verze jazyka OWL

Jazyk OWL nabízí dvě jakési podmnožiny, které by mohly zpřístupnit použití tohoto jazyka různým uživatelům a vývojářům s různou úrovní schopností nebo s různými nároky na implementaci konkrétního projektu. Verze OWL Light je navržena pro jednoduchou implementaci i pro začínající uživatele. Verze OWL DL (DL - deskripční logika) by měla nabízet vhodné výpočetní vlastnosti pro rozhodovací systémy. Kompletní jazyk OWL je pak kvůli odlišení od dalších verzí nazýván OWL Full.

Platí, že:

- Každá platná OWL Light ontologie je platnou OWL DL ontologií.
- Každá platná OWL DL ontologie je platnou OWL Full ontologií.

6.1.1 OWL Full, DL

OWL Full a OWL DL podporují stejné konstruktory, které jazyk OWL používá. Rozdíl mezi těmito verzemi spočívá v omezení použití některých těchto prvků a také v používání prvků RDF.

Verze OWL Full umožňuje libovolně propojovat OWL prvky s prvky RDFS a podobně jako RDFS netrvá na striktním oddělování tříd, vlastností, instancí a datových typů. Tato verze jazyka je navržena pro maximální kompatibilitu s formátem RDF.

Verze OWL DL naopak v jistém smyslu omezuje propojování OWL a RDF a také vyžaduje, aby byly třídy, vlastnosti, instance a datové typy vzájemně odděleny. Hlavním důvodem vzniku DL verze je existence mocných rozhodovacích systémů, které podporují ontologie svázané omezeními vyžadovanými právě pro DL verzi jazyka OWL.

OWL Full je, jak již název napovídá, kompletní verzí jazyka OWL bez žádných omezení. Je určena pro uživatele, kteří očekávají naprosto maximální výrazovou schopnost jazyka a volnost při tvorbě ontologií. V OWL Full je například možné pokládat třídu za soubor individuí, nebo také přímo za jedno individuum.

6.1.2 OWL Light

Verze OWL Lite je vhodná pro uživatele, kteří potřebují vytvořit základní klasifikační hierarchickou strukturu dokumentu s jednoduchými omezeními. OWL Lite používá jen některé prvky jazyka OWL a obecně se řídí omezeními, která jsou definována pro verzi DL. Například všechny třídy musí být pojmenované.

6.2 Syntaxe

Syntaxe OWL

OWL ontologie je ve své podstatě vytvořena pomocí RDF trojic a tvoří tedy RDF graf a proto může být reprezentována kteroukoliv konkrétní syntaxí vhodnou pro RDF (viz kapitola 3.3). Jak již bylo řečeno, nejpříhodnější syntaxí, doporučovanou konsorciem W3C, pro reprezentaci RDF trojic je RDF/XML syntaxe. Přípustná je jakákoliv forma této syntaxe.

Na následujícím řádku je příklad deklarace třídy *Wine* v OWL, reprezentované pomocí syntaxe RDF/XML, která bude používána i v dalších příkladech:

```
<owl:Class rdf:ID=”Wine”/>
```

Jen doplníme, že pomocí jiného typu syntaxe RDF/XML bychom mohli významově stejné tvrzení zapsat následovně:

```
<rdf:Description rdf:about=”Wine”>
  <rdf:type rdf:Resource=
        ”http://www.w3.org/2002/07/owl#Class”/>
</rdf:Description>
```

6.3 Struktura OWL dokumentu

Struktura OWL dokumentu

Struktura OWL dokumentu zachycuje jednotlivé pojmy ontologie. Obvykle obsahuje jednu hlavičku a následují definice tříd, vlastností a případně také individuí (instancí).

6.3.1 Hlavička

Hlavička

Hlavička dokumentu ontologie obsahuje některé důležité informace o ontologii. Může obsahovat také komentáře, informace o verzích dokumentu nebo o vnořených ontologiích. Následující příklad zobrazuje hlavičku OWL dokumentu, jehož obsahem je, výše zmiňovaná, ontologie o vínech.

Příklad 6.1 Hlavička

```
<owl:Ontology rdf:about="">
  <rdfs:comment>An example OWL ontology
  </rdfs:comment>
  <owl:priorVersion>
    <owl:Ontology rdf:about=
"http://www.w3.org/TR/2003/WD-owl-guide-
20030331/wine"/>
```

```
  </owl:priorVersion>
  <owl:imports rdf:resource=
"http://www.w3.org/TR/2003/CR-owl-guide-
20030818/food"/>

  <rdfs:comment>
Derived from the DAML Wine ontology at
http://ontolingua.stanford.edu/doc/chimaera/ontologie
s/wines.daml Substantially changed, in particular the
Region based relations.
  </rdfs:comment>
  <rdfs:label>Wine Ontology</rdfs:label>
</owl:Ontology>
```

