IME-USP

Instituto de Matemática e Estatística da Universidade de São Paulo

Monografia - MAC5701 - Tópicos em Computação

Ontologias, Web Semântica e Aplicações

São Paulo, Junho de 2003

Aluno
Márcio Rodrigo de Freitas Carneiro

Orientadora
Prof. Dra. Renata Wassermann

Sumário

Lista de Figuras

1 Introdução

A rede mundial de computadores, ou a Internet, e mais especificamente seu conteúdo hipertexto, a WWW, cada vez mais se torna senso comum na vida das pessoas.

Talvez o motivo mais forte pelo grande sucesso da WWW seja sua simplicidade de acesso. Entretanto, com o vertiginoso crescimento da quantidade e diversidade de informação disponibilizada, o paradigma simples de organização dos documentos e recursos da teia se torna insuficiente.

As fraquezas que surgem na atual estrutura da teia [SAB+03]:

- Busca de informação: as ferramentas atuais de busca, pois mais rápidas que sejam, em geral trazem informações não desejadas, bem como ignoram o contexto dos termos ou relação entre eles.

- Extração de informação: a obtenção de informações dificilmente pode ser automatizada, pois exige interpretação humana para seleção e filtragem.

- Manutenção: a manutenção do conteúdo de grandes sítios pode ser bastante complicada, considerando a estrutura simples de organização dos documentos.

O caminho para suprir essas necessidades da teia aparece através da adição de semântica ao conteúdo da Web. A idéia é a criação de uma estrutura lógica, bem formada, que descreva algum nível de semântica dos documentos e estruturas da teia, a qual humanos e agentes autônomos computacionais possam aproveitar para melhor realizarem suas tarefas. Essa visão, chamada de *Web Semântica* surge principalmente de Berners-Lee [BL00].

Para a realização da Web Semântica, é necessária uma arquitetura que permita a adição de semântica aos documentos e recursos da teia. Para isso, faz-se necessária uma linguagem de marcação de páginas, para a descrição de metadados, isto é, informação sobre informação. Mas somente isso não é suficiente, pois queremos também descrever semântica, isto é, por

exemplo, descrever como um documento ou trechos dele se encaixa em conceitos de determinada área. Portanto, precisamos de modelos teóricos, uma linguagem que vá além da marcação, que de maneira formal e estruturada descreva os conceitos da área de informação, para que então possamos descrever as instâncias daquela área de informação, ou seja, descrever a informação de maneira propriamente dita.

Nesse âmbito aparecem as pesquisas de lógicas descritivas e conseqüentemente as ontologias e linguagens relacionadas. A comunidade de pesquisa de lógica rapidamente se encaixou na busca de padrões da teia, e vários modelos já se apresentam para uso prático.

Finalmente, aplicações reais devem ser desenvolvidas. Somente ter o conteúdo semântico disponível nada nos traz: é preciso utilizá-lo de maneira sistemática para facilitar a busca, manipulação, visualização e transformação da informação. Vários projetos buscam desenvolver ferramentas diversas.

2 Ontologias e semântica

Um bom ponto de partida é o entendimento do que é uma ontologia. Ao procurarmos o significado de ontologia num dicionário, encontraremos ontologia como um ramo da filosofia, que estuda a natureza do que é ser:

> Ontologia [De ont(o)- + -logia.] S. f. Parte da filosofia que trata do ser enquanto ser, i.e., do ser concebido como tendo uma natureza comum que é inerente a todos e a cada um dos seres. (Novo Aurélio, Editora Nova Fronteira)

Entretanto, para a comunidade computacional, o termo ainda é bastante discutido e usado de maneira diversa. Claramente as intenções são extremamente parecidas. Mas podemos adotar algum vocabulário restrito e específico, como tratado na boa discussão de [GG95]:

conceitualização estrutura semântica **intencional**[1] que codifica as regras implícitas que

[1] Semânticas intencionais são semânticas que dependem da situação ou estado do mundo descrito. Esse tipo de estrutura é bem qualificada para descrever realidades dinâmicas

limitam a estrutura de parte da realidade

ontologia como sinônimo de conceitualização, ou uma teoria lógica que contabiliza explicitamente e parcialmente uma conceitualização

Podemos resumir a idéia, dizendo que uma ontologia é um vocabulário que define a estrutura semântica de uma realidade. Gradativamente é possível esclarecer melhor a idéia e portanto uma comparação com conceitos conhecidos, como os tesauros, pode ser útil.

