Ontologie a WWW

Vojtěch SVÁTEK*

Katedra informačního a znalostního inženýrství, VŠE Praha
Nám. W. Churchilla 4, 130 67 Praha 3
svatek@vse.cz

Abstrakt. Předmětem ontologického inženýrství je na jedné straně vývoj obecných jazyků, metodik a softwarových nástrojů, na druhé straně konstrukce samotných ontologií popisujících různé věcné oblasti, a aplikací, které je budou využívat. Podstatný nárůst zájmu o ontologie nastal v souvislosti s rozšířením WWW v komerční sféře, a se vznikem iniciativ, usilujících o využívání internetu jako prostředku výměny informací srozumitelných nejen pro člověka, ale i pro počítače. Plynulý přechod od současného WWW k sémantickému webu má být realizován prostřednictvím systematické tvorby a vkládání metadat. Pro jednoznačné vyjádření sémantiky používaných termínů je nutno použít jazyky vycházející právě z výzkumu v oblasti ontologií. Jako hlavní oblasti využití ontologií jsou v současnosti chápány: znalostní management, elektronické obchodování, zpracování přirozeného jazyka, inteligentní integrace informací z distribuovaných zdrojů, vyhledávání informací, sémantické webové portály, a inteligentní výukové systémy.

Klíčová slova: ontologie, znalostní inženýrství, World-Wide Web, sémantický web, metadata.

1 Úvod

Když se na začátku 90. let začal objevovat termín „ontologie“ jako označení specifického „informatického artefaktu“, vyvolalo to jistou nevoli ve filosofických kruzích, které se (vcelku oprávněně) obávaly zatemnění významu již zavedeného pojmu. Navzdory tomu se však použití termínu i samotných „informatických“ ontologií stále rozšiřovalo, a přerostlo až v módní vlnu. Neustálé zdokonalování možností sítě WWW si pak vyžádalo začlenění ontologií přímo do koncepce tohoto ústředního nástroje internetové komunikace. To vedlo, naštěstí pro uživatele, ke zvýšené snaze o standardizaci (nebo alespoň harmonizaci) vývoje, takže je dnes již pojem ontologie v informatice o něco přesněji vymezen než před deseti lety.

Cílem předkládaného textu[1] je pomoci čtenáři zorientovat se v aktuálním dění okolo ontologií, s důrazem na okruh aplikací souvisejících se sítí WWW, který je pro

* Výzkum autora je částečně podporován grantem GAČR č.201/00/D045 „Tvorba znalostních modelů ve vazbě na textové dokumenty“.

[1] Text je dostupný online na http://nb.vse.cz/~svatek/onto-www.pdf. Jeho zkrácená verze byla zveřejněna ve sborníku konference DATAKON 2002, viz http://www.datakon.cz.

vývoj ontologického inženýrství již delší dobu určující. Kapitola 2 objasňuje některé základní pojmy, se kterými se při práci s ontologiemi můžeme setkat. Kapitola 3 podává historický přehled nejvýznamnějších ontologických jazyků, se snahou poukázat na shodné a naopak odlišné rysy; největší prostor je ovšem věnován novým jazykům vyvíjeným specificky pro sémantický web. Kapitola 4 zmiňuje některé programové nástroje pro práci s ontologiemi, a podává přehled hlavních aplikačních oblastí. Konečně, kapitola 5 je sbírkou postřehů týkajících se dalšího směrování sémantického webu a role, kterou v něm ontologie budou sehrávat. Práce je opatřena třemi dodatky: vedle kolekce doplňkových odkazů jde o stručné představení jazyka RDF a deskripční logiky.

2 Základní pojmy ontologického inženýrství

2.1 Pojem ontologie

Hned na začátek je nutno poznamenat, že pojem ontologie v informatice je spíše jen volně spjat s filosofickým pojmem ontologie. Ve filosofii se ontologie chápe jako nauka (či soubor nauk) o „bytí“, popřípadě jako univerzální soustava znalostí popisující objekty, jevy a zákonitosti světa „tak jak je“ (tj. maximálně nezávisle na lidském usuzování o něm). Ontologie jako předmět praktického *ontologického inženýrství*, tj. jako „informační artefakt“, je do jisté míry odvozena z druhé možnosti: popisuje to, co „existuje“ a může být tudíž reprezentováno v informačním resp. znalostním systému. V celé zbývající části textu se již budeme zabývat pouze informatickým pojetím ontologie, a pokusíme se nejprve o jeho přesnější vymezení.

[50] uvádí celkem sedm historicky vzniklých definic pojmu ontologie, které se do značné míry překrývají, jiné zdroje zahrnují definic ještě více. Zde si z nich uvedeme pouze definici formulovanou T. Gruberem, jedním z „duchovních otců“ ontologií: „ontologie je explicitní specifikace konceptualizace“ [51], a její modifikaci provedenou W. Borstem: „…formální specifikace sdílené konceptualizace“ [42]. V první se požaduje pouze to, že *konceptualizace* (tj. systém pojmů modelující určitou část světa) musí být specifikována *explicitně*, tj. nikoliv jen „skryta“ v hlavě svého autora. Ve druhé již vystupuje požadavek jednak na *formalizaci*, tj. použití jazyka s přesně definovanou syntaxí (event. i sémantikou), jednak na *sdílenost* – ontologie není individuální záležitostí, nýbrž je výsledkem konsensu určité zájmové skupiny lidí. Tyto dva dodatečné požadavky již ovšem nebývají striktně dodržovány, např. znázornění struktury tříd a relací pomocí diagramu se samo o sobě někdy označuje jako (neformální či semi-formální) ontologie, a právě tak jsou za ontologie považovány i nově vzniklé modely, které dosud neprošly kolektivní diskusí, zejména pokud syntakticky vyhovují danému formálnímu jazyku.

Z hlediska znalostního inženýrství lze ontologie používané v procesu vývoje znalostní aplikace rovněž chápat jako *znalostní modely*, tedy abstraktní popisy (určité části) znalostního[2] systému, které jsou relativně nezávislé na finální reprezentaci a

[2] Termín „znalostní“ se v tomto kontextu objevuje ve dvou odlišných významech. Jednak jako vyjádření principu fungování systému, který je modelován (modelují se znalosti a ne třeba primární struktura dat nebo vzhled uživatelského rozhraní), jednak ve smyslu zavedeném

implementaci znalostí. Podstatné je, že jde o modely *sdílitelné* („sharable“) více procesy (např. softwarovými agenty) v rámci jedné aplikace, a *opakovaně použitelné* („reusable“) pro různé aplikace, které mohou být oddělené časově, prostorově i personálně.

2.2 Účel ontologií

Jako základní způsoby využití ontologií jsou tradičně jmenovány tyto (viz např. [73]):

- podpora porozumění mezi lidmi (např. experty a znalostními inženýry)
- podpora komunikace (interoperability) mezi počítačovými systémy
- usnadnění návrhu znalostně-orientovaných aplikací.

Ve všech třech rolích se ontologie mohou uplatnit v širokém spektru problémových oblastí a úloh. Oblasti a konkrétní aplikace si představíme v části 4.2.

2.3 Typy ontologií

Pojem ontologie je zastřešením pro mnoho heterogenních informačních artefaktů. Zde si stručně uvedeme hlavní dimenze členění (volně zpracováno podle [73] a [75]):

Členění podle „historických paradigmat“

Je zřejmé, že ontologie nejsou v informatice něčím zcela novým, zásadně odlišným – informační zdroje s obdobnou strukturou se již dříve vyskytovaly v různých disciplínách. V důsledku toho se problematika ontologií rozpadá (byť ne zcela disjunktně) na přinejmenším tři hlavní oblasti, které lze chápat jako součást vývoje tradičnějších oborů:

- *Terminologické* či *lexikální* ontologie lze ztotožnit s pokročilými tezaury, používanými v knihovnictví a dalších oborech orientovaných na textové zdroje. Jejich charakteristickým rysem[3] je ústřední role *termínů*, které již nejsou dále (formálně) definovány. Používané *relace* mají z velké části taxonomický charakter (vymezení vztahu obecnějšího a speciálnějšího termínu), vedle toho bývá vyjádřena synonymie, meronymie (vztah termínů označujících celek a jeho část) a další relace obecného charakteru. Nejznámější terminologická ontologie je nepochybně *WordNet* [35]; z něj byl odvozen např. *Sensus* [26] nebo vícejazyčná varianta *EuroWordNet* [9].
- *Informační ontologie* jsou rozvinutím *databázových* konceptuálních schémat. Hrají roli nadstavby nad primárními (strukturovanými, např. relačně-databázovými) zdroji, pro které zabezpečují jednak konceptuální abstrakci potřebnou pro pojmové dotazování, jednak vyšší úroveň kontroly integrity než běžné nástroje.
- *Znalostní ontologie* navazují na výzkum v oblasti reprezentace znalostí v rámci *umělé inteligence*. Ontologie jsou zde chápány důsledně jako *logické*

A. Newellem [63]: systém je modelován na tzv. úrovni znalostí („knowledge level“), ne na úrovni reprezentace/implementace čili konkrétních symbolů („symbol level“). Podrobnější rozbor souvisejících terminologických problémů přesahuje rámec tohoto textu.

[3] Ve srovnání s oběma zbývajícími typy ontologií (informačními a znalostními), které lze souhrnně charakterizovat jako *konceptuální*.

teorie, a jejich vazba na reálné objekty (instance) je oproti informačním ontologiím relativně volná. Třídy (koncepty) a relace jsou systematicky definovány prostřednictvím formálního jazyka.

Přestože se v důsledku vzájemných kontaktů všech tří odborných komunit v poslední době řada odlišností stírá, vědomí jejich existence stále ještě usnadňuje orientaci v používaných přístupech. Tento text se zabývá převážně znalostními ontologiemi.

Členění podle míry formalizace

Přestože je formalizace, jak již bylo řečeno, do jisté míry definiční vlastností ontologií, smysluplné využití mají i „ontologie" zcela neformální či „semi-formální". Jde zpravidla o glosáře, v nichž jsou jednotlivé pojmy vysvětleny přirozeným jazykem (volnou či strukturovanou formou). Ontologie vyjádřené ve formálních jazycích pak lze dále rozlišovat podle formálně-logických vlastností daného jazyka, jako je úplnost a rozhodnutelnost; tyto vlastnosti vycházejí z vlastností logického kalkulu, na kterém je jazyk založen, např. deskripční logiky (viz dodatek C). Formálním jazykům pro reprezentaci ontologií se budeme věnovat v kapitole 3.

Většina formálních ontologií v sobě ovšem svým způsobem zahrnují i ontologii neformální. Jednotlivé konstrukty bývají totiž vybaveny dokumentačních položkou, umožňující vyjádřit obsah přirozeným jazykem.

Členění podle předmětu formalizace

Jedná se o tradiční členění s řadou variant navržených různými autory; zde uvedeme pouze hlavní typy.

- *Doménové* ontologie jsou typem daleko nejfrekventovanějším. Jejich předmětem je vždy určitá specifická věcná oblast, vymezená šířeji (např. celá problematika medicíny nebo fungování firmy) či úžeji (problematika určité choroby, poskytování úvěru apod.). Příklady doménových ontologií se širokým záběrem jsou *Enterprise Ontology* [5] nebo lékařská *On9* [16].
- *Generické* ontologie usilují o zachycení obecných zákonitostí, které platí napříč věcnými oblastmi, např. problematiky času, vzájemné pozice objektů (topologie), skladby objektů z částí (mereologie) apod. Někdy se ještě výslovně vyčleňují tzv. ontologie *vyšší úrovně* („upper-level"), které usilují o zachycení nejobecnějších pojmů a vztahů, jako základu taxonomické struktury každé další (např. doménové) ontologie; nejnovějším výsledkem tohoto směru je *SUMO – Standard Upper Merged Ontology* [33]. Ontologie typu *common-sense* („přirozeného rozumu") mohou naopak obsahovat řadu velmi specifických, avšak relativně doménově-nezávislých znalostí, které lidé používají v každodenním životě. Nejznámějším příkladem je *Cyc*, viz část 3.1.
- Jako *úlohové ontologie* jsou někdy označovány generické modely znalostních úloh a metod jejich řešení. Na rozdíl od ostatních ontologií, které zachycují znalosti o světě („tak, jak je"), se zaměřují na procesy odvozování.

Mezi úlohy tradičně zachycené pomocí takových znalostních modelů patří např. diagnostika, zhodnocení („assessment“), konfigurace, nebo plánování[4].

- *Aplikační ontologie* jsou nejspecifičtější. Jedná se o konglomerát modelů převzatých a adaptovaných pro konkrétní aplikaci, zahrnující zpravidla doménovou i úlohovou část (a tím automaticky i generickou část).

2.4 Struktura ontologií

Přestože je základní struktura znalostních ontologií prakticky ve všech hlavních projektech, jazycích a nástrojích obdobná, používaná terminologie se značně liší, což znesnadňuje orientaci. Největší „propast“ je v tomto směru mezi tradičními, „silnými“ jazyky jako je Ontolingua (viz část 3.2) a novými[5], „odlehčenými“ jazyky, zvl. webovými. Pokusíme se uvést všechny podstatné konstrukty, zaznamenat terminologické odlišnosti, a naopak vymezit vůči sobě pojmy, které se často nesprávně zaměňují. Potřebné příklady budou podány formou přirozeného jazyka; možnou syntaxi konstruktů naleznete až v kapitole 3 věnované formálním jazykům.