Příklad 6.1 Hlavička

SPUSTIT

- Hlavička se uzavírá do značek <*owl:Ontology*>. Jejím atributem je prázdný atribut *rdf:about*, který vyjadřuje, že popisovaným subjektem bude vlastní dokument.
- Značka <*rdfs:comment*> označuje komentář, který nijak neovlivňuje strukturu dokumentu.
- Značka <*owl:priorVersion*> označuje předchozí verzi dokumentu (může být využito automatizovanými systémy). Hlavička může obsahovat také značku <*owl:versionInfo*>, která obsahuje označení aktuální verze (v nějakém předem dohodnutém formátu).
- Další značka <*owl:imports*> se využívá k importování jiné ontologie. K tomu využívá atribut *rdf:resource*.
- Poslední značka <*rdfs:label*> má podobný význam jako *rdfs:comment* a slouží k popisu zdroje, pro člověka čitelným textem.

6.3.2 Třídy

Třídy

Třída je základním hierarchickým prvkem ontologie. Je to uspořádaná struktura (množina) prvků, které spolu nějakým způsobem souvisejí. Tyto prvky se nazývají instancemi nebo individui třídy. Třídu jednoznačně popisuje její název (URI) a soubor prvků. Existuje několik způsobů, jak může být třída definována:

- Identifikátorem třídy (URI reference)
- Výčtem prvků, které budou tvořit instance třídy
- Omezením vlastností
- Sjednocením nebo průnikem dvou a více definicí tříd
- Doplňkem definice třídy

Třídy definované identifikátorem

Taková třída je definována pouze uvedením svého jména. Neobsahuje tedy žádné prvky ani vlastnosti.

Příklad 6.2 Třída definovaná identifikátorem

```
<owl:Class rdf:ID="Wine"/>
```

Příklad 6.2 Třída definovaná identifikátorem

Třídy definované výčtem prvků

Třída definovaná výčtem prvků je tzv. třída anonymní. Na takovou třídu se nelze přímo odkazovat, ale má využití ve složitějších konstrukcích (sjednocení, průniky…). V následujícím příkladu je třída *WineColor* definovaná výčtem prvků *White*, *Rose* a *Red*. Hodnotou vlastnosti *owl:oneOf* je seznam individuí, která se stanou instancemi třídy.

Příklad 6.3 Třída definovaná výčtem prvků

```
<owl:Class rdf:ID="WineColor">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#White"/>
    <owl:Thing rdf:about="#Rose"/>
    <owl:Thing rdf:about="#Red"/>
  </owl:oneOf>
</owl:Class>
```

Příklad 6.3 Třída definovaná výčtem prvků

SPUSTIT

Třídy definované omezením vlastností

Třída definovaná omezením vlastností je speciálním typem definice třídy. Popisuje anonymní třídu všech individuí, která budou odpovídat podmínkám omezení. Existují dva typy omezení: omezení hodnoty a omezení kardinality. Obecný formát definice třídy omezením vlastností vypadá následovně:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="nějaká vlastnost"/>
  (řádek s omezením hodnoty nebo kardinality)
</owl:Restriction>
```

Omezení hodnotou mohou využívat tří základních owl vlastností, viz následující tabulka:

OWL vlastnost	Popis
owl:allValuesFrom	Popisuje třídu všech individuí, pro která daná vlastnost nabývá pouze hodnot definovaných omezením (může to být i třída)
owl:someValuesFrom	Popisuje třídu všech individuí, pro která alespoň jedna hodnota dané vlastnosti nabývá hodnoty definované omezením.
owl:hasValue	Popisuje třídu všech individuí, pro která daná vlastnost nabývá hodnoty definované omezením (pouze jedna konkrétní hodnota).

Tabulka 6.1

Následující příklad ukazuje definici třídy definované omezením vlastnosti *hasSugar*, která musí nabývat hodnoty *Sweet*, tu bychom pak mohli použít například při definici třídy sladkých vín:

Příklad 6.4 Třída definovaná omezením vlastnosti

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasSugar"/>
  <owl:hasValue rdf:resource="#Sweet">
</owl:Restriction>
```

Příklad 6.4 Třída definovaná omezením vlastnosti

Obdobně je tomu u definování tříd omezením kardinality. Zde také existují tři owl vlastnosti, které lze při omezování kardinality využít, viz následující tabulka:

OWL vlastnost	**Popis**
owl:maxCardinality	Popisuje třídu všech individuí, pro která daná vlastnost nabývá maximálně počtu hodnot definovaných omezením
owl:minCardinality	Popisuje třídu všech individuí, pro která daná vlastnost nabývá alespoň počtu hodnot definovaných omezením.
owl:Cardinality	Popisuje třídu všech individuí, pro která daná vlastnost nabývá přesně počtu hodnot definovaných omezením.

Tabulka 6.2

Následující příklad ukazuje definici třídy definované omezením vlastnosti *madeFromGrape* na kardinalitu *1* (tedy víno vyráběné z jednoho typu hroznů).