2.1 Tesauros

Tesauros são um conceito mais comum na vida das pessoas. Um tesauro é uma organização de uma área de conhecimento, de forma estruturada. Em geral, é formado de termos organizados em hierarquia, podendo conter descrição como num dicionário, mas também contendo relacionamentos entre os termos. Claramente a semântica na descrição do conhecimento pelo tesauro é bastante limitada, tornando-o mais uma organização sintática das palavras.

Um exemplo simples de um tesauro segue na figura 1.

Podemos ver que um tesauro pode ser representado por um conjunto de árvores, já que a hierarquia entre os termos não permite mais de um termos geral (apenas um pai). A relação TE (termo específico) e TG (termo geral) é relação de herança (filho e pai, respectivamente), e a relação TR conecta nós de níveis diferentes ou de árvores diferentes. A relação SN e UP são inversas e restringem o uso do vocabulário. Ao se indicar o sinônimo de um termo (SN), indica-se o termo oficial para aquele conceito, e a relação UP indica quais termos são sinônimos do termos oficial. O atributo NE (nota de escopo) fornece uma descrição textual do termo, como num dicionário.

Apesar de parecer simples, pouco semântica e mais sintática, essa organização do conhecimento já traz diversas vantagens na busca informações dentro de uma base de conhecimento. Veremos adiante formas mais expressivas de formalizar a estrutura do conhecimento.

Empregador TR *trabalhador* TR *trabalho* TG *pessoa*

Pessoa NE ser na sociedade, seja ele humano ou apenas representativo.

TE *trabalhador* TE *empregador*

Trabalho NE (nota de escopo).

TE *trabalho manual*

TR *trabalhador* TR *empregador*

Trabalhador TE *trabalhador operario* TR *empregador* TR *trabalho* TG *pessoa*

Trabalhador operario TR *trabalho manual* TG *trabalhador*

UP *operario*

Trabalho manual TR *trabalhador operario* TG *trabalho*

Operario SN *trabalhador operario*

Figura 1: Exemplo de tesauro.

2.2 Lógicas de descrição [BHS03]

Lógicas de descrição (*description logics* ou DLs) são uma família de linguagens de descrição de conhecimento que podem ser usadas para descrever o conhecimento de um domínio de maneira estruturada e bem formada.

As LDs utilizam um formalismo para descrição de conceitos atômicos (predicados unários) e papéis atômicos (predicados binários). Os construtores em geral estão associados a teoria dos conjuntos, como por exemplo o construtor *conjunção* ($\sqcap$), que equivale à intersecção de conjuntos, ou o construtor *negação* ($\neg$), equivalente ao complemento de conjuntos.

Um exemplo simples pode ilustrar uma descrição:

$$\mathsf{Pessoa} \sqcap \exists \mathsf{temTrabalho}.\mathsf{TrabalhoManual}$$

Que representa o conceito de "pessoa que tem um trabalho que é um trabalho manual". Poderíamos nomear esse conceito de Operario. Repare que o tesauro não afirma isso explicitamente, apenas relaciona o termo operário com o termo trabalho manual. Aqui, temos

um conceito (Pessoa), e um papel (temTrabalho), que é um predicado binário, pois relaciona Pessoa com o outro conceito TrabalhoManual.

As LDs também se utilizam de um formalismo terminológico e de afirmações. Podemos criar *axiomas terminológicos*:

$$\exists \mathsf{temTrabalho}.\mathsf{Trabalho} \sqsubseteq \mathsf{Trabalhador}$$

Que restringe o papel temTrabalho ao conceito de Trabalhador, isto é, apenas trabalhadores podem ter trabalho.

Também há o *formalismo afirmativo*, para afirmações sobre indivíduos:

$$\mathsf{TrabalhadorManual}(\mathsf{PEDRO}),\ \mathsf{temTrabalho}(\mathsf{PEDRO}, \mathsf{SOLDADOR})$$

O ponto mais interessante das lógicas descritivas é a capacidade de inferência e dedutibilidade. Para isso, alguns algoritmos são oferecidos, como o algoritmo de *subsumption*, que determina a hierarquia entre conceitos, o algoritmo de *instância*, que determina a relação entre as instâncias, e o algoritmo de *consistência*, que determina se uma base de conhecimento é consistente ou não.