Třídy, koncepty, kategorie, rámce

Základem znalostních ontologií jsou *třídy*, které označují množiny konkrétních objektů. Termínu třída odpovídá v některých formalismech termín *koncept* (tj. „pojem“), popřípadě *kategorie*, a úzce souvisí i s termínem *rámec* jako základním konstruktem mnoha systémů umělé inteligence.

Na rozdíl od tříd v objektově-orientovaných modelech a jazycích nezahrnují „ontologické“ třídy procedurální metody. Jejich interpretace je spíše odvozena z pojmu relace (viz níže), v tom smyslu, že třída odpovídá *unární relaci* na dané doméně objektů. Třídy, pro které jsou specifikovány *podmínky nutnosti i postačitelnosti* (příslušnosti individua) se označují jako *definované*, ostatní třídy (u kterých jsou specifikovány jen nutné podmínky nebo ani ty ne) pak jako *primitivní*. V tradičních jazycích mohou mít tyto podmínky tvar libovolného logického výrazu; v „odlehčených“ jazycích jsou přípustné jen předdefinované podmínky charakteru izolovaných omezení na konkrétním slotu, viz níže.

---

[4] Podrobnější výklad problematiky *generických modelů úloh* (v angličtině se nejčastěji používají termíny „problem-solving model“ nebo „problem-solving method“) přesahuje možnosti tohoto článku. Zájemce odkážeme na specializovanou literaturu. [43] je souborem původních prací, chápaných jako zásadní pro vývoj znalostního inženýrství (generickým modelům úloh je věnována zvl. kap. 2 „Expertise and Expert Systems“), lze využít spíše pro historický přehled. Novější směry jsou popsány ve sbornících série workshopů konaných v letech 1994-2000 jako součást konferencí IJCAI, ECAI a KAW, viz např. http://delicias.dia.fi.upm.es/WORKSHOP/ECAI00/. Nejznámější metodiku CommonKADS, která dokázala spojit problematiku znovupoužívání generických modelů znalostních úloh s obvyklým modelem organizace, a díky tomu jako jediná doznala rozšíření i v podnikové praxi, podrobně popisuje [69]. Nejnovějším směrem vývoje je pak využívání generických modelů úloh pro automatické zprostředkování znalostních komponent prostřednictvím WWW, podporované tzv. knihovnou UPML [49].

[5] Vymezení „nový“ nelze chápat kalendářně – stále vznikají i „silné“ jazyky (např. GOL [10]), naopak OKBC Lite (viz část 3.4) z poloviny 90.let je typickým příkladem „odlehčeného“.

Na množině tříd bývá definována *hierarchie* (taxonomie). Přestože „filosofický“ směr někdy zdůrazňuje požadavek na stromovou strukturu, v praxi všechny hlavní ontologické jazyky podporují vícenásobnou dědičnost, a tato možnost se v praxi hojně využívá.

Individua, instance

Individuum (angl. „individual“) odpovídá konkrétnímu objektu reálného světa, a je tak do jisté míry „protipólem“ třídy. Termín *instance* je často chápán jako ekvivalentní, asociuje však příslušnost k určité třídě, což nemusí být nutně skutečností – individuum může být do ontologie „provizorně“ vloženo i bez vazby na třídy.

Je ovšem třeba si uvědomit, že ontologie principiálně slouží k popsání konceptů (tříd) a nikoliv faktů o konkrétních objektech. Některé jazyky proto individua jako součást ontologie interně nepodporují, a pokud je potřeba individuum využít např. ve výrazu definujícím příslušnost k určité třídě[6], pracují s ním jako s atomickou třídou. Jiné přístupy se pokoušejí zahrnout pojmy (ontologie) a individua do společného nadřazeného modelu, např. v KAON [61] se uvažuje tzv. *ontology-instance model* (OIM), složený z ontologie a z jí odpovídající báze instancí. Takový přístup se přibližuje databázovým schématům (tj. informačním ontologiím, viz část 2.3).

Prohlášení určité entity za třídu nebo instanci často není dané objektivním stavem světa, ale závisí na úhlu pohledu. Typickým příkladem mohou být živočišné druhy: v ontologii popisující chování zvířat budou druhy (jakožto množiny svých příslušníků) zřejmě reprezentovány třídami, zatímco v ontologii fylogenetického vývoje (kde jednotlivá zvířata nemá smysl uvažovat) půjde spíše o individua, mezi kterými budou definovány relace typu *předchůdce*. Případná integrace takových ontologií vyžaduje zvláštní řešení (viz např. [61, 70]).

V minulosti se pod vlivem programovacích jazyků používal pro „vytčené instance“ i termín *konstanta*; většinou ovšem vesměs nešlo o pravé („objektové“) instance nýbrž o *primitivní datové hodnoty*, viz níže.

Relace, funkce, sloty, vlastnosti, role, atributy

Podobně jako v databázových modelech jsou podstatnou složkou ontologií vztahy čili *relace* n-tic objektů (individuí). V tradičních jazycích mohou být pojmenované relace (tak jako třídy, které jsou jejich speciálním případem) specifikovány pomocí libovolných logických podmínek. V „odlehčených“ jazycích je možné pouze přiřadit jim předdefinovaná omezení, ať už globálně nebo lokálně (ve vazbě na určitou třídu), viz níže.

Pro binární relace (na které se „odlehčené“ jazyky omezují) se používá pojem *slot*, vzniklý v oblasti znalostních systémů založených na rámcích[7], popřípadě *vlastnost*

---

[6] Příkladem takové definice je (ve slovním vyjádření) např.: „X je instancí třídy *anglicky-hovořící-člověk* právě tehdy, když je pro něj slot *mluví-jazykem* nabývá hodnoty (resp. jedné z hodnot) *angličtina*“, přičemž individuum *angličtina* je instancí třídy *jazyk*.

[7] Mezi reprezentací znalostí v rámcových systémech a v „odlehčených“ ontologických jazycích je z formálního hlediska poměrně malý rozdíl. Rámcové systémy jsou určeny pro usuzování nad konkrétními daty, předpokládají proto ve větší míře práci s instancemi (připouštějí např. možnost definovat sloty i pro jednotlivé instance, a nikoliv jen pro třídy). Odlišujícím znakem ontologií je dále důraz kladený na sdílitelnost a znovupoužitelnost znalostí, a

(„property“). To může sugerovat představu podřízenosti binárních relací třídám z jejich definičního oboru. Na rozdíl od objektově-orientovaného přístupu jsou však sloty (relace) v ontologii „občany první kategorie“. Nejsou napevno spojeny s žádnou třídou jako část její definice, a vazba na definiční obor (resp. obor hodnot) je zprostředkována pouze omezeními, viz níže. Naopak ve vztahu k objektu, který je jeho hodnotou, se někdy slot označuje jako *role*[8]; v deskripční logice (viz dodatek C) se tak označují binární relace všeobecně.

Jako zvláštní typ relace bývají chápány *funkce*; v souladu s běžným matematickým chápáním jde o relace, u nichž je hodnota *n*-tého argumentu jednoznačně určena předchozími *n-1*. Funkční *slot* se také označuje jako *atribut*, předpokládá se, že je definován pro všechny instance třídy; to ostatně odpovídá pojmu atributu v procedurálních programovacích jazycích.

Podobně jako třídy mohou mít i relace (sloty) nad sebou definovanou hierarchii. Lze ji interpretovat tak, že množina n-tic argumentů podřazené relace je podmnožinou množiny n-tic argumentů nadřazené relace. Příkladem mohou být třeba dvojice binárních relací *má-otce* a *má-předka*.

Meta-sloty, omezení na sloty, facety

Na rozdíl od relací v predikátovém kalkulu 1. řádu je v ontologiích možné slotům přiřazovat vlastnosti – mohli bychom je označit jako *meta-sloty*. Nejčastějším meta-slotem je právě *hierarchický* vztah podřazeného a nadřazeného slotu. Ten můžeme chápat jako tranzitivní binární (meta-)relaci. Symetrickou binární (meta-)relací je vztah slotu vůči slotu k němu *inverznímu*; ostatní meta-sloty vesměs odpovídají unárním relacím (vlastnostem slotu samotného) jako je *symetrie*, *tranzitivita*, *funkčnost* nebo *inverzní funkčnost*. Vedle obecných matematických vlastností patří mezi vlastnosti slotů také *definiční obor* („domain“) a *obor hodnot* („range“), vymezené pomocí konkrétních tříd.

Uvedené vlastnosti bychom mohli označit jako *globální omezení*, protože se vztahují ke slotu bez ohledu na způsob jeho použití. V řadě případů však potřebujeme vymezit hodnoty slotu aplikovaného na konkrétní třídu ze svého *definičního oboru*. Taková *lokální omezení* na slotech (“slot constraints”, „property restricions“) se označují jako *facety*[9]. V jazycích založených na deskripční logice víceméně odpovídají *konceptuálním výrazům* (viz dodatek C), protože vymezují určitý logický koncept[10]. Jde zejména o omezení *oboru hodnot* slotu a/nebo jeho *kardinality* –

---

zejména v posledních letech i na precizní formální sémantiku zabezpečenou prostřednictvím vazby na deskripční logiku.

[8] Např. tvrzení „hodnotou slotu *matka* pro objekt *Karel* je objekt *Eliška*“ lze vyjádřit jako „objekt Eliška je v roli *matka* vůči objektu *Karel*“. Zde ovšem snadno hrozí záměna (nejednoznačně pojmenovaného) slotu se slotem k němu inverzním.

[9] Lze chápat jako ternární (meta-)relaci mezi třídou, slotem a určitou hodnotou, např. „slot X nabývá pro třídu Y hodnoty ze třídy Z“, nebo „slot X nabývá pro třídu Y nejvýše 3 hodnot“.

[10] V následujícím příkladu omezení odpovídá *konceptu* „mít právě jednoho otce, a to takového, který je osobou“.

příkladem kombinovaného omezení na slot *má-otce* aplikovaného na třídu *osoba* může být, že hodnotou je opět instance třídy *osoba*, a to právě jedna[11].

Primitivní hodnoty a datové typy

Na začátku jsme si relaci vymezili jako popis vztahu mezi n-ticí objektů. Nebylo to ovšem přesné. Argumenty relací (resp. hodnotami slotů, na které se dále omezíme) mohou být i *primitivní hodnoty*, které žádnému objektu neodpovídají. V tom případě hovoříme o tzv. *dato-typových* ("datatype") *slotech*, na rozdíl od *objektových slotů*. Obor hodnot dato-typového slotu bývá vymezen některým základním datovým typem (string, integer, float...), číselným/alfanumerickým intervalem, nebo výčtem[12]. Někdy se primitivní hodnoty označují i jako *dato-typové instance*[13] (a samotné datové typy pak jako „dato-typové třídy"): např. číslo 4 pak bude instancí datového typu *integer*.

Pro objektově-orientovaného informatika může být překvapivé, že sloty (včetně dato-typových) mohou v principu nabývat více hodnot současně – to vyplývá z jejich „relační" podstaty. Sloty odpovídající jednoznačným vlastnostem objektů (jako např. dato-typový *má-hmotnost*, podobně jako objektový *má-otce*) se z toho důvodu explicitně deklarují jako funkční.

Axiomy, pravidla

Vedle výrazů explicitně vymezujících příslušnost ke třídám a relacím je obvykle možné do ontologií zařazovat další logické formule, vyjadřující např. ekvivalenci/ subsumpci tříd či relací, disjunknost tříd, rozklad třídy na podtřídy apod.. Nejčastěji se označují jako *axiomy*; ty mají buďto v ontologii zcela samostatné postavení (např. v OIL), nebo jsou syntakticky zařazeny jako součást definic tříd a relací (např. v Ontolingua nebo DAML+OIL), samy však vůči nim nemají definiční roli. V "pragmatických" jazycích těsněji spjatých s určitým odvozovacím mechanismem (např. OCML nebo OntoBroker) se spíše označují jako *pravidla* a mají i tomu odpovídající omezenější sémantiku.

Souhrnné informace

Vedle jednotlivých znalostních konstruktů obsahují (zejména formální) ontologie další, souhrnné údaje, umísťované nejčastěji do hlavičky. Patří sem zejména odkazy na *importované* ("imported", "included" apod.) ontologie, jejichž obsah je do stávající ontologie začleněn implicitně. Dále může jít o dokumentační položku, údaje týkající se autora, verze, času vytvoření, způsobu vytvoření (informace o použitém editoru) apod. Tyto údaje ovšem nemají význam pro logickou interpretaci ontologie.

---

[11] V principu bychom omezení kardinality pro daný slot mohli specifikovat i globálně, v praxi se však tato možnost nepoužívá. Cožpak si třeba můžeme být jisti, že ve vesmíru neexistují bytosti, které by měly více než jednoho otce…?

[12] To může mít smysl např. pro barvy – mohli bychom je sice definovat jako instance třídy *barva*, nebylo by to však příliš přirozené.

[13] Je tomu tak v návrhu nového ontologického jazyka OWL, viz část 3.6.

2.5 Další významné pojmy

V této části se alespoň stručně zmíníme o pojmech, které se tvorby a využívání ontologií bezprostředně týkají.