Příklad 6.5 Třída definovaná omezením vlastnosti

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#madeFromGrape"/>
  <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNeg
ativeInteger">1
  </owl:maxCardinality>
</owl:Restriction>
```

Příklad 6.5 Třída definovaná omezením vlastnosti

Třídy definované průnikem

Další možností jak definovat třídy je pomocí množinových operací nad třídami. K definici třídy průnikem se využívá owl vlastnosti *owl:intersectionOf*. V příkladu 6.6 je uvedena definice třídy *WhiteWine* průnikem dvou tříd. Tou první je třída *Wine* (jejími instancemi jsou všechna vína) a třída definovaná omezením vlastnosti *hasColor* na hodnotu *White* (tedy všechna bílá vína).

Příklad 6.6 Třída definovaná průnikem

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasColor"/>
      <owl:hasValue rdf:resource="#White"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Příklad 6.6 Třída definovaná průnikem

SPUSTIT

Třídy definované sjednocením

Třída definovaná sjednocením bude obsahovat prvky, které náleží alespoň do jedné ze tříd, na které aplikujeme sjednocení. K tomu se využívá vlastnosti *owl:unionOf*. Následující příklad obsahuje definici třídy *WineDescriptor* pomocí sjednocení prvků dvou tříd *WineTaste* a *WineColor*.

Příklad 6.7 Třída definovaná sjednocením

```
<owl:Class rdf:ID="WineDescriptor">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#WineTaste"/>
    <owl:Class rdf:about="#WineColor"/>
  </owl:unionOf>
</owl:Class>
```

Příklad 6.7 Třída definovaná sjednocením

SPUSTIT

Třídy definované doplňkem

Definice třídy pomocí doplňku se od předchozích dvou definicí liší tím, že lze použít pouze jednu třídu, ke které budeme tvořit doplněk. Výsledkem tedy může být velmi velká třída, obsahující obrovské množství prvků. V ontologii o vínech se takto definovaná třída nevyskytuje, pravděpodobně by neměla význam. Pro úplnost uvedeme příklad, kdy pomocí doplňku ke třídě *WhiteWine* definujeme "abstraktní" třídu, jejíž prvky jsou všechna ostatní vína, tedy *Rose* a *RedWine*. Využijeme přitom vlastnosti *owl:complementOf*.

Příklad 6.8 Třída definovaná doplňkem

```
<owl:Class>
  <owl:complementOf>
    <owl:Class rdf:about="#WhiteWine"/>
  <owl:complementOf/>
</owl:Class>
```

Příklad 6.8 Třída definovaná doplňkem

Kromě výše popsaných možností definic tříd existují možnosti, jak upravit definice tříd pomocí dalších tvrzení. K tomu lze využít následující vlastnosti:

- rdfs:subClassOf – označuje, že prvky definované třídy jsou podmnožinou prvků jiné třídy (nadřazené).
- owl:equivalentClass – označuje, že množina prvků jedné třídy je přesně stejná jako množina prvků jiné třídy.
- owl:disjointWith – označuje, že množina prvků jedné třídy neobsahuje žádné společné prvky s množinou prvků jiné třídy.

6.3.3 Vlastnosti

Vlastnosti

Vlastnosti jsou jistým druhem binárních relací, díky kterým lze ke třídám a jejich prvkům přidávat tvrzení a spojení mezi jednotlivými objekty a datovými typy.

Vlastnost, která spojuje dva objekty, bývá označována jako objektová vlastnost a lze ji definovat elementem *owl:ObjectProperty*.

Vlastnost, která spojuje určitý objekt s nějakým datovým typem, označujeme jako dato-typová vlastnost a definujeme ji pomocí elementu *owl:DatatypeProperty*. Obecně má definice vlastnosti následující formát:

```
<owl:ObjectProperty rdf:ID="název vlastnosti">
  Další podmínky a typy vlastnosti
</owl:ObjectProperty>
```

Při definici vlastností využívá OWL následujících konstruktorů:

- RDFS konstruktory:
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range
- Vztahy mezi vlastnostmi:
  - owl:equivalentProperty
  - owl:inverseOf
- Globální omezení kardinality:
  - owl:FunctionalProperty
  - owl:InverseFunctionalProperty
- Logické charakteristiky
  - owl:SymmetricProperty
  - owl:TransitiveProperty

Vztahy mezi vlastnostmi

RDFS konstruktory již byly popsány v kapitole 5.2.2, proto se zaměříme na popis dalších konstruktorů. Konstruktor *owl:equivalentProperty* označuje, že dvě vlastnosti mají naprosto stejné prvky (hodnoty). Neznamená to však, že vlastnosti jsou stejné, mohou mít jiný význam.

Konstruktor *owl:inverseOf* se používá k označení vztahů v obou směrech, tedy doplnění jedné relace opačným případem. V následujícím příkladu je definována objektová vlastnost *producesWine* a inverzní vlastnost *hasMaker*.

Příklad 6.9 Inverzní vlastnost