As lógicas descritivas podem ser mais expressivas ou não, o que se opõe à possibilidade de decisão dos algoritmos. Esse balanço entre poder e praticidade é pesquisado desde o surgimento das LDs, na década de 80.

Como podemos ver nos exemplo, as lógicas de descrição são mais expressivas que um tesauro. Veremos adiante como as LDs se encaixam nas descrições de ontologia, e como essas ontologias são casos mais genéricos e expressivos dos tesauros.

2.2.1 A LD $\mathcal{SHIQ}$

A linguagem de lógica descritiva $\mathcal{SHIQ}$ [HST99] se contrasta com outras linguagens de lógica descritiva por também permitir construção de papéis bastante expressivos, ao invés de se concentrar apenas nas construções de conceitos. Os exemplos acima são baseados nessa lógica de descrição.

Um fator importante para a aplicação prática das LDs é a possibilidade de implementação e uso de inferência efetivamente. Para a linguagem $\mathcal{SHIQ}$ existe um programa de raciocínio chamado FaCT que realiza otimizações que tornam possíveis as inferências na prática, mesmo as que levam a uma complexidade computacional exponencial.

2.3 Ontologias e OIL

Podemos entender melhor agora o que é uma ontologia. Mais que o tesauro, uma ontologia é a descrição formal da semântica de uma realidade. Portanto, além de descrevermos os conceitos, criamos relações entre os conceitos, ou seja papéis.

Nas linguagens e pesquisas de ontologias, os termos classe e propriedade são usados equivalentemente aos termos conceito e papel das lógicas de descrição. Essa terminologia se assemelha às idéias de orientação a objetos. Entretanto, nas ontologias as classes e propriedades são criadas independentemente, e as relações entre elas são formalizadas através de axiomas.

Em meados da década de 90, dois grandes projetos se iniciaram paralelamente, o OnToKnowledge (http://www.ontoknowledge.org), do EU IST, e o DAML (http://www.daml.org), financiado pela agência de projetos de defesa americana, a DARPA, com o objetivo principal de definir linguagens, ferramentas, e aplicações para a Web Semântica.

O primeiro resultado do projeto OnToKnowledge foi o OIL, ou *Ontology Inference Layer*. É uma proposta de camada de inferência para a Web Semântica, contendo principalmente uma linguagem para descrição de ontologias. A OIL está intimamente ligada às lógicas de descrição, em particular à LD $\mathcal{SHIQ}$. Inicialmente, a linguagem foi desenvolvida com uma sintaxe simples, sem nenhuma ligação com a Web e outros padrões que veremos adiante (seção 3.1).

Um exemplo simples de uma ontologia descrita em OIL pode ser visto na figura 2.

Podemos verificar uma semelhança estrutural com um tesauro. Entretanto, a ontologia traz semântica à formalização dos conceitos. Tal como as DLs, podemos verificar restrições

class-def Pessoa

class-def Trabalho

class-def Trabalhador

    subclass-of Pessoa

slot-def TemTrabalho

    domain Trabalhador

    range Trabalho

instance-of PEDRO Trabalhador

instance-of SOLDADOR Trabalho

related TemTrabalho PEDRO SOLDADOR

Figura 2: Exemplo de descrição em OIL.

e associações específicas, para cada classe e propriedade existente no mundo real.

A partir dos algoritmos de inferência podemos formalizar conceitos implícitos na ontologia. A herança de classes e propriedades nos permite deduzir que todo trabalhador é uma pessoa, e contém as características inerentes a esse conceito. A partir da afirmação de que a instância PEDRO tem trabalho SOLDADOR, podemos inferir que PEDRO é um trabalhador, mesmo isso não estando explícito (caso contrário haveria uma inconsistência na ontologia ou nas descrições de instância).

3 Web Semântica

Como já explicado, a Web Semântica é a idealização de uma expansão da WWW, de um nível apenas sintático, para um nível semântico, no que diz respeito a sua estrutura e organização.