Ontologický závazek

Ontologický závazek ("ontological commitment") lze chápat jako rozhodnutí (závazek) určitého subjektu *používat* pro vyjádření pojmů prvky a strukturu dané ontologie. Označuje se tak ovšem také každé zásadní rozhodnutí provedené při samotné *konstrukci* ontologie, kdy se vybírá z různých variant konceptualizace. Uživatel tedy při volbě ontologie jistým způsobem akceptuje ontologické závazky v ní zahrnuté.

Problémy rozsahu a interakce

V průběhu vývoje ontologických aplikací byly identifikovány dva významné problémy, komplikující tvorbu a využívání ontologií.

První z nich je *problém rozsahu* (angl. "hugeness "). Jeho podstatou je skutečnost, že pojmů, které by se za určitých okolností mohly chápat jako relevantní zvolené doméně, je nezvládnutelné množství. Při tvorbě ontologie je proto nutné systematicky provádět selekci, a zařazovat zejména ty pojmy, které mají jasnou vazbu na pojmy z "jádra" ontologie. Dalším přirozeným opatřením je rozdělení ontologie do více nezávislých modulů.

Druhým je *problém interakce*. Optimální tvar ontologie je závislý na způsobu, jakým se bude používat, jinými slovy, mezi „statickými" doménovými znalostmi (na které bychom se v doménové ontologii v principu měli omezit) a postupy usuzování nad nimi existuje interakce. Někdy se také mluví o nutnosti kompromisu mezi *použitelností* a *znovupoužitelností* ontologie („usability–reusability tradeoff"): čím je ontologie nezávislejší na konkrétní aplikaci, tím méně efektivním se stává její přímé využívání.

3 Formální reprezentace ontologií

Během nepříliš dlouhé historie ontologického inženýrství bylo vyvinuto několik desítek více či méně používaných formálních jazyků. Zde si stručně představíme jen ty opravdu významné (obsáhlejší přehled je uveden v [29]). Podrobněji se v závěru budeme věnovat jazykům nejnovějším, úzce provázaným s vývojem okolo WWW.

3.1 Cyc

Jedním z prvních pokusů zachytit ve velkém rozsahu znalosti o světě je (v současnosti stále "živý") projekt *Cyc* [3], jehož název je odvozen ze slova "enCYClopedia". Projekt zahájený pod vedením D. Lenata již v r. 1984 usiluje o shromáždění *všeobecných znalostí* („common sense"), které by ve znalostních systémech fungovaly komplementárně ke znalostem expertním, a zabraňovaly absurdnímu

chování [14]. Svou podstatou i zaměřením aplikací jde o projekt, který jednoznačně patří do oblasti ontologií, ačkoliv jeho tvůrci tento termín příliš často nepoužívají[15]. Hovoří namísto toho o „mikroteoriích“ složených z dílčích tvrzení. Jako formální reprezentaci Cyc využívá svůj vlastní jazyk CycL (se základní notací převzatou z funkcionálního jazyka LISP), který má plnou vyjadřovací sílu predikátového kalkulu, kombinuje ho ovšem mj. s prvky rámcových jazyků.

Příkladem tvrzení z oblasti naivní fyziky, zapsaného v CycL, je

```
#$ist #$NaiveStateChangeMt
  (#$implies
      (#$and
          (#$isa ?FREEZE #$Freezing)
          (#$outputsCreated ?FREEZE ?OBJ))
      (#$stateOfMatter ?OBJ #$SolidStateOfMatter)))
```

Vyjadřuje následující: v mikroteorii *#$NaiveStateChangeMt* je pravdou (*#$ist* – „is true“), že objekt (označený proměnnou ?OBJ), který je výsledkem události (označené proměnnou ?FREEZE) typu „zmražení“, je v pevném skupenství.

Projekt Cyc se po relativně dlouhou dobu orientoval na „uzavřené“ komerční aplikace (nabízené prostřednictvím firmy CyCorp), a zveřejněna byla jen velmi malá část znalostí (tzv. „Upper Cyc Ontology“). V r. 2001 se však i tato skupina přiklonila k zásadě veřejné přístupnosti zdrojů, a v současnosti na adrese http://www.opencyc.org/ volně nabízí již okolo 6 000 konceptů a 60 000 tvrzení o nich.

3.2 Ontolingua

V jistém protikladu k “uzavřené” koncepci Cyc byla iniciativa vedená na začátku 90. let. T. Gruberem a jeho spolupracovníky ze stanfordské Knowledge System Laboratory (KSL)[16]. Jejich cílem bylo vyvinout dostatečně mocný a zároveň přehledný jazyk, který by umožňoval sdílet ontologie v rámci odborných komunit používajících vzájemně nekompatibilní znalostní systémy.

Jazyk, označený jako *Ontolingua* [51], je koncipován jako nadstavba jazyka *KIF* (Knowledge Interchange Format) [12], což je varianta predikátového kalkulu využívající (stejně jako CycL a mnohé další jazyky) syntaxe LISP. Základními konstrukty jazyka Ontolingua jsou definice tříd, relací a funkcí, přičemž vymezující podmínky pro příslušnost instancí jsou vyjádřeny právě v KIF.

Jako příklad definice *třídy* si uvedeme *Sale-Offer* („nabídka prodeje“) z Enterprise Ontology [5]. Třída je vymezena nutnou a postačující podmínkou (*iff-def*); tato podmínka je vyjádřena jako konjunkce dvou podvýrazů:

---

[14] Otřepaným příkladem takového chování je doporučení testu gravidity pro pacienta – muže, v lékařském expertním systému.

[15] Vztahu Cyc a ontologií se nově věnoval článek [67].

[16] Mimochodem, z téže laboratoře pocházel i tvůrce Cyc D. Lenat.

```
(Define-Class Sale-Offer (?X)
"A For-Sale situation with a Specified-Potential-Customer"
:Iff-Def
  (And
    (For-Sale ?X)
    (Exists (?Le) (Specified-Potential-Customer ?X ?Le))))
```

Jako příklad definice *relace* si uvedeme *excretes* (vztah „vyměšujícího“ a „vyměšovaného“) z lékařské ontologie On9 [16]:

```
(define-relation excretes (?a ?b)
"some body system or part embodies a function which has a
body substance as product."
:def
  (exists (?c) (and (embodies ?a ?c)(has-product ?c ?b)))
:constraints
  (or (anatomical-structure ?a)(organism ?a))
:axiom-def
  (arity excretes 2)
:axiom-constraints
  (range excretes body-substance))
```

V tomto případě je uvedeno několik vymezujících podmínek. První dvě mají (z čistě logického hlediska shodně) charakter nutné podmínky příslušnosti k relaci. Liší se od sebe tím, že podmínka *def* je definiční podmínkou vyjadřující podstatu relace, zatímco *constraints* je podmínkou z konceptuálního hlediska druhotnou. Poslední dvě podmínky (axiomy) tvoří obdobnou dvojici, v jejich zápisu ovšem nejsou uvedeny proměnné ze záhlaví – jde vlastně o meta-formule vyjadřující vlastnosti predikátu *excretes* jako takového.

Charakteristické vlastnosti různých tříd a relací (de facto relací 2. řádu), jako je vedle uvedených *arity* nebo *range* kupříkladu symetrie relace, rozklad třídy apod., se využijí ve velkém množství ontologií. Aby byla zajištěna jejich jednotná sémantika, vznikla tzv. *Frame-Ontology*. Jde o normální ontologii v jazyce Ontolingua, obsahující příslušné definice; po jejím začleněním do nové ontologie je možné na tyto vlastnosti odkázat.

Ontolingua byla od začátku koncipována jako „mezi-jazyk“, primárně určený k *výměně* ontologických informací mezi systémy, které interně používají vlastní (zpravidla omezenější avšak výpočtově efektivnější) reprezentaci; z toho vyplývají i omezené možnosti odvozování přímo v tomto jazyce. Na druhé straně ovšem Ontolingua, vzhledem ke své rozšířenosti, plnila po řadu let i roli jazyka „první volby“ pro tvorbu ontologií nezávisle na konkrétním znalostním systému.

Zásadním nástrojem pro zpřístupňování ontologií v jazyce Ontolingua se stal tzv. *Ontolingua server* [16], umístěný právě na KSL.

3.3 OCML

Omezené možnosti odvozování v jazyce Ontolingua motivovaly E. Mottu z Open University ve Velké Británii k návrhu jazyka, který by (při zachování “ontologické” podstaty) výrazněji podporoval přímý vývoj programových aplikací, aniž by bylo

nutno model překládat do jiného jazyka. Vývoj *Operational Conceptual Modelling Language* (OCML) [62] proto těsně propojil s tvorbou jeho interpretu, implementovaného v prostředí CommonLISP. Základem interpretu jsou algoritmy pro Prologovské dokazování a dědění v hierarchii tříd; třídy a jejich atributy jsou ovšem důsledně chápány jako unární resp. binární relace, takže primární (vnitřní) reprezentací jsou Hornovy klauzule.

"Deklarativní" část OCML je prakticky shodná s jazykem Ontolingua. Vedle toho je však podporována řada konstruktů převzatých z procedurálních jazyků a expertních systémů (podmínky, cykly, produkční pravidla), a navíc i pohodlné volání LISPovských funkcí. V OCML je proto možné napsat libovolnou aplikaci, která s ontologiemi nemusí ani mít nic společného.

Jako příklad si uvedeme relaci vyjadřující výpočet rozdílu dvou dat. Popis relace samotné v tomto případě neobsahuje její definici; ta je uvedena nepřímo, prostřednictvím pravidla se zpětným řetězením:

```
(def-relation date-difference
 (?date-1 ?date-2 ?diff ?diff1))

(def-rule date-difference
 ((date-difference ?date-1 ?date-2 ?diff ?diff1)
    if
  (and
   (date ?date-1 day ?d1 month ?m1 year ?y1)
   (date ?date-2 day ?d2 month ?m2 year ?y2)
   (not (== ?date-1 ?date-2))
   (= ?diff1
    (+
     (* (- ?y2 ?y1) 365)
     (* 30 (- ?m2 ?m1)) (- ?d2 d1)))
   (> ?diff1 ?diff))))
```

OCML si v komunitě znalostního inženýrství získal značný respekt, vyjádřený často používaným přídomkem „operacionální Ontolingua". Je ovšem skutečností, že se (zčásti asi i vinou vázanosti na LISP jako implementační prostředí) příliš nerozšířil mimo Open University (tj. místo vzniku) a některé její projektové partnery[17].

3.4 OKBC a XOL

Na základě analogie s ODBC začal již v r.1993 vznikat návrh aplikačního rozhraní, které by umožňovalo otevřenou komunikaci mezi rámcově–orientovanými znalostními systémy: *Open Knowledge Base Connectivity* (OKBC) [15]. Protokol specifikuje způsob předávání jak konstruktů znalostních bází (jako jsou třídy, individua nebo sloty), tak i volání operací nad těmito konstrukty. Soubor konstruktů podporovaných OKBC (tzv. znalostní model) vychází z analýzy řady existujících znalostních systémů, a sémanticky do značné míry odpovídá souboru konstruktů ontologických jazyků jako je Ontolingua.

[17] Autor tohoto textu, stejně jako několik dalších českých výzkumníků, měl možnost s OCML pracovat v rámci evropského projektu „Medical Guideline Technology" (1999-2001)

Právě proto se jednoduchá verze OKBC, nazvaná OKBC-Lite, v r.1999 stala výchozím bodem pro návrh dalšího ontologického jazyka: *eXtensible Ontology Language* (XOL) [38,55]. Jeho motivací byla potřeba *bioinformatické* komunity sdílet struktury znalostí o genovém výzkumu, které žádný ze stávajících jazyků plně nevyhovoval [60]. Podstatnou inovací oproti předcházejícím jazykům bylo zakotvení v syntaxi XML, která umožnila efektivní využití řady obecných nástrojů již existujicích pro tento značkovací jazyk. Na rozdíl od *XML Schematu*[18] [36] XOL zavádí pouze jedinou, generickou definici typu dokumentu (DTD), která umožňuje definovat strukturu tříd, jejich (pouze binární!) sloty, obory hodnot slotů apod. Jména tříd, atributů apod. tudíž neodpovídají názvům, nýbrž hodnotám prvků XML, např.:

```
<class>
  <name>osoba</name>
</class>

<slot>
  <name>věk</name>
  <domain>osoba</domain>
  <value-type>integer</value-type>
  <numeric-max>150</numeric-max>
</slot>
```

(tj., vlastností třídy *osoba* je *věk*, který nabývá číselné hodnoty nejvýše do 150).

3.5 Historické „webové“ ontologické jazyky: SHOE a Ontobroker

SHOE

Prvním jazykem vzniklým specificky pro účely přidání sémantiky k webovým stránkám byl v r. 1996 *SHOE* (Simple HTML Ontology Extension) [27], vyvinutý týmem J. Hendlera na University of Maryland. Jazyk SHOE umožňuje začleňovat do zdrojového kódu webových stránek jednak metadata o objektech, kterých se tyto stránky týkají, jednak samotné ontologie, definující sémantiku těchto metadat. Oproti jazykům typu Ontolingua je SHOE podstatně jednodušší – zachycuje pouze třídy a relace bez odlišení facet. Od pozdějších webových jazyků se liší také tím, že v něm je možné definovat relace s libovolnou aritou – pro každý argument se v definici uvede jeden subelement *def-arg*.