```
<owl:ObjectProperty rdf:ID="producesWine">
  <owl:inverseOf rdf:resource="#hasMaker"/>
</owl:ObjectProperty>
```

Příklad 6.9 Inverzní vlastnost

Globální omezení kardinality

Konstruktor *owl:FunctionalProperty* označuje funkcionální vlastnost, která může mít pouze jednu jedinečnou hodnotu pro každou instanci. K vymezení hodnoty se často využívá konstruktorů rdfs:domain a rdfs:range. Jako příklad uvedeme objektovou vlastnost *hasColor*. Tím, že je označena funkcionální vlastností vlastně říkáme, že její hodnotou může být pouze jeden prvek ze třídy *WineColor* (víno má pouze jednu barvu):

Příklad 6.10 Funkcionální vlastnost

```
<owl:ObjectProperty rdf:ID="hasColor">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineColor"/>
</owl:ObjectProperty>

<owl:FunctionalProperty rdf:about="hasColor"/>
```

Příklad 6.10 Funkcionální vlastnost

Inverzní funkcionální vlastnost *owl:InverseFunctionalProperty* omezuje vlastnost v opačném směru. Neříká jaký může být objekt vlastnosti, ale vymezuje hodnotu subjektu.

Logické charakteristiky

Symetrická vlastnost, definovaná konstruktorem *owl:SymmetricProperty* označuje, že pokud dvojice prvků (a,b) je instancí dané vlastnosti, pak také dvojice (b,a) je instancí dané vlastnosti. V praxi to znamená, že konstruktory *rdfs:domain* a *rdfs:range* vymezující hodnotu vlastnosti bývají stejné.

Transitivní vlastnost, definovaná konstruktorem *owl:TransitiveProperty* označuje, že pokud dvojice prvků (a,b) je instancí dané vlastnosti a také dvojice prvků (b,c) je instancí dané vlastnosti, pak také platí, že dvojice (a,c) bude instancí dané vlastnosti.

7 Editor ontologií Protégé-owl

S přibývajícím zájmem o ontologie a jejich implementaci vznikají softwarové nástroje, které usnadňují tvorbu ontologií a jejich editování. Od vzniku ontologického inženýrství spatřilo světlo světa již velké množství podpůrných nástrojů, ale osudem většiny z nich bylo, že krátce po vzniku byl jejich další vývoj zastaven. Jedním z důvodů mohlo být neustálé experimentování a nejednotnost v názorech na vhodný jazyk pro reprezentaci znalostních informací. V poslední době se však zdá, že favoritem se stal jazyk OWL a začíná být hojně podporován novými editory.