A realização desse sonho não é uma tarefa simples. Um dos processos importantíssimos é a definição de padrões. É necessária uma linguagem para definir ontologias. Além disso, uma

sintaxe para a marcação das páginas com informações semânticas, ou mais especificamente, ontológicas. Os projetos atuais caminham para essa primeira padronização: garantir as ferramentas necessárias para a expansão da teia.

O primeiro passo para essa padronização é uma linguagem bem definida, com um sintaxe dentro das tendências da WWW. A própria W3C já desenvolvia idéias de marcação de página antes mesmo da emergência da Web Semântica.

3.1 RDF e RDFS

O padrão de marcação de páginas para diagramação, tão conhecido entre desenvolvedores e usuário, o HTML, atingiu um sucesso enorme, talvez principalmente dada sua simplicidade. O próprio padrão contém termos específicos para a inclusão de metadados, isto é, dados sobre os dados, para descrição de termos relacionados ao documento em si. Entretanto, esses cabeçalhos de metadados foram pouco usados ao longo dos anos, e os próprios agentes de buscas ignoram esse conteúdo.

Certamente não foi atingido o sucesso na necessidade de se marcar informações extras sobre os documentos e recursos da teia. Para isso, um novo padrão foi criado, e começou a ser discutido sem vínculo com ontologias.

O RDF, ou *Resource Definition Framework*, é um modelo para a descrição de metadados de recursos da teia. Tudo descrito pelo RDF é recurso, preferencialmente referenciado por uma URI (*Unified Resource Identification*). A idéia básica é simples: um documento RDF é um conjunto de afirmações, que são triplas, formadas de **sujeito**, **predicato**, e **objeto**. Os predicados são recursos especiais, que no RDF são chamados de Propriedades (`rdf:Property`). Assim, podemos definir propriedades, e usá-las para descrever um documento, através do RDF. O exemplo mais claro relacionado aos documentos da teia são informações como título, autor, data de criação, etc. Os sujeitos e objetos são em geral recursos (`rdf:Resource`), entretanto os objetos também podem ser literais, ou seja, cadeias de caracteres ou *strings*.

Como podemos perceber, esse modelo pode ser associado a um grafo. Os predicados são os arcos, e os objetos e predicados são nós. Como tudo no RDF é recurso, inclusive as propriedades, nada impede que afirmações sejam feitas com as propriedades no papel sujeito ou objeto.

A sintaxe criada para o RDF foi baseada no XML, já que o objetivo principal é padronização na WWW. Entretanto, a estrutura intrínseca ao XML é de uma árvore (elementos contendo elementos), e o RDF é um modelo de grafos. Portanto, a descrição é baseada em elementos que descrevem triplas, como podemos ver na figura 3.

```
<rdf:Description rdf:about="http://exemplo.br/doc/index.html">
    <ex:criador rdf:resource="http://exemplo.br/user/marcio"/>
    <ex:data>10 de junho de 2003</ex:data>
    <ex:titulo>Documento de exemplo de RDF</ex:titulo>
</rdf:Description>
```

Figura 3: Exemplo de RDF.

De fato, podemos ter mais de uma tripla descrita numa afirmação. Em geral, o atributo `rdf:about` contém o sujeito (pode-se utilizar o termo `rdf:nodeID` para criação de nós "anônimos" — define-se uma identidade para ser referenciada no documento como um recurso). Os elementos internos ao `rdf:Description` são os arcos (propriedades, no exemplo `ex:criador`, `ex:data`, e `ex:titulo`). Em geral, usa-se um atributo para referenciar o recurso que é o objeto (pode-se utilizar `rdf:nodeID` para referenciar um nó "anônimo"), ou um elemento texto, para a criação de um literal (como o caso da data e do título).

O RDF originalmente apenas provê vocabulário para definição de propriedades (`rdf:Property`). Também provê o termo `rdf:type`, para definição do tipo de um recurso. Entretanto, não há como definir classes de recursos, isto é, essa idéia não existe no RDF originalmente. Como podemos ver, o RDF é mais um modelo sintático e estrutural, e não especifica uma semântica bem formada e definida. A interpretação das afirmações ficam de certa maneira livre.