Jako ukázku z ontologie si můžeme uvést několik definic kategorií (tj. tříd) a relací, týkající se kateder informatiky („computer science departments“):

```
<DEF-CATEGORY NAME="Organization" ISA="base.SHOEEntity">
<DEF-CATEGORY NAME="Person" ISA="base.SHOEEntity">
<DEF-CATEGORY NAME="Publication" ISA="base.SHOEEntity">
```

[18] O XML Schematu se také v jisté době hovořilo jako o možném jazyku pro tvorbu ontologií. Brzy se ukázalo, že jeho prostředky jsou pro tento účel zcela nedostatečné. Významnou roli však v současnosti hraje jako doplněk ontologických jazyků řešící specifikaci datových typů, které tím mohou být o tuto problematiku „odlehčeny“. Viz část 3.6, podrobněji [57].

```
<DEF-CATEGORY NAME="ResearchGroup" ISA="Organization">
<DEF-RELATION NAME="member">
  <DEF-ARG POS="1" TYPE="Organization">
  <DEF-ARG POS="2" TYPE="Person">
  </DEF-RELATION>
<DEF-RELATION NAME="publicationAuthor">
  <DEF-ARG POS="1" TYPE="Publication">
  <DEF-ARG POS="2" TYPE="Person">
  </DEF-RELATION>
<DEF-RELATION NAME="publicationDate">
  <DEF-ARG POS="1" TYPE="Publication">
  <DEF-ARG POS="2" TYPE=".DATE">
```

Anotace samotné stránky je pak obsažena v elementu HTML nazvaném *instance*; instance je identifikována pomocí URL stránky:

```
<INSTANCE KEY="http://www.cs.umd.edu/users/george/">
```

Jeho subelementy jednak vymezují používanou ontologii

```
<USE-ONTOLOGY ID="cs-dept-ontology"
 URL="http://www.cs.umd.edu/projects/plus/SHOE/onts/cs.html"
 VERSION="1.0" PREFIX="cs">
```

jednak obsahují metadata o vlastníkovi stránky, vyjadřující například skutečnost, že je instancí třídy *GraduateStudent*, nebo že pro něj druhý[19] argument relace *age* nabývá hodnoty 52:

```
<CATEGORY NAME="cs.GraduateStudent">
<RELATION NAME="cs.age"> <ARG POS=TO VALUE="52">
```

Do „hlavní“ instance mohou být vnořeny další elementy *instance*, pro osoby, které nemají vlastní stránku a na příslušné stránce proto „hostují“. Z hlediska webového adresování jsou odlišeny pomocí návěští, např.:

```
<INSTANCE
KEY="http://www.cs.umd.edu/users/george/#BRUNHILDA">
```

SHOE rovněž umožňuje zápis inferenčních pravidel ve formě zjednodušených Hornových klauzulí 1. řádu.

To, že se v SHOE instance implicitně ztotožňují s fyzickými stránkami nebo jejich částmi, má nezvyklý efekt: webový objekt (stránka identifikovaná svým URL) do jisté míry splývá s objektem reálného světa (člověk, firma, výrobek...). S nadsázkou by se dalo říci, že „kdo nemá svou stránku, ten neexistuje“. Tento efekt byl v pozdějších webových jazycích eliminován zejména zobecněním pojmu „zdroje“ (v RDF, viz dodatek B), který už zdaleka nemusí mít povahu fyzické stránky HTML.

[19] Pořadí argumentu je „skryto“ za výrazem *to*. Vedle pořadových čísel je totiž první argument možné identifikovat i pomocí *from* a druhý pomocí *to*.

Ontobroker

Přibližně ve stejné době jako SHOE vznikala na naší straně oceánu (na univerzitě v Karlsruhe) koncepce projektu *Ontobroker* [48]. Základní myšlenka obohacování webových stránek o sémantické anotace se od SHOE neliší, rozdíl je však v navrhované architektuře, která je důsledně centralizovaná. Předpokládá existenci ontologického serveru, který spravuje ontologie a umožňuje jejich editaci pouze oprávněným (a kvalifikovaným) uživatelům. Také sbírání dat z anotovaných stránek není chápané jako "náhodné" prohledávání webu, nýbrž jako cílené návštěvy stránek *registrovaných* poskytovatelů informací, patřících k určité (např. profesní) komunitě.

V důsledku toho Ontobroker nepoužívá jednotný jazyk[20] pro ontologie a anotace: zatímco jazyk anotací je extrémně jednoduchým obohacením HTML o dodatečné atributy (tím se liší od SHOE, která zavádí vlastní sadu elementů), jazykem ontologií je propracovaný formalismus tzv. *F-logic*, jednoho z řady logických kalkulů vytvořených pro potřeby rámcových systémů (F v názvu symbolizuje "frame"). Uveďme si pro ilustraci příklad části ontologie zaměřené (obdobně jako výše uvedený příklad SHOE) na problematiku výzkumných komunit:

```
Person :: Object.
Researcher :: Person.
Person   [lastName =>> STRING;
         ...;
         publication =>> Publication].
FORALL Per,Pub
         Pub:Publication [author ->> Per]
         <-> Per:Person [publication ->> Pub]
```

První část fragmentu vyjadřuje hierarchii tříd; pro třídu *Person* jsou definovány sloty, a to jak dato-typové (*lastName*), tak objektové (*Publication*). Druhou částí je ekvivalenční axiom vyjadřující vztah dvou vzájemně inverzních slotů. Ontobroker na rozdíl od SHOE připouští pouze *binární* relace, ovšem toto omezení vyjadřovací síly kompenzuje zařazením *inferenčních pravidel*; mezi ně patří i uvedená ekvivalence.

Odpovídající znalostní anotace (zvýrazněná tučně) může pak být do kódu HTML vnořena takto:

```
<a onto="page:Researcher"></a>
<a onto="page[affiliation=href]"
   href="http://www.iiia.csic.es/">
   IIIA - Artificial Intelligence Institute </a>
<a onto="page[firstName=Body]">Richard</a>
```

Hodnoty *page*, *href* a *Body* zde hrají roli zástupných symbolů pro URL stránky (které podobně jako u SHOE splývá s dotyčnou „fyzickou" osobou), URL příslušného odkazu, a text obsažený v odkazu. URL stránky je tedy klasifikováno jako instance třídy *Researcher*, a jsou mu přiřazeny patřičné hodnoty slotů *affiliation* a *firstName*.

Stojí za zmínku, že zatímco pro webové použití SHOE a Ontobroker již (jako nekompatibilní s RDF, viz níže) zastaraly, instalace systému Ontobroker jsou (s řadou

[20] Zařazení Ontobrokeru mezi ontologické *jazyky* je proto samozřejmě nepřesné, jde o projekt, v jehož rámci bylo využito několik jazyků.

nově vyvinutých komponent) stále ještě poskytovány firmou OntoPrise [16] na komerční bázi pro účely firemních *intranetů*, kde jeho centralizovaná koncepce plně vyhovuje.

3.6 Nové „webové“ ontologické jazyky: RDFS, OIL, DAML+OIL a OWL

RDF jako atomická struktura

Na konci 90. let se stalo zřejmým, že podmínkou rozšíření sémantických metadat je jejich kompatibilita s metadatovým standardem vytvořeným konsorciem W3C, tj. s jazykem *Resource Description Framework* (RDF), viz dodatek B. Přirozeně se proto vynořila otázka, zda a jakým způsobem budou s RDF kompatibilní i ontologické jazyky, které budou metadatům onu požadovanou sémantiku zabezpečovat. Následovala intenzivní diskuse o vhodnosti datových modelů RDF versus “čistého” XML [74]. V ní se postupně prosadil názor, že grafový model RDF je na rozdíl od stromového, („gramatického“) modelu XML dostatečně otevřený pro vyjádření sémantiky metadat prostřednictvím ontologií – zejména tím, že jednotlivá tvrzení jsou v RDF nezávislá na ostatních, a přitom je možné je propojovat pomocí URI zdrojů. Vyjádření složitějších struktur pomocí RDF ovšem vyžaduje jejich „rozbití“ do trojic provázaných proměnnými, a tyto trojice jsou posléze serializovány pomocí XML. K problémům s tím spojeným se vrátíme v kapitole 5.1.

Zásadní příklon k RDF jako univerzální vyjadřovací struktuře se časově zhruba shoduje se vznikem iniciativy označované jako *sémantický web*[21]. I proto se někdy sémantický web (poněkud povrchově) definuje jako „síť sémantických metadat zapsaných pomocí RDF“, a problematika ontologických jazyků je od té doby z velké části podřazována právě sémantickému webu.

RDF Schema

První ze sémantických jazyků orientovaných na RDF vznikl již v r.1999, relativně nezávisle na hlavním proudu „ontologického“ výzkumu, přímo na půdě W3C. *RDF Schema* (RDFS) [25] představuje nadstavbu, která doplnila do struktury RDF hlavní konstrukce z rámcových (či objektových) systémů, tj. třídy a binární sloty s možností stanovit definiční obor a obor hodnot; nad třídami i sloty může být definována hierarchie. Zdroje z RDF lze pak snadno přiřazovat třídám z RDFS jako jejich instance, pomocí atributu *type*.

RDFS splňuje intuitivní požadavky webových návrhářů na možnost zachycení sémantiky obsahu stránek, zejména pokud jde definování. hierarchických struktur (např. témat). Oproti tradičním ontologickým jazykům mu však chybí možnost precizněji specifikovat podmínky příslušnosti ke třídám (lokální omezení), např. v disjunktivním tvaru, a zcela postrádá datové typy. Plná verze RDFS, zahrnující i možnost reifikace tvrzení (tedy zařazení konstruktů vyšších řádů), má na druhé straně nežádoucí vlastnosti z hlediska možností odvozování[22].

---

[21] Jedním z ústředních aktérů iniciativy je Tim Berners Lee, který stál před deseti lety u zrodu WWW. Obsáhlé informace o sémantickém webu lze nalézt na portálu http://semanticweb.org, stručným populárním výkladem je např. [41].

[22] Pan & Horrocks [65] kupříkladu dokázali, že reifikace může vést na Russellův paradox.

intersectionOf unionOf complementOf	Booleovský výraz nad třídami
oneOf	Třída jako množina primitivních hodnot
toClass hasClass	Třída splňující univerzální resp. existenční omezení na slot (tj. na třídy, které jsou jeho hodnotami)
hasValue	Třída splňující omezení na konkrétní hodnotu slotu (kombinace hasClass a oneOf)
minCardinalityQ maxCardinalityQ CardinalityQ	Třída splňující omezení na kardinalitu slotu

Tab. 1. Konstruktory tříd v DAML+OIL

subClassOf sameClassAs disjointWith	vztah obecnější a speciálnější třídy ekvivalence tříd prázdný průnik tříd
subPropertyOf samePropertyAs inverseOf transitiveProperty uniqueProperty unambiguousProperty	vztah obecnější a speciálnější vlastnosti (slotu) ekvivalence vlastností vztah vzájemně inverzních vlastností tranzitivita vlastnosti vlastnost je funkcí inverzní vlastnost k dané vlastnosti je funkcí
sameIndividualAs differentIndividualFrom	totožnost individuí odlišnost individuí

Tab. 2. Typy axiomů v DAML+OIL

DAML, OIL a deskripční logika

V polovině roku 2000 byl oficiálně zahájen projekt *DAML* (DARPA Agent Mark-up Language) [4], sponzorovaný vojenskou institucí DARPA, na kterém se od počátku podílela celá řada výzkumníků sémantického webu včetně Tima B. Lee. Cílem bylo vytvořit sémantický jazyk pro RDF s větší vyjadřovací silou než má RDFS. Jazyk označený jako DAML-ONT již vycházel z předchozího výzkumu v oblasti ontologických jazyků, a zahrnoval celou řadu konstruktů pro vymezení vztahů tříd a hodnot slotů.

Souběžně s objevováním významu RDF dospěla koncem 90. let jak evropská, tak postupně i americká větev výzkumu webových ontologií k potřebě postavit nové jazyky na některém propracovaném logickém kalkulu, který by umožňoval konstrukci složitějších podmínek při zachování výhodných vlastností pro výpočty. Volba padla

na *deskripční logiku* (viz dodatek C), která se stala základem[23] jazyka nazvaného *Ontology Inference Layer - OIL* [13] (ukázka OIL je v části 5.1). Jeho částečným sloučením s DAML-ONT (v pozoruhodně bezkonfliktní transatlantické spolupráci) vznikl jazyk s nepříliš elegantním názvem *DAML+OIL*, který je v době psaní tohoto textu stále ještě chápán jako "jazyk první volby" pro tvorbu webových ontologií.

Původnímu DAML-ONT se nebudeme věnovat, protože je považován za překonaný novějším DAML+OIL. Ten si představíme v následujícím odstavci; zmíníme přitom i některé rozdíly DAML+OIL oproti původnímu OIL, které jsou v poslední době znovu diskutovány.

DAML+OIL

Základem DAML+OIL [6] jsou *třídy* reprezentované buď svým *jménem* tj. URI (pojmenované třídy) nebo určitým *logickým výrazem* (anonymní třídy). Pro tvorbu logických výrazů vymezujících třídy se používají *konstruktory*[24] uvedené v Tab. 1. Konstruktory je možno libovolně skládat, a tak vytvářet složité výrazy. Vlastní náplň ontologie tvoří *axiomy*, vybudované právě nad výrazy reprezentujícími třídy. Typy axiomů jsou uvedeny v Tab. 2.