Jedním z takových stabilnějších editorů je editor Protégé-owl. Je jedním z nejpoužívanějších editorů a mimo jiné také jeden z mála editorů, který je stále vyvíjen a jsou k němu vytvářeny další přídavné balíčky.

7.1 Instalace

Instalace editoru Protégé-owl

Editor Protégé-owl je tzv. freeware software a je distribuován pod licencí open-source Mozilla Public License. Je tedy možné jej volně stáhnout z Internetu. Aktuální stabilní verzi (Protégé-owl 3.2.1) lze najít na adrese:

http://protege.stanford.edu/download/release/full/

Editor Protégé-owl je implementován pomocí programovacího jazyka Java a pro spuštění je tedy třeba mít nainstalovaný virtuální stroj Java Virtual Machine. Ten je však možno nainstalovat přímo při instalaci editoru, resp. stáhnout verzi instalačního balíčku, ve kterém je Java Virutal Machine obsažen.

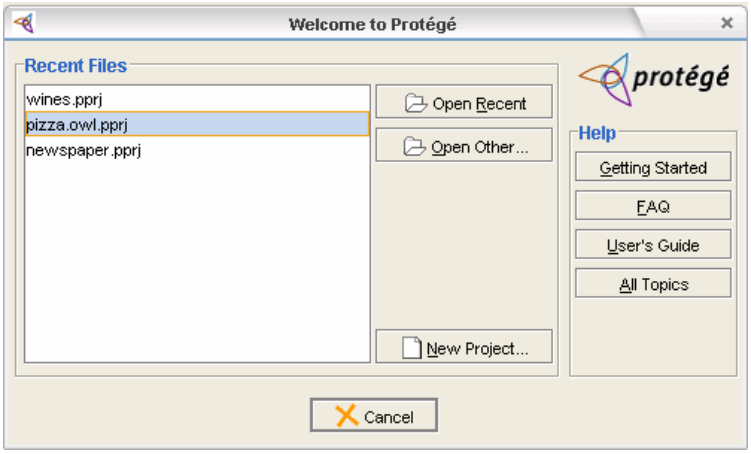
Jak již bylo uvedeno výše, pro tento editor existují přídavné balíčky, které přidávají nové možnosti a funkce. Jedním z takových vhodných doplňků je OWLViz. Jak již napovídá název, jde o vizualizační balíček, díky kterému lze zobrazit graf hierarchie tříd. Tyto grafy lze exportovat do různých grafických formátů. V aktuální verzi editoru je již tento balíček importován, ale k jeho správné funkci je třeba nainstalovat podpůrný nástroj GraphViz. Verze pro systémy Windows a Linux lze stáhnout na adresách:

- Windows: http://www.graphviz.org/Download_windows.php
- Linux: http://www.graphviz.org/Download_linux.php

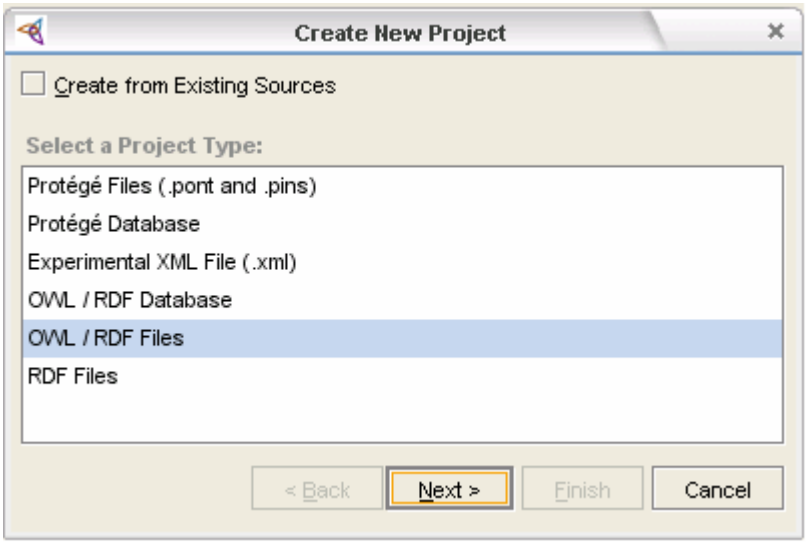
7.2 Vytvoření nové ontologie

Vytvoření nové ontologie

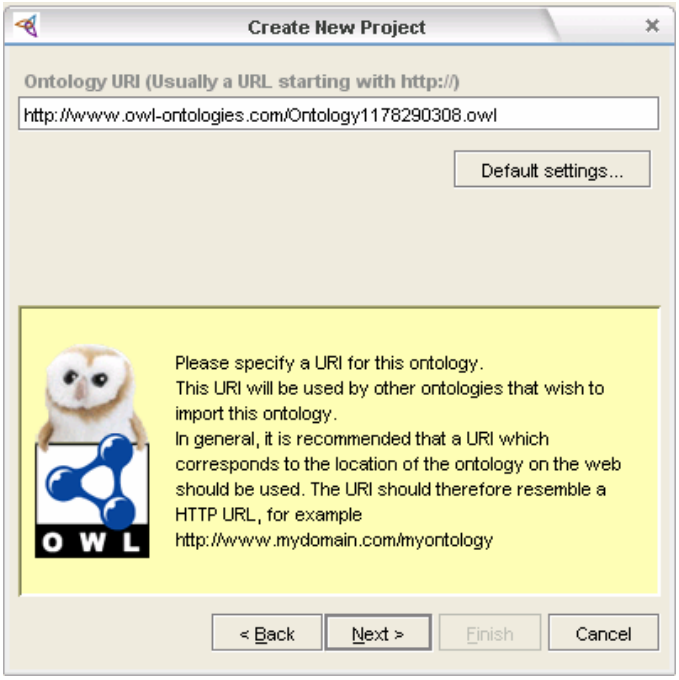
Po spuštění programu se zobrazí uvítací dialogové okno, kde je možné zvolit otevření existujícího projektu, nebo vytvoření nového projektu, viz následující obrázek.



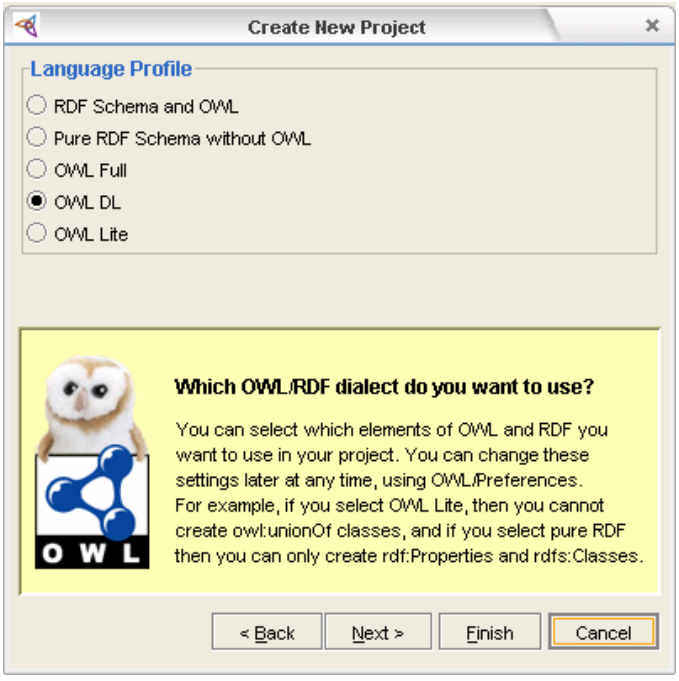