Há também outras estruturas que o RDF define, como contêineres (`rdf:Bag`, `rdf:Set`, `rdf:Alt`), e coleções (`rdf:Collection`) que são listas ligadas, além de termos como `rdf:seeAlso`, e `rdf:defineBy`. Entretanto, esses detalhes não são tão interessantes ao próximo passo, que é a definição de vocabulário e finalmente de ontologias.

3.1.1 Definição de vocabulário: RDF Schema

Da mesma maneira que o XML requer uma linguagem para a definição de como um documento deve se estruturar, o RDF requer uma linguagem para a descrição de vocabulários. Assim, o RDFS, ou *RDF Schema*, está para o *XML Schema* como o RDF está para o XML.

O RDFS aumenta o vocabulário do RDF, e o conjunto RDF/RDFS se assemelha muito a uma linguagem de ontologia. O conceito de classe é adicionado, através do termo `rdfs:Class`. Além disso, a idéia de domínio e alcance de propriedades aparece, com os termos `rdf:domain`, e `rdf:range`. Podemos então definir a quais classes como sujeito (domínio) e objeto (alcance) uma propriedade se aplica.

O RDF/RDFS se mostrou efetivo como padrão de sintaxe e descrição de vocabulário. O modelo não contém toda expressividade que as linguagens de ontologia permitem, apesar de ser mais abrangente em outro aspecto. O modelo RDF permite reificação, isto é, pode-se criar um recurso que é uma afirmação. Para tanto, o termo `rdf:Statement` pode ser usado, criando um recurso que é uma tripla, e que pode ser referenciado nas descrições. Essa característica é bastante abrangente, mas traz problemas ao processo de inferência e dedutibilidade.

Apesar dessas diferenças, a sintaxe ou linguagem RDF/RDFS se tornou realmente o padrão para as linguagens de ontologia da teia, o que veremos adiante.

3.2 Linguagem de ontologia para a Web Semântica

Alguns fatores são muito importantes na decisão de projeto para uma linguagem de ontologia. Os três mais importantes são a complexidade computacional, técnica e conceitual

[vH02b].

A complexidade computacional tem sido uma preocupação freqüente nas pesquisas, e mesmo que haja casos extremos, pode-se verificar bom comportamento das ferramentas de raciocínio e inferência na prática e na maioria dos casos reais.

A complexidade técnica, que diz respeito a implementação de ferramentas e aplicações, tem se mostrado superada, principalmente pela adoção de padrões da W3C.

Já a complexidade conceitual, que está relacionada com a dificuldade dos usuários comuns compreenderem e utilizarem a linguagem de ontologia, ainda está sob atenção dos desenvolvedores, para se descobrir o que realmente é importante na linguagem, deixando-a completa e ao mesmo tempo simples no entendimento.

Veremos o principal resultado das pesquisas recentes, e a atual tendência de padrão da W3C como linguagem de ontologia.

3.3 DAML+OIL

No processo de pesquisa de uma linguagem de ontologia para a Web Semântica, o RDF/RDFS se tornou padrão de sintaxe, já que estava sendo bem aceito e usado por outras atividades.

Ao mesmo tempo que a OIL se desenvolvia e se encaixava na sintaxe do RDF/RDFS, a pesquisa do DAML também se baseava nessa sintaxe, e adicionava conceitos semelhantes ao OIL, como classe, restrições locais, e outras construções inerentes das lógicas de descrições.

Um esforço em conjunto foi então realizado para a união de experiências e características de ambas linguagens semelhantes, a OIL e a DAML. O resultado final desse esforço foi a especificação chamada DAML+OIL.

Basicamente, a DAML+OIL é equivalente a lógica de descrição $\mathcal{SHIQ}$, com a adição do construtor `oneOf` e dos tipos de dados (chamados de domínios concretos nas LDs) [PSHvH02].

Os construtores de classe da linguagem podem ser vistos na figura 4, comparativamente com a sintaxe de LD e com exemplos. Podemos ver claramente que temos construtores

mais expressivos que no RDFS. Podemos definir cardinalidade das propriedades, bem como restrições locais às propriedades, i.e., restrições específicas para cada classe.

Construtor	Sintaxe DL	Exemplo