Základní představu o vzhledu ontologií v jazyce DAML+OIL lze získat z následujícího fragmentárního příkladu (jde o upravené ukázky z ontologie „vládní podpory vědy a výzkumu" [5]):

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY daml 'http://www.daml.org/2001/03/daml+oil#'>
 <!ENTITY old 'http://www.daml.org/xyz/20000911.daml#'>
 <!ENTITY xsd 'http://www.w3.org/2000/10/XMLSchema#'>
]>
<rdf:RDF xmlns:rdfs="&rdfs;"
         xmlns:rdf="&rdf;"
         xmlns:daml="&daml;">

<daml:Ontology rdf:about="">
  <daml:versionInfo>xyz</daml:versionInfo>
  <rdfs:comment>Ontology to describe organizations and
individuals participating in a government R&D
program.</rdfs:comment>
  </daml:Ontology>
```

[23] Někdy se i samotná deskripční logika zařazuje mezi ontologické jazyky. To však není zcela korektní, protože nejde o jazyk se strojově zpracovatelnou syntaxí, nehledě na rozmanitost kalkulů, které se za tímto zastřešujícím pojmem skrývají.

[24] Můžeme si povšimnout, že hlavní obohacení konstruktorů tříd DAML+OIL oproti konceptuálním výrazům v deskripční logice (viz dodatek C) představuje práce s *instancemi*.

```
<rdfs:Class rdf:ID="Agency">
        <rdfs:subClassOf rdf:resource="#Organization"/>
        <daml:sameClassAs rdf:resource="&old;Agency"/>
        <rdfs:subClassOf>
          <daml:Restriction>
            <daml:onProperty rdf:resource="#name"/>
            <daml:toClass
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
          </daml:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
          <daml:Restriction>
            <daml:onProperty rdf:resource="#partOf"/>
            <daml:toClass rdf:resource="#Agency"/>
          </daml:Restriction>
        </rdfs:subClassOf>
</rdfs:Class>
```

Po deklaraci jmenných prostorů následuje hlavička ontologie: element *daml:Ontology*. Prázdná hodnota jeho atributu *rdf:about* odkazuje na dokument, v němž je tato deklarace obsažena.

Poté uvádíme výraz vymezující třídu. Třída *Agency* je deklarována jako podtřída třídy *Organization*, a jako ekvivalentní třídě *Agency* ze starší verze téže ontologie. Hodnota (resp. všechny případné hodnoty) dato-typové vlastnosti *name* je pro třídu *Agency* deklarována pomocí lokálního omezení („restriction“) jako instance datového typu[25] *string* (tj. „jméno třídy je řetězec“). Obdobně, hodnota (resp. všechny případné hodnoty) objektové vlastnosti *partOf* je pro třídu *Agency* deklarována jako instance téže třídy *Agency* (tj. „agentura může být součástí jedině jiné agentury“).

DAML+OIL používá jednotný způsob, jak vyjádřit tvrzení o určité třídě: všechny axiomy jsou syntakticky začleněné do elementu odpovídajícího třídě. Pro srovnání, v původním OIL existují varianty dvě: omezení na slot zařazené jako *součást definice třídy*[26], a *samostatný axiom*. K této odlišnosti se ještě vrátíme v kapitole 5.2.

Vrstvená architektura

Na uvedeném příkladu si můžeme povšimnout, že elementy XML pocházejí z různých jmenných prostorů, označených prefixy *rdf:*, *rdfs:* a *daml:* (používaným pro DAML+OIL). Vztah RDF, RDFS a DAML+OIL (popřípadě OIL nebo OWL) můžeme chápat jako soustavu jazykových vrstev, ve které vyšší jazyk vždy dědí konstrukty z jazyka nižšího – ty pak ovšem není nutno znova definovat.

Předností takto vrstvené architektury ontologií je částečná zpětná kompatibilita. Zejména jde o situaci, kdy by ontologie měla být využita (pro doplnění sémantiky k metadatům) webovým agentem schopným pracovat pouze s RDFS. Ten bude schopen nejen provést základní syntaktickou analýzu kódu (vyjádřeného v korektním RDF), ale také identifikovat explicitní hierarchii tříd (popsanou pomocí konstruktů RDFS);

---

[25] Hodnoty dato-typových slotů mohou být specifikovány pomocí XML Schématu. Sféry tříd (objektů) a datových typů jsou tím od sebe v DAML+OIL důsledně odděleny.

[26] Zařazení této varianty souvisí se snahou návrhářů OIL zachovat návaznost na velmi rozšířené systémy založené na rámcích.

ztratí se pouze „sofistikovaná“ informace obsažená v logických výrazech DAML+OIL (v lokálních omezeních či axiomech).

OWL – Web Ontology Language

Na základě zhruba dvouletých zkušeností s využíváním DAML+OIL vzniká v současnosti pod hlavičkou *W3C Ontology Working Group* [34] nový, v některých rysech odlišný ontologický jazyk, nazvaný *OWL*. Pracovní verze definice jazyka jsou postupně zveřejňovány od května 2002, a začínají se postupně ustalovat.

Z praktického hlediska bude zřejmě významnou novinkou vyčlenění “minimální množiny” konstruktů jazyka, označené jako *OWL Lite* – to by mělo usnadnit implementaci programových nástrojů, která byla pro plnou verzi DAML+OIL (i pro plnou verzi OWL) velmi komplikovaná. OWL Lite podporuje jen některé typy lokálních omezení na sloty, zejména jsou oslabeny možnosti využívání anonymních tříd a omezení na kardinalitu, u kterých byla v případě DAML+OIL empiricky zjištěna malá míra využívání.

3.7 Specifické alternativy

Vedle hlavního proudu ontologického inženýrství existuje několik dalších velmi rozšířených jazyků pro konceptuální modelování, které mají k ontologiím blízko, ačkoliv tento termín nepoužívají. Zde se zmíníme o dvou, které mají výsadní postavení v oblasti databázových resp. textových informačních systémů: UML a TopicMaps.

UML a ontologie

UML [30] je v současnosti základním modelovacím nástrojem používaným při vývoji informačních systémů. Při prvním setkání s problematikou ontologií proto často databázoví inženýři kladou otázku: Nejsou nové, specializované jazyky luxusem? Přinesou něco nového oproti UML?

Primárním rozdílem ontologií oproti UML je, že jejich třídy v sobě nezahrnují procedurální *metody* jako vyjádření zodpovědnosti za určitou funkci zkoumaného nebo navrhovaného systému. Důraz je naopak položen na logické vztahy mezi třídami, které mohou mít komplikovanější sémantiku, než je možno v UML graficky zachytit pomocí asociací. To souvisí i se skutečností, že UML je primárně určen k návrhu softwarových systémů, zatímco ontologie (zejména doménové) slouží k vyjádření konceptuálních vztahů relativně nezávisle na programových aplikacích.

Dalším rozdílem je postavení *instancí*. Zatímco v UML je každá instance napevno spojena s třídou, do které náleží, v ontologiích instance (individua) odpovídají specifickým objektům reálného světa, jsou „občany první kategorie“, a příslušnost ke třídě (často i více třídám současně!) je pouze jejich vlastností, která se může v čase případně i měnit.

Uvedené rozdíly samozřejmě neznamenají, že by modely reprezentované v UML nemohly být (zejména v podnikové praxi) velmi dobrým východiskem pro vznik ontologií, případně, že by nemohly samy o sobě posloužit pro některý z účelů, pro které se běžně používají semi-formální ontologie

Topic Maps a ontologie

Dalším konceptuálně odlišným standardem[27], který aspiruje na roli ontologického jazyka, jsou *Topic Maps* [37]. Základním konstruktem je u nich *téma* („topic"), které zastupuje[28] určitý *předmět* („subject", vlastně téma v obecném smyslu). Tématům jsou přiřazovány konkrétní informační zdroje, které se označují jako *výskyty* („occurrences"). Témata mohou být propojena *asociacemi*. K odlišení témat se stejným názvem a odlišným obsahem slouží tzv. *rozsah působnosti* („scope"). Strukturu *tříd* a *instancí* je v Topic Maps možné vybudovat prostřednictvím příslušných asociací (nadtřída-podtřída, třída-instance); jak třídy tak instance jsou ovšem tématy, a nakládání s nimi se nijak neliší.

Pro funkci „univerzálního" ontologického jazyka Topic Maps postrádají vyjadřovací sílu, které mají formálně-logické jazyky. Neumožňují tudíž složitější inference, a odvozování nad nimi se víceméně omezuje na *slučování* témat. Pro běžné použití při sémantickém indexování zdrojů ovšem mohou být přitažlivé jako snadno zvládnutelná alternativa k „hlavním" jazykům sémantického webu.

4 Tvorba a využívání ontologií

V první části této kapitoly si letmo představíme nově vzniklé typy software, které jsou s tvorbou a využíváním ontologií spojeny, a to jednak "klasické" editory[29], tak i příklad komplexního řešení, které se zdá být vhodné pro začlenění do databázově orientovaných podnikových informačních systémů. Ve druhé části se zaměříme na aplikační oblasti a na projekty dotažené do praxe.

4.1 Programové nástroje

Editory ontologií

Přestože lze prakticky všechny ontologické jazyky zpracovávat běžným textovým editorem (a ty novější také editorem XML), jejich rutinní používání je obtížně představitelné bez specializovaného editoru. V části 3.2 jsme se zmínili o *Ontolingua serveru*, jehož součástí je pochopitelně editor; webové editační rozhraní pro OCML zase zprostředkovává systém *WebOnto* [46], a pro ontologii *Sensus* [26] editor *Ontosaurus*. Další, velice propracovaný systém, který není bezprostředně spojen s žádným jazykem, avšak je z něj možné exportovat do všech hlavních formátů, je *Protégé* [22] vyvinutý týmem M. Musena v institutu Stanford Medical Informatics. Komerčním produktem s řadou nadstandardních funkcí (např. podpora slučování ontologií, jejich kolaborativní editování apod.) je *OntoEdit* [72]. Na závěr zmiňme systém *OilEd* [14], určený specificky pro webové jazyky odvozené od OIL). Ten

[27] Jde dokonce o standard ISO 13250 XTM.

[28] Používá se zde termín „reifikuje", ovšem v poněkud odlišném smyslu než v logice.

[29] Vedle editorů jako základního nástroje pro práci s ontologiemi existuje i řada nástrojů specializovaných např. na integraci ontologií, učení ontologií z textu, nebo anotování textů na základě ontologií. Ty jsou ovšem zpravidla těsněji spjaté s konkrétním výzkumným projektem. Rozsáhlý přehled všech typů nástrojů je např. v [29].

kompenzuje relativně omezený okruh editačních funkcí jednoduchostí instalace a obsluhy, a zdá se být vhodným zejména pro výukové a demonstrační účely; sami tvůrci ho označují za „poznámkový blok pro ontologie“. Jeho nezanedbatelnou předností je navíc spojení s odvozovacím mechanismem *FaCT* [54].

KAON – ontologická infrastruktura pro podnikové informační systémy

Příkladem zajímavého přístupu, který přibližuje ontologie realitě databázových informačních systémů, je *KAON* – KArlsruhe ONtology infrastructure [11]. Jde o soubor nástrojů pro tvorbu a využívání ontologií, vyvinutý na Universitě v Karlsruhe a zpřístupňovaný prostřednictvím licence typu Open Source. Jeho ústředním prvkem je tzv. KAON API [61] – aplikační rozhraní zpřístupňující primární datovou vrstvu ontologie, využívající relačně-databázové technologii. Významnou vlastností rozhraní je jeho variabilita – existuje v:

- Implementaci pro *ontologické inženýrství*, soustředěné na operace nad koncepty (třídami) – přidávání, odstraňování, úpravy apod. – včetně jejich transakčního zpracování.
- Implementaci pro přístup k databázím *metadat RDF*.
- Implementaci pro přístup ke klasickým *relačním databázím*, s využitím mapování mezi strukturou ontologie a schématy RDBMS (obdoba relační implementace objektového databázového modelu). Existující relační databáze se tak mohou stát přirozeným základem ontologií.

Tomu lze uzpůsobit i organizaci primárních dat. V implementaci pro ontologické inženýrství mohou být například hodnoty slotů ukládány přímo do atributů tabulky odpovídající konceptu (třídě), což usnadňuje editaci konceptů. V implementacích pro přístup k většímu objemu instancí (se strukturou konceptů již ustálenou) je naopak výhodnější mít hodnoty slotů v oddělených tabulkách.

Nad KAON API pak může běžet řada uživatelských aplikací: pro tvorbu ontologií (editor SOEP), generování webových portálů na základě ontologií, anotování webových stránek metadaty (nástroj CREAM), extrakci metadat z relačních databází (nástroj REVERSE), učení ontologií z textu (Text-To-Onto) apod.

4.2 Praktické aplikace

Za hlavní oblasti využití ontologií se v současnosti označují např.:

- *Znalostní management* ve firmách. Pro efektivní fungování organizace je třeba, aby se informace a znalosti (jak interního tak externího původu) neztrácely, a včas se dostávaly k těm pracovníkům, kteří je mohou využít. S pomocí ontologie je možné zachytit věcnou podstatu znalostí, a tím jednak zabezpečit jejich konzistenci, jednak usnadnit jejich vyhledání.
- *Elektronické obchodování* typu B-to-C i B-to-B. V prvním případě může ontologie usnadnit vyhledání požadovaného produktu zákazníkem, ve druhém se jedná o rychlé vyhledání potenciálního partnera, ale perspektivně také o automatizaci procesu sjednání obchodních podmínek.
- *Zpracování přirozeného jazyka* – terminologické ontologie mohou napomáhat např. při překladu nebo automatické sumarizaci textů.
- *Inteligentní integrace informací*. Ontologie může sloužit jako zastřešení datových schémat distribuovaných zdrojů (strukturovaných nebo semi-

strukturovaných databází, případně „tabulárních“ webových stránek) na vysoké úrovni abstrakce.
- Pojmové *vyhledávání informací* jako vylepšení stávajících internetových vyhledávačů.
- Sémantické *webové portály* konstruované polo-automaticky na základě metadat od poskytovatelů informace.
- Inteligentní *výukové systémy*.

Skutečností ovšem je, že ambiciózní myšlenky akademického výzkumu značně předbíhají praxi, která k potřebě využívání ontologií (a sémantických metadat) dospívá jen pozvolna. Vzhledem k překotnému vývoji problematiky také nepřekvapí, že již realizované praktické aplikace často odrážejí starší stupeň vývoje ontologických jazyků, a postrádají přímou vazbu na aktuální vývoj sémantického webu. Terminologické ontologie jsou zastoupeny mnohem četněji než ontologie konceptuální (které jsou hlavním předmětem tohoto textu). Z mála případů uplatnění netriviálních ontologií ve firemní praxi si uvedeme alespoň dva; některé další jsou uvedeny v přehledové studii [29] zpracované v rámci projektu OntoWeb [18].

CoMMA – budování paměti organizace

Projekt 5. rámcového programu EU *CoMMA* (“Corporate Memory Management through Agents”, 2000-2002) [2] se zaměřil na zachycení tzv. *paměti organizace* („corporate memory“): informační struktury, umožňující dlouhodobě uchovávat a sdílet znalosti a informace pracovníky organizace. Projekt specificky řešil scénáře zapojení nového pracovníka a monitorování nových technologií. Na projektu koordinovaném francouzským výzkumným střediskem INRIA se podílela např. konzultační firma ATOS, stavebně projektantská organizace CSTB a Deutsche Telekom.

Soubor znalostí získaných z různých zdrojů a různým způsobem (rozhovory s pracovníky, pozorování pracovního prostředí, rozbor firemních směrnic apod.) je nutně nekonzistentní a nepřehledný: jeho strukturování je úkolem ontologického inženýrství. Ontologie vzniká spojením dvou přístupů:
- *zdola* – zobecněním termínů z dokumentů a zápisů z rozhovorů/pozorování
- *shora* – znovupoužitím částí existujících ontologií jako je Enterprise [8], a neformalizovaných zdrojů (knihy apod.).

Vlastní paměť organizace má charakter databáze dokumentů opatřených metadaty RDF, se sémantikou definovanou pomocí příslušné ontologie.

ALICE – navigace po elektronickém obchodě

Projekt *ALICE* [1], koordinovaný Knowledge Media Institute na Open University, Velká Británie, se zaměřil na problematiku online nakupování. Ontologie (v jazyce OCML, viz část 3.3) zde funguje jako nástroj pro usnadnění navigace uživatele. Zahrnuje koncepty vyjadřující vlastnosti produktů, zákazníků, firem, a také samotných elektronických nákupních seancí. Výsledky projektu byly údajně komercializovány ve formě produktu islandskou internetovou firmou INNN.

5 Ontologie a sémantický web – otevřené problémy

V této kapitole uvedeme řadu problémů, které v současnosti příslušnou odbornou komunitu významně zaměstnávají, a často i názorově rozdělují. Budeme se věnovat výhradně problémům spojeným s ontologiemi, popřípadě s reprezentací metadat. Řada jiných problémů spojených s rozšiřováním sémantického webu do praxe (zvl. tzv. „business“ aspekty) je diskutována např. v [22].

5.1 Základem RDF nebo XML?

Přestože se role RDF jako základu sémantického webu zdá být stále neotřesitelná, jeho „maximalistické“ využívání pro reprezentaci ontologií je spojeno s řadou problémů. Zmínili jsme se o tom, že složitější konstrukty ontologií (logické výrazy) musí být rozloženy do několika trojic RDF, přičemž každá trojice je dále serializována do elementů XML. Bráno z opačné strany, nejen že elementy XML definující trojice RDF již nemají sémantiku odpovídající modelu XML (jde o směsku elementů z různých jmenných prostorů, s ohodnocenými atributy, avšak bez vlastního datového obsahu), ale i samotné trojice RDF již sémanticky neodpovídají modelu RDF. Nejsou na sobě vzájemně nezávislé, namísto regulérních zdrojů obsahují na mnoha místech proměnné, a smysl dávají tudíž jen v kontextu – hovoří se proto někdy o „temných“ trojicích („dark triples“).

Pokud chceme např. v jazyce OIL (v DAML+OIL je situace obdobná) definovat „malou firmu“ jako firmu s ročním obratem do 1000000, musíme použít tři syntaktické trojice:

```
<oil:DefinedClass
  rdf:about="http://x.y.com/onto.rdfs#small_company">
    <oil:hasPropertyRestriction rdf:resource="_anon1"/>
    <rdfs:subClassOf
        rdf:resource="http://x.y.com/onto.rdfs#company"/>
</oil:DefinedClass>

<oil:HasValue rdf:about="_anon1">
  <oil:toClass rdf:resource="_anon2"/>
  <oil:onProperty
    rdf:resource="http://x.y.com/onto.rdfs#ann-turnover"/>
</oil:HasValue>

<oil:Max rdf:about="_anon2"
  oil:integerValue="1000000"/>
```

První přiřazuje třídě *small_company* dvě vlastnosti: podřazenost třídě *company* a platnost zbytku omezení identifikovaného pomocí proměnné *anon1*. Druhá je vlastně serializací instance ternární relace („hodnotou vlastnosti *ann_turnover* pro třídu *_anon1* musí být třída *_anon2*“). Třetí pak už jen specifikuje (dato-typovou) anonymní třídu *_anon2* jako celočíselnou hodnotu, která nesmí být větší než 1000000.

Někteří výzkumníci sémantického webu v čele s P. Patel-Schneiderem (Bell Labs Research, USA) z těchto důvodů dokonce znovu otevírají již zdánlivě uzavřenou otázku: RDF nebo XML? Patel-Schneider [64] upozorňuje, že v XML (nebo v jemu

blízkých jazycích jako je HTML) je v současnosti zaznamenáno daleko více relevantních dat, než v RDF. Řešením tedy není XML pro jeho omezenou vyjadřovací sílu zavrhnout, nýbrž doplnit ho o vlastnosti potřebné pro sémantický web. V překládaném návrhu již RDF není základem sémantického webu, nýbrž jen zabezpečuje identifikátory (URI) zdrojů, přes které jsou propojovány datové uzly XML. Možnost vyjádřit v ontologiích např. disjunktivní informaci je teoreticky podepřena tím, že analyzovanému dokumentu namísto jediného datového modelu může odpovídat kolekce interpretací.

5.2 Role automatické inference

Webové ontologie byly od počátku chápány nikoliv jako pasivní soubory platných vztahů, ale jako znalostní báze, nad kterými lze strojově odvozovat. Pro jazyk OIL tak kupříkladu od samého počátku existoval odvozovací nástroj FaCT [54], který umožňuje ověřovat konzistenci teorií a dovozovat implicitní vztahy subsumpce mezi třídami. Nároky na efektivní inferenci ovšem mohou být při standardizaci jazyka někdy v rozporu s nároky na snadný vývoj ontologií z pohledu uživatele. Příkladem může být již rozdíl mezi OIL a DAML+OIL, zmíněný v části 3.6. Dvojí způsob reprezentace tvrzení v OIL (omezení slotů uvedená v definici třídy, a samostatné axiomy) je z hlediska *inference* zcela ekvivalentní a zbytečně zvyšuje nároky na syntaktickou analýzu kódu. Pokud má ovšem s ontologií pracovat jiný člověk, než ten, který ji vytvořil, může být rozlišení „definičních vlastností“ třídy od ostatních tvrzení, která se o ní zmiňují, významné pro pochopení hlubšího smyslu.

Jinou otázkou je, zda je skutečně nutné umožnit nad ontologiemi neomezené odvozování v některém *obecném kalkulu* jako je deskripční logika nebo třeba Hornova logika 1. řádu. Architektura *KAON* [61] v tomto směru naopak sází na *předdefinované typy pravidel* (sémantické vzory) [71], které se zdají pro rozsáhlý okruh aplikací postačovat, a výrazně zvyšují efektivitu zpracování.

5.3 Tradiční reprezentace znalostí vs. webový pragmatismus

Ontologické jazyky pro WWW se svým pragmatismem a snahou o jednoduchost (byť z pohledu firemní praxe možná ještě nedostatečnou!) stále více vymykají z tradičního pojetí formálně-logické reprezentace znalostí v umělé inteligenci. Dokonce i často zdůrazňované zakotvení webových ontologií v deskripční logice je v poslední době zpochybňováno[30].

To přirozeně vyvolává v etablované komunitě reprezentace znalostí nedůvěru, a novým jazykům je soustavně vytýkána nízká *vyjadřovací síla*. Protiargumentem ovšem je, že bez jednoduchosti není naděje na širší používání. Jako precedens může sloužit porovnání s historickým vývojem značkovacích jazyků: zatímco „silný“ SGML zůstal omezen na malou komunitu uživatelů, „zjednodušený“ XML (který věcně v první fázi nepřinesl téměř nic nového) zaznamenal raketový nástup.

Dalším aspektem WWW, který bývá v souvislosti s ontologiemi často podrobován kritice, je jeho “nevázanost”. Tradiční směr reprezentace znalostí zde někdy stojí na

[30] Ukazuje se například, že inferenční mechanismy jako je FaCT se používají nanejvýše pro testování konzistence modelu, ale téměř nikdy pro automatickou konstrukci hierarchie konceptů, která měla být jeho hlavní předností.

téže straně fronty s filosofickým pohledem, který staví do protikladu vznik ontologie jako výsledek *sdílení* soukromých konceptualizací („Jak bude vypadat lékařská ontologie jako sdílená konceptualizace komunity stoupenců kultu woodoo?“ [21]) a “čisté”, dokazatelné *pravdy*. Větší vyjadřovací síla jazyků může skutečně zvyšovat míru axiomatizace teorií, a tím snad i objektivitu ontologií (ve srovnání s ad-hoc hierarchiemi témat, ke kterým web inklinuje). Otázkou je, jak velký podíl mezi obory, jejichž konceptualizace má smysl prostřednictvím webu využívat, tvoří exaktní, matematizovatelné disciplíny. Kupříkladu už u medicíny (která je stále ještě bezprostředně spjata s fyzickou realitou) by byl matematický způsob dosahování objektivity v řadě případů kontroverzní, o většině společenských věd ani nemluvě.

Přestože, jak se zdá, „pragmatický“ webový pohled na ontologie stále více převládá, „principiální“ směr ontologického inženýrství se rovněž vyvíjí dál, a to zejména ve směru budování vysoce abstraktních, generických ontologií, jako je *SUMO* [33] nebo ontologie vyvíjené v novém jazyce *GOL* [10]. Dlouhodobě zřejmě nadále půjde o zajímavou oblast akademického výzkumu, jejíž některé prvky budou v omezené míře pronikat i do webových ontologií.

5.4 Revoluční vs. evoluční nástup sémantického webu

Vedle problémů bezprostředně souvisejících s ontologickými jazyky se zmíníme i o obecnějším, a možná i palčivějším problému sémantického webu, kterým je „propast“ dělící ho od webu stávajícího.

WWW vděčí za své rozšíření do značné míry jednoduchosti a průhlednosti jazyka HTML, která umožňovala i naprostému začátečníku vytvořit během několika minut (formou znovupoužití existujícího zdrojového kódu) funkční webovou stránku. Sémantický web oproti tomu předpokládá u autorů stránek systematické vkládání metadat RDF, a tudíž alespoň základní znalost konstruktů tohoto ne zcela triviálního jazyka. Z řady možností, jak sobě oba „světy“ přiblížit, se zmíníme o dvou.

Etzioni [47] doporučuje netrvat pro první fázi na sémantické interoperabilitě (globálně závazných ontologiích). Tím se uvolní prostor pro *lokální* (tj. proprietární) sémantické značkování, realizované pomocí elementů XML se „samovysvětlujícími“ názvy, vnořenými do běžného HTML. To povede nejprve ke vzniku „místních sémantických webů“ univerzit či firem; teprve dodatečně se na sebe příslušné lokální ontologie budou *mapovat*, a tím dojde k propojení do většího celku. Hezký příklad takového mapování uvádí např. Rousset [68]: pokud hledáme informace o místě, kde bychom mohli povečeřet a přitom shlédnout hudební představení, budou nás zajímat jak instance třídy *restaurace* se slotem *kategorie* naplněným hodnotou *s_hudebním_programem*, tak i instance třídy *kabaret* se slotem *nabídka_stravování* naplněným hodnotou *večeře*. Mapování zajistí vazbu mezi oběma terminologicky i strukturně odlišnými koncepty.

Obdobně i Haustein [52] usuzuje na nutnost udržování informací ve formátu HTML. Prosté přidávání metadat RDF by pak vedlo k duplicitě informací a riziku nekonzistence. Jako částečné řešení proto navrhuje tvorbu informačního obsahu stránek v XML, s využitím jednoduchých ontologií a šablon (za intenzivní softwarové podpory). Podstatné je, že viditelným výstupem celého procesu jsou stránky HTML (s vysokou mírou sjednocení stylu); vedle toho automaticky vznikají i metadata RDF, ovšem jen jako vedlejší produkt, který uživatele nestojí žádné úsilí.

5.5 Rozsáhlé, heterogenní a nekonzistentní zdroje

Zatímco v samotných počátcích sémantického webu se pozornost soustředila téměř výhradně na standardizaci *jazykové reprezentace*, v posledních dvou letech vystupuje stále více do popředí otázka adaptace existujících, rozsáhlých zdrojů znalostí jako *obsahu* ("content") sémantických metadat resp. základu doménových ontologií. Lze dokonce říci, že „šarvátky" okolo formálních vlastností reprezentačních jazyků, které jsou předmětem velké části odborné literatury z oblasti ontologií a sémantického webu, blednou před velikostí problémů spojených s adaptací existujících standardů i dalších zdrojů. Bez ní ovšem nelze doufat v rozšíření sémantického webu do každodenního života běžných uživatelů internetu.

Zásadní roli hrají existující standardy terminologických a klasifikačních znalostí různých oborů. Příklady zdrojů, které vznikly na ontologickém inženýrství nezávisle avšak mohou pro standardizaci obsahu sémantického webu sehrát významnou roli, jsou knihovnická klasifikační schémata jako *Dublin Core* [7], produktové katalogy jako *UNSPSC* [32], nebo lékařské tezaury, mezi nimiž hraje prvořadou roli *UMLS* [31]. Předběžná studie kandidátů na „content standards" pro různé obory lidské činnosti [28] zahrnuje okolo 60 zdrojů, které se liší rozsahem, mírou formalizace i základními reprezentačními paradigmaty. Nehledě na vzájemnou odlišnost navíc podrobná analýza zdrojů odhaluje řadu vnitřních nekonzistencí. Můžeme jmenovat např. situaci zmiňovanou v [66]: metatezaurus UMLS používá termín „salmonella" jak pro označení bakteriálního kmene, tak i choroby jím způsobené, což vede k neadekvátní vícenásobné dědičnosti.

Vedle zdrojů, které již mají ve svém oboru autoritu oficiálního či neoficiálního standardu, existují na WWW tisíce zdrojů „nezaručených", které přesto mohou být pro vznik ontologií přínosné. Jedná se jak o primární webové stránky, tak o různé slovníky, souhrny a adresáře (včetně katalogů typu Open Directory, viz [56]), obsahující velké množství textového materiálu. Automatické využívání takových zdrojů pro konstrukci ontologií se označuje jako *učení ontologií* („ontology learning") [59], v posledních dvou letech se hovoří i o *dolování sémantického webu* ("semantic web mining") [39]. V jeho rámci byly navrženy techniky pro identifikaci konceptů (tříd) v textu, sběr instancí, a nalézání taxonomických i netaxonomických relací. Vzhledem ke způsobu vzniku mají ovšem vzniklé ontologie spíše terminologický než formálně-logický charakter, zachycují povrchové vazby, a jejich přetvoření do „skutečné" ontologie se bez lidského zásahu samozřejmě neobejde.