Pro vyzkoušení funkcí editoru je možné zvolit otevření některého z testovacích souborů, které jsou součástí instalovaného editoru. Pokud však zvolíme vytvoření nového projektu, zobrazí se další dialogové okno, ve kterém je třeba zvolit jaký formát dokumentu chceme vytvořit. Pro naše účely je vhodné zvolit formát OWL / RDF Files, viz následující obrázek:



Následující dialogové okno požaduje zadání jedinečného identifikátoru URI ontologie. Nastavení jedinečného URI při vytváření ontologie může zabránit pozdějším problémům například při importování jiných ontologií.



V posledním dialogovém okně je třeba zvolit úroveň OWL, která bude pro ontologii použita. Podle zvolené úrovně se upraví také uživatelské rozhraní a některé ovládací prvky budou, například v případě volby nižší úrovně, nedostupné.



7.3 Popis funkcí uživatelského rozhraní

Popis funkcí uživatelského rozhraní

Uživatelské rozhraní je rozděleno pomocí záložek na samostatné části. Nejdůležitějšími z nich jsou záložky týkající se tříd (karta OWL Classes) a vlastností (karta Properties). Jednotlivé karty lze zapínat/vypínat v hlavní nabídce: OWL > Preferences a v následně otevřeném dialogovém okně pod záložkou Tabs zaškrtnout/odškrtnout v seznamu možných záložek. Zde je také vhodné zaškrtnout kartu OWLVizTab, která nepatří mezi karty, jež jsou zvoleny při výchozím nastavení programu. Tato karta zpřístupní část uživatelského rozhraní, ve kterém se zobrazuje graf tříd ontologie.

V následujících podkapitolách představíme zmíněné základní části editoru a popíšeme jejich základní funkce. Detailní popis všech ovládacích prvků je ve formě tutoriálu, ale v anglickém jazyce, dostupný na adrese:

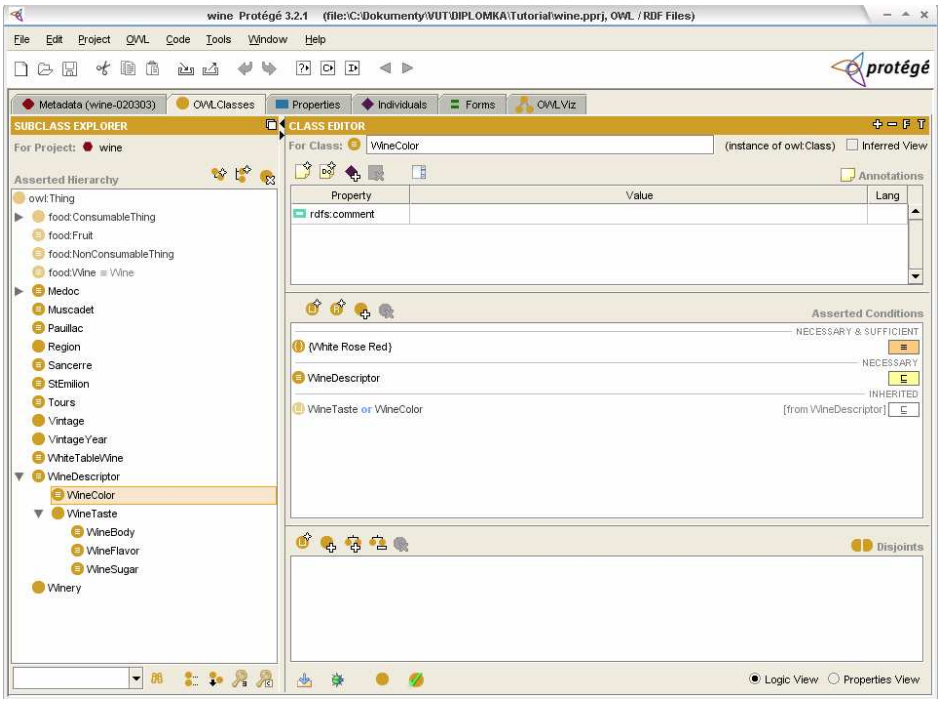
http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

7.3.1 Karta OWL Classes

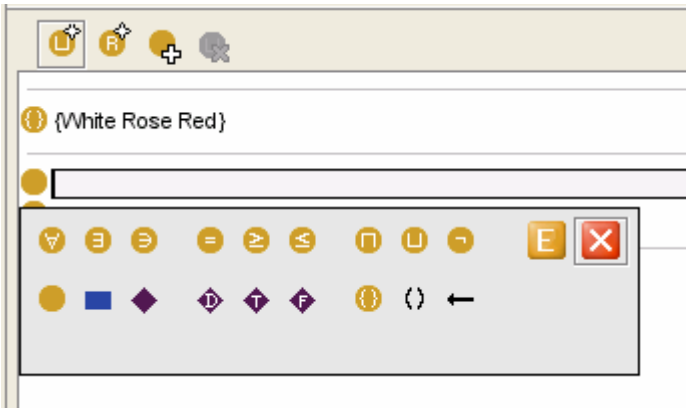
Karta OWL Classes

Tato část editoru je určena pro práci se třídami. V levé části se nachází uspořádaný strom tříd. Jako kořenová třída je vždy vložena třída owl:Thing, která je "nadtřídou" všech tříd. Další třídy vlastní ontologie se pak přidávají jako její podtřídy (subclasses).