6 Závěr

V tomto doprovodném textu k tutoriálu se autor pokusil shrnout na malém prostoru jak historii ontologického inženýrství, tak i jeho aktuální problémy, s důrazem na vztah k webovým zdrojům. Vážnějším zájemcům o problematiku lze doporučit literaturu uvedenou v bibliografii a odkazy uvedené v dodatku A.

Autor si je vědom toho, že vzhledem k rychlému vývoji oboru budou některé informace uvedené v textu zastaralé dokonce již v okamžiku publikování, doufá však, že zpracovaný materiál bude přesto čtenáři přínosný pro úvodní orientaci v problematice a pochopení kontextu vzniku dalších ontologických jazyků, projektů a standardů, které spatří světlo světa v příštích letech.

7 Dodatek A – Doplňkové odkazy

V této části uvedeme několik odkazů, které mají zásadní význam pro celou oblast, nebo je nelze adekvátně zařadit do seznamu literatury.
Zastřešující projekty:

- Iniciativa *Semantic Web*, http://semanticweb.org
- Evropský síťový projekt *OntoWeb* (Ontology-based information exchange for knowledge management and electronic commerce), http://www.ontoweb.org/
- W3C Ontology Language Working Group
- Projekt DAML http://www.daml.org/ (zahrnuje mj. rozsáhlou kolekci ontologií)

Hlavní konference věnované z podstatné části problematice ontologií (z pohledu znalostního inženýrství) a sémantického webu:

- *K-CAP – International Conference on Knowledge Capture* (dříve KAW – Knowledge Acquisition Workshop), http://sern.ucalgary.ca/ksi/K-CAP
- *EKAW – International Conference on Knowledge Engineering and Knowledge Management* (původně European Knowledge Acquisition Workshop), http://babage.dia.fi.upm.es/ekaw02/ekaw02.htm
- *ISWC – International Semantic Web Conference*, http://iswc.semanticweb.org
- V tuzemsku se problematika ontologií objevuje zejména v programu nově vzniklé série konferencí *Znalosti*, viz http://www.cs.vsb.cz/znalosti.

8 Dodatek B – Základní informace o RDF

Resource Description Framework (RDF) [24] je doporučením W3C z r. 1999 pro reprezentaci struktury webových metadat. Datový model RDF je založen na lingvisticky inspirované konstrukci složené ze tří prvků: *subjektu*, *predikátu* a *objektu*, která se dohromady označuje jako *tvrzení* ("statement"). Univerzálním prvkem RDF je *zdroj* ("resource"), identifikovaný svým URI[31]. Zdrojem může být nejen webová stránka, ale např. celé webové sídlo nebo i dokument umístěný mimo web. Zdroj může vystupovat jak v roli subjektu, tak v roli objektu. Predikáty odpovídají sledovaným vlastnostem subjektů, a objekty hodnotám těchto vlastností. Objektem může být kromě zdroje také *literál*, tj. primitivní datová hodnota.

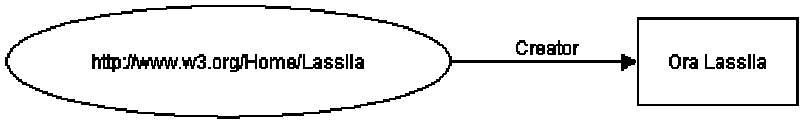
Datový model RDF sám o sobě nepředjímá konkrétní reprezentaci – ta může být např. *grafová* (subjekty a objekty jako uzly, predikáty jako orientované hrany) nebo vyjadřovat tvrzení prostě jako posloupnost tří výrazů na téže řádce[32]. Základní syntaxe doporučovaná W3C je ovšem založena na XML[33]. Hovoří se o tzv. *serializaci:*

---

[31] *Universal Resource Identifier*, zobecnění běžně používaných URL (Universal Resource Locator).

[32] Tzv. notace *N3* [40], určená „pro začátečníky".

[33] Výklad jazyka XML (eXtensible Mark-up Language) do tohoto textu nezařazujeme, neboť předpokládáme, že většina čtenářů je s ním (na rozdíl od dosud podstatně méně rozšířeného



Obr. 1. Příklad grafové interpretace RDF

jednotlivé prvky tvrzení RDF jsou specifickým způsobem řazeny za sebe do elementů a atributů XML. Ukažme si jako příklad základní variantu zápisu tvrzení odpovídající Obr. 1: „Tvůrcem (= predikát) zdroje *http://www.w3.org/Home/Lassila* (= subjekt) je *Ora Lassila* (= objekt)“:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

Obálkový element *RDF* obsahuje atributy definující jmenné prostory: *rdf* pro konstrukty samotného RDF, a *dc* pro metadatový standard Dublin Core [7], z něhož je převzata sledovaná vlastnost „být tvůrcem“. Vlastní tvrzení je pak obsaženo v elementu *Description*, jehož atribut *about* odkazuje na subjekt, název subelementu odpovídá predikátu, a obsah subelementu objektu.

Specifikace RDF obsahuje řadu možností, jak vyjádřit složitá tvrzení úsporněji. Je zjevné, že pokud chceme témuž subjektu přiřadit různé vlastnosti, můžeme to učinit v rámci jediného elementu *Description* s více subelementy. Pokud chceme naopak přiřadit tutéž vlastnost více zdrojům, lze je sdružit do tzv. *kolekce*. Vedle základní notace také existuje notace zkratková, která hodnoty metadat umísťuje do atributů namísto do subelementů.

Velmi silným prostředkem je v RDF tzv. *reifikace*[34]. Její podstatou je možnost chápat celé tvrzení jako zdroj, a vypovídat o něm prostřednictvím dalších tvrzení. Motivací vzniku této konstrukce byla zejména potřeba usuzovat o *věrohodnosti* jednotlivých tvrzení na základě důvěry v osobu, která je „vyslovila“ – takto se příslušná „meta-tvrzení“ dostávají na stejnou úroveň jako tvrzení základní, a není

RDF) v dostatečné míře obeznámena. Těm, kdo si základy XML potřebují doplnit, můžeme doporučit např. původní českou práci [58].

[34] Pojem reifikace (tj. „zvěcnění“, podle latinského „res“ – „věc“) se používá v celé řadě formálních kalkulů pro vyjádření transformace *tvrzení* jazyka na atomický *objekt*. Jde o syntakticky elegantní způsob, jak do jazyka zahrnout konstrukty (libovolných!) vyšších řádů, což s sebou ovšem nutně přináší odpovídající formálně-logické důsledky. V tomto textu budeme ovšem hovořit pouze o reifikaci v konkrétním případě RDF.

potřeba pro ně vytvářet zvláštní syntaktický aparát. Druhou stranou mince je ovšem podstatné zkomplikování formálních vlastností modelu.

9 Dodatek C – Základní pojmy deskripční logiky[35]

Deskripční logika („description logics“, DL) je pojem zastřešující celou řadu příbuzných logických formalismů. Jejich společným rysem je snaha zachytit strukturu tříd a relací chápaných obdobně, jako v systémech založených na rámcích resp. ontologiích. Od tradičních ontologických (a rámcových) jazyků se liší zejména v tom, že nepředpokládají apriorní vymezení vztahu třídy a podtřídy (taxonomie). Namísto toho je *subsumpce* tříd (konceptů) vyhodnocována na základě jejich popisů (deskripcí), což mohou být i poměrně komplikované logické výrazy – taxonomie tedy vzniká dynamicky. Zjišťování subsumpce je v DL jednou ze základních inferenčních operací. Vedle toho je podporováno testování *konzistence modelu*, tj. splnitelnosti formulí vymezujících třídy, a také zjišťování *příslušnosti instancí* ke třídám.

Teorie v DL (označovaná též jako „terminologie“) odpovídá konečné množině *axiomů*; k jejich konstrukci jsou použity *koncepty* vymezené *konceptuálními výrazy* („concept expressions“), *role* (binární relace) vymezené *relačními výrazy* („role expressions“), a *atributy* (funkce) vymezené *funkčními výrazy* („attribute expressions“). *Pojmenované* koncepty, role a atributy odpovídají atomickým výrazům, ze kterých se ostatní skládají.

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 složených výrazů je odvozena ze sémantiky jejich složek. Pro představu si uvedeme přehled konstrukcí definovaných v jedné z nejrozšířenějších DL[36].

Možné konceptuální výrazy (kromě explicitně pojmenovaných) jsou:

- *nejobecnější třída* („top“) a *nejspecifičtější třída* („bottom“), interpretované jako universum Δ^I resp. prázdná množina $\varnothing$
- *konjunkce*, *disjunkce* a *negace* konceptů, interpretované (pro koncepty *C* a *D*) jako průnik $C^I \cap D^I$, sjednocení $C^I \cup D^I$, a doplněk $\Delta^I - C^I$
- *existenční omezení* $\exists R.C$, interpretované (pro roli *R* a koncept *C*) jako množina instancí $x \in \Delta^I$, pro které platí $R^I(x) \cap C^I \neq \varnothing$

[35] Volně zpracováno podle [53], kap. 2 „Formal Foundations of Description Logics“. České překlady termínů jsou dílem autora, mají provizorní charakter, a v žádném případě neaspirují na obecné používání.

[36] Jde o deskripční logiku označovanou jako **ALC**.

- *univerzální omezení* $\forall R.C$, interpretované (pro roli *R* a koncept *C*) jako množina instancí $x \in \Delta^I$, pro které platí $R^I(x) \subseteq C^I$
- *omezení kardinality* $\geq nR$, resp. $\leq nR$, interpretované (pro roli *R*) jako množina instancí $x \in \Delta^I$, pro které platí $|R^I(x)| \geq n$, resp. $|R^I(x)| \leq n$
- *kvalifikované omezení kardinality* $\geq nR.C$, resp. $\leq nR.C$, interpretované (pro roli *R* a koncept *C*) jako množina instancí $x \in \Delta^I$, pro které platí $|R^I(x) \cap C^I| \geq n$, resp. $|R^I(x) \cap C^I| \leq n$
- *existenční omezení pro atribut* $\exists A.C$, interpretované (pro atribut *A* a koncept *C*) jako množina instancí $x \in dom(A^I)$, pro které platí $A^I(x) \in C^I$
- *univerzální*[37] *omezení pro atribut* $\forall A.C$, interpretované (pro atribut *A* a koncept *C*) jako množina instancí $x \in \Delta^I$, pro které platí $x \in dom(A^I) \Rightarrow A^I(x) \in C^I$
- *mapování hodnot atributů* $A = B$, resp. $A \neq B$, interpretované (pro atribut *A* a koncept *C*) jako množina instancí $x \in \Delta^I$, pro které $A^I(x) = B^I(x)$ resp. $A^I(x) \neq B^I(x)$.

Možné relační a funkční výrazy (kromě explicitně pojmenovaných) jsou:

- *nejobecnější role* („top role“), interpretovaná jako kartézský součin $\Delta^I \times \Delta^I$
- *konjunkce* a *disjunkce* rolí, interpretované (pro role *R* a *S*) jako průnik $R^I \cap S^I$ a sjednocení $R^I \cup S^I$
- *složení* rolí $R \circ S$, interpretované jako složení binárních relací $R^I \circ S^I$
- *identita* $id(C)$, interpretovaná jako množina uspořádaných dvojic $\langle x, x \rangle$, kde $x \in C^I$
- *inverzní role* R^{-1}, interpretovaná jako množina uspořádaných dvojic $\langle x, x' \rangle$, kde $\langle x', x \rangle \in R^I$
- *tranzitivní uzávěr* R^+, interpretovaný jako $\bigcup_{1 \leq n} (R^I)^n$