V pravé části se nachází editor tříd, ve kterém je možné přidávat a editovat popisky třídy (rdfs:comment) a také zapisovat výrazy nebo omezení, které třídu definují.



Pomocí ikon (viz následující obrázek) lze vytvořit nová tvrzení nebo omezení pro danou třídu.



Význam symbolů pro vytvoření tvrzení o třídě nebo omezení třídy je následující:

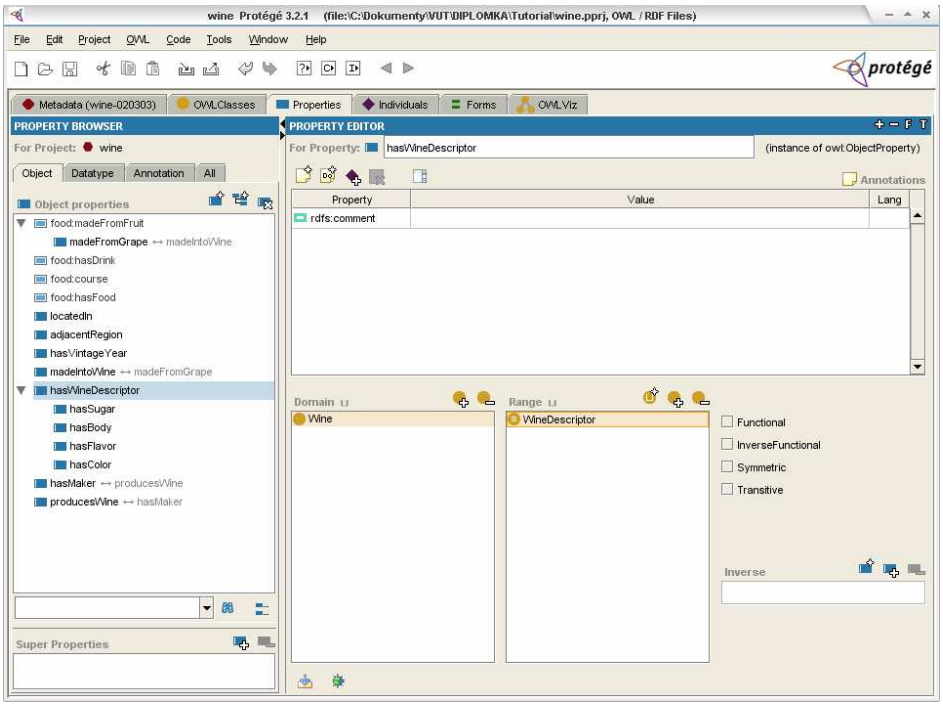
Symbol	OWL element
	allValuesFrom
	someValuesFrom
	hasValue
	cardinality
	minCardinality
	maxCardinality
	intersectionOf
	unionOf
	complementOf
	enumeration

7.3.2 Karta Properties

Karta Properties

Tato část uživatelského rozhraní slouží k tvorbě a editování vlastností (predikátů tvrzení). V levé části je opět hierarchicky uspořádaný seznam vlastností, který je ještě rozdělen na vlastnosti objektové, datotypové, anotační a je zde možnost zobrazit všechny vlastnosti najednou.

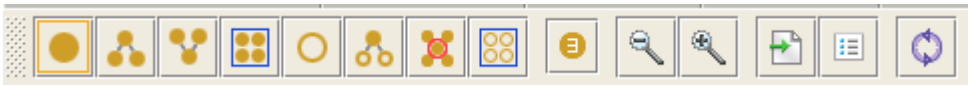
Pravá část slouží k upravování parametrů vlastností. Stejně jako u tříd je možné zapisovat poznámky a komentáře, ale hlavně nastavovat omezení vlastností pomocí rdfs:domain a rdfs:range. Mimo jiné lze také nastavit, zda bude daná vlastnost symetrická, transitivní, funkcionální nebo inverzně-funkcionální.



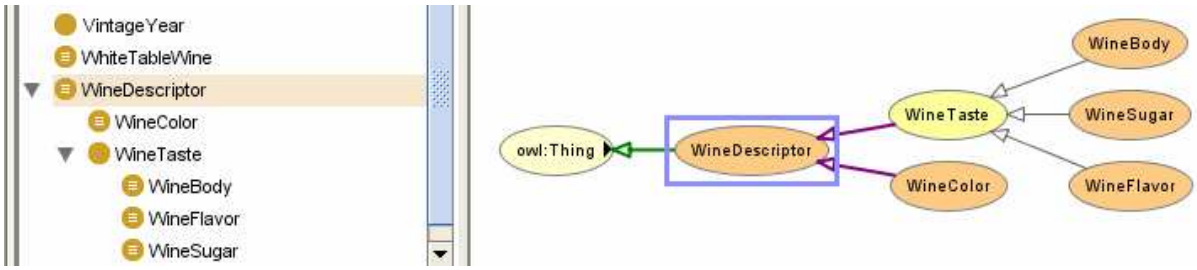
7.3.3 Karta OWLViz

Karta OWLViz

Poslední zmiňovanou kartou je karta OWLViz, jejíž funkce obsluhují graf tříd. Tato část editoru disponuje stejným prohlížečem tříd jako karta OWL Classes, avšak při výběru třídy se v pravé části zobrazí graf tříd a podtříd. Je zde také panel nástrojů (viz následující obrázek), který umožňuje nastavení výstupního formátu grafu, nastavení úrovně pro zobrazení podtřídy vybrané třídy, přiblížit/oddálit určitou část grafu apod.



Na následujícím obrázku je zobrazen výstupní graf tříd pro třídu WineDescriptor z ontologie vín, která byla použita v příkladech v kapitole 6.



Literatura

[Svátek02] Svátek V.: Ontologie a WWW, Katedra informačního a znalostního inženýrství, VŠE Praha, 2002. Dostupné z WWW: http://nb.vse.cz/~svatek/onto-www.pdf

[Bechhofer04] Bechhofer S., aj.: OWL Web Ontology Language Reference, W3C, 2004. Dostupné z WWW: http://www.w3.org/TR/owl-ref/

[Patel-Schneider04] Patel-Schneider P., Hayes P., Horrocks I.: OWL Web Ontology Language Semantics and Abstrakt Syntax, W3C, 2004. Dostupné z WWW: http://www.w3.org/TR/owl-absyn/

[Smith04] Smith M. K., Welty C., McGuinness D. L.: OWL Web Ontology Language Guide, W3C, 2004. Dostupné z WWW: http://www.w3.org/TR/owl-guide/

[Barners-Lee94] Barners-Lee T., aj.: Uniform Resource Locators (URL), Network Working Group, 1994. Dostupné z WWW: http://www.ietf.org/rfc/rfc1738.txt

[Moats97] Moats R.: URN Syntax, Network Working Group, 1997. Dostupné z WWW: http://tools.ietf.org/html/rfc2141

[Barners-Lee05] Barners-Lee T., aj.: Uniform Resource Identifier (URI), Network Working Group, 2005. Dostupné z WWW: http://tools.ietf.org/html/rfc3986

[Bratková99] Bratková E.: Metadata jako nástroj pro komunikaci webových informačních zdrojů, Národní knihovna ČR, 1999. Dostupné z WWW: http://knihovna.nkp.cz/Nkkr9904/9904178.html

[Klyne04] Klyne G., Carrol J. J.: Resource Description Framework Concept and Abstract Syntax, W3C, 2004. Dostupné z WWW: http://www.w3.org/TR/rdf-concepts/

[Becket04] Backet D.: RDF/XML Syntax Specification, W3C, 2004. Dostupné z WWW: http://www.w3.org/TR/rdf-syntax-grammar/

[Uschold01] Uschold M.: Where are the Semantics in the Semantic Web?, The Boeing Company, 2001. Dostupné z WWW: http://lsdis.cs.uga.edu/SemWebCourse_files/WhereAreSemantics-AI-Mag-FinalSubmittedVersion2.pdf

[Horrocks] Horrocks I.: Description Logics, Information Management Group University of Manchester. Dostupné z WWW: http://www.cs.man.ac.uk/~horrocks/Slides/IJCAR-tutorial/Print/p1-introduction.pdf

[Luke00] Luke S., Heflin J.: SHOE 1.01 Proposed Specification, 2000. Dostupné z WWW: http://www.cs.umd.edu/projects/plus/SHOE/spec.html

[Brickley04] Brickley D., Guha R. V.: RDF Vocabulary Description Language 1.0: RDF Schema, W3C, 2004. Dostupné z WWW: http://www.w3.org/TR/rdf-schema/

[Conolly01] Conolly D., aj.: DAML+OIL Reference Description, W3C, 2001. Dostupné z WWW: http://www.w3.org/TR/daml+oil-reference

Rejstřík