- *tranzitivní reflexivní uzávěr* R^*, interpretovaný jako $\bigcup_{0 \leq n} (R^I)^n$.

[37] Povšimněme si významné odlišnosti univerzálního omezení oproti existenčnímu: zatímco existenční omezení vyžaduje, aby instance příslušné relace *existovala*, v případě univerzálního omezení takový požadavek neplatí (relace může být prázdná). Tento jev se přenáší i do ontologických jazyků odvozených z DL.

Konečně, samotné axiomy se rozdělují na *uvozovací* („introduction axioms“), *obecné* („general terminological axioms“), a konečně *relační* axiomy („role axioms“) označující určitou roli za *funkční* (tj. za atribut) či *tranzitivní*.

Uvozovací axiomy jsou tyto:

- Uvození konceptu, role nebo atributu, spočívající v přiřazení konceptuálního (resp. relačního, funkčního) *výrazu* určitému *jménu* konceptu (resp. role, atributu). Interpretace pojmenovaného konceptu (role, atributu) se chápe jako *ekvivalentní* interpretaci příslušného výrazu.
- Uvození *primitivního* konceptu, role nebo atributu. Syntakticky obdobné, avšak interpretace pojmenovaného konceptu (role, atributu) se chápe jako (množinově) *obsažena* v interpretaci příslušného výrazu. Výraz tedy pro daný koncept (roli, atribut) představuje pouze *nutnou* ale nikoliv *postačující* podmínku.

Obecné axiomy se podobají uvozením *konceptu*, namísto jména konceptu však navzájem přiřazují *dva* výrazy. Hovoří se o *konceptuální rovnici* („concept equation“) resp. *obecné inkluzi konceptů* („general concept inclusion“).

Zjišťování *subsumpce* konceptů (a následná konstrukce částečného uspořádání nad koncepty, označovaná jako „klasifikace“), je založena na testování *množinové inkluze interpretací* příslušných konceptuálních výrazů. Výrazy jsou obvykle pro usnadnění výpočtu *rozloženy* („unfolded“), tj. za jména konceptů jsou rekurzivně dosazeny výrazy, kterými jsou tyto koncepty uvozeny[38].

Literatura

1. Alice, http://kmi.open.ac.uk/projects/alice/
2. CoMMA, http://www.ii.atos-group.com/sophia/comma/HomePage.htm
3. Cyc, http://www.cyc.com/
4. DAML, http://www.daml.org/
5. DAML ontology no.37, http://www.daml.org/ontologies/37
6. DAML+OIL, http://www.w3.org/TR/daml+oil-reference
7. Dublin Core, http://dublincore.org/
8. Enterprise Ontology, http://www.aiai.ed.ac.uk/~entprise/enterprise/ontology.html
9. EuroWordNet, http://www.illc.uva.nl/EuroWordNet/
10. GOL - General Ontological Language, http://www.ontology.uni-leipzig.de/
11. KAON, http://kaon.semanticweb.org/
12. KIF, http://www-ksl.stanford.edu/knowledge-sharing/kif/
13. OIL, http://www.ontoknowledge.org/oil/
14. OilEd, http://oiled.man.ac.uk/
15. OKBC, http://www.ksl.stanford.edu/software/OKBC/
16. Onions, http://saussure.irmkant.rm.cnr.it/onto

[38] V případě uvození primitivních konceptů je nutno do rozloženého výrazu přidat umělý primitivní koncept, který vyjadřuje „zjemnění“ výrazu na *postačující* podmínku.

17. Ontolingua Server, http://www-ksl-svc.stanford.edu:5915/
18. OntoPrise, http://www.ontoprise.de/
19. OntoWeb, http://www.ontoweb.org
20. OpenCyc, http://www.opencyc.org
21. Personal communication with Barry Smith, May 2002.
22. Potential business scenarios. OntoWeb Deliverable 1.2.1, online at http://www.ontoweb.org/download/deliverables/D1.2_V1.PDF
23. Protégé, http://protege.stanford.edu/
24. RDF, http://www.w3.org/RDF/
25. RDF Schema, http://www.w3.org/TR/rdf-schema/
26. Sensus, http://www.isi.edu/natural-language/projects/ONTOLOGIES.html
27. SHOE, http://www.cs.umd.edu/projects/plus/SHOE/
28. State of the art in content standards. OntoWeb Deliverable 3.1, online at http://www.ontoweb.org/download/deliverables/D3.1.pdf
29. Technical Roadmap v. 1.0. OntoWeb Deliverable 1.1, online at http://www.ontoweb.org/download/deliverables/D11_v1_0.pdf
30. UML, http://www.omg.org/uml/
31. UMLS, http://www.nlm.nih.gov/research/umls/
32. UNSPSC, http://eccma.org/unspsc/
33. Upper Ontology, http://ontology.teknowledge.com/
34. Web-Ontology (WebOnt) Working Group, http://www.w3.org/2001/sw/WebOnt/
35. WordNet, http://www.cogsci.princeton.edu/~wn/
36. XML Schema, http://www.w3.org/XML/Schema
37. XML Topic Maps, http://www.topicmaps.org/xtm
38. XOL Ontology Exchange Language. http://www.ai.sri.com/~pkarp/xol/
39. Berendt, B., Hotho, A., Stumme, G.: Towards Semantic Web Mining. In: (Horrocks, I., Hendler, J., eds.) The Semantic Web – ISWC 2002. Springer 2002.
40. Berners-Lee, T.: Primer: Getting into RDF & Semantic Web using N3. Online at http://www.w3.org/2000/10/swap/Primer.html
41. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American*, May 2001.
42. Borst, W.N.: *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD dissertation, University of Twente, Enschede, 1997.
43. Buchanan, B.G., Wilkins, D.C. (eds.): *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems.* Morgan Kaufmann, USA, 1993.
44. Chaudhri, V.K., Farquhar, A., Fikes R., Karp, P., Rice, J.P.: A Programmatic Foundation for Knowledge Base Interoperability. In: Proc. AAAI-98.
45. Decker, S. et al.: The Semantic Web: The roles of XML and RDF. *IEEE Internet Computing*, (2000), Vol. 15, No. 3, 63-74.

46. Domingue, J.: Tadzebao and WebOnto: Discussing, browsing, and editing ontologies on the web. In Proceedings of the 11th Knowledge Acquisition for KnowledgeBased Systems Workshop, April 18th-23rd. Banff, Canada,
47. Etzioni, O., Gribble, S., Halevy, A., Levy, H., McDowell, L.: An Evolutionary Approach to the Semantic Web. In: Collected Posters, First International Semantic Web Conference (ISWC2002).
48. Fensel, D., Decker, S., Erdmann, M., Studer, R.: Ontobroker: Or How to Enable Intelligent Access to the WWW. In: Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW98).
49. Fensel, D. et al.: The Unified Problem-solving Method Development Language UPML. *Knowledge and Information Systems* (to appear), 2002.
50. Gómez Pérez, A., Benjamins, V.R.: Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. In: *Proc. of IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends.* Proceedings available online at http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/.
51. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2) (1993).
52. Haustein, S., Pleumann, J.: Is Participation in the Semantic Web Too Difficult? In: (Horrocks, I., Hendler, J., eds.) The Semantic Web – ISWC 2002. Springer 2002.
53. Horrocks, I.: *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
54. Horrocks, I.: The FaCT system. In H. de Swart, editor, Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98, number 1397 in Lecture Notes in Artificial Intelligence, pages 307-312. Springer-Verlag, Berlin, May 1998.
55. Karp, P.D., Chaudhri, V.K., Thomere, J.: XOL: An XML-Based Ontology Exchange Language. In: Bio-Ontologies'99 Meeting, 1999. Online at ftp://smi.stanford.edu/pub/bio-ontology/xol.doc.
56. Kavalec, M., Svátek, V.: Information Extraction and Ontology Learning Guided by Web Directory. In: ECAI Workshop on Natural Language Processing and Machine Learning for Ontology Engineering. Lyon, 2002.
57. Klein, M., Fensel, D., van Harmelen, F., Horrocks, I.: The relation between ontologies and XML schemas, *Electronic Transactions on Artificial Intelligence* (ETAI), Linköping Electronic Articles in Computer and Information Science, vol. 6(4), 2001.
58. Kosek, J.: *XML pro každého*. Grada Publishing, Praha, 2000.
59. Maedche, A.: *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, 2002.
60. McEntire, R. et al.: An Evaluation of Ontology Exchange Languages for Bioinformatics. In: 8th Intl. Conf. Intelligent Systems for Molecular Biology, ISMB 2000, La Jolla, California.

61. Motik, B., Maedche, A., Volz, R.: A Conceptual Modeling Approach for Semantics-driven Enterprise Applications (white paper). Přístupné na adrese http://kaon.aifb.uni-karlsruhe.de/conc-model.
62. Motta, E.: *Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design*. IOS Press, Amsterdam, 1999.
63. Newell, A.: The knowledge level. *Artificial Intelligence*, 18 (1982), 87-127.
64. Patel-Schneider, P.F., Simeon, J.: Building the Semantic Web on XML. In: (Horrocks, I., Hendler, J., eds.) The Semantic Web – ISWC 2002. Springer 2002.
65. Pan, J., Horrocks, I.: Metamodeling architecture of web ontology languages. In: Proc. of the First Semantic Web Working Symposium (SWWS'01).
66. Pisanelli, D.M., Gangemi, A., Steve, G.: An Ontological Analysis of the UMLS Metathesaurus, *J. Amer. Medical Informatics Association* 5 (1998), 810-814.
67. Reed, S.L., Lenat, D.B.: Mapping Ontologies into Cyc (white paper). Přístupné na http://www.cyc.com/doc/white_papers/mapping-ontologies-into-cyc_v31.pdf.
68. Rousset, M. C.: The Semantic Web Needs Languages for Representing (Complex) Mappings Between Simple Ontologies. *IEEE Intelligent Systems*, 17(2):79-80, March/April, 2002. Contribution to the section "Trends & Controversies: Ontologies KISSES in Standardization", edited by S. Staab.
69. Schreiber, G. et al.: *Knowledge Engineering and Management – The CommonKADS Methodology*. MIT Press, Cambridge, Massachusetts; London, England, 1999.
70. Schreiber, G.: The Web is not well-formed. *IEEE Intelligent Systems*, 17(2):79-80, March/April, 2002. Contribution to the section "Trends & Controversies: Ontologies KISSES in Standardization", edited by S. Staab.
71. Staab, S., Erdmann, M., Maedche, A.: Engineering ontologies using semantic patterns. In A. Preece & D. O'Leary (eds.), Proceedings of the IJCAI-01 Workshop on E-Business & the Intelligent Web. Seattle, WA, USA, 2001.
72. Sure, Y., Erdmann, M., Studer, R.: OntoEdit: Collaborative Engineering of Ontologies. In: On-To-Knowledge: Semantic Web enabled Knowledge Management. J. Davies, D. Fensel, F. van Harmelen (eds.), Wiley, to appear 2002.
73. Uschold, M., Gruninger, M.: Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, Vol.11:2, 1996, 93-136.
74. van Harmelen, F., Fensel, D.: Practical Knowledge Representation for the Web. In: Proceedings of the Workshop on Intelligent Information Integration (III99) during IJCAI-99, Stockholm, Sweden, August 1999.
75. van Heijst, G., Schreiber, G., Wielinga, B.J.: Using Explicit Ontologies in KBS Development. *Int. J. Human-Computer Studies* - special issue, 1996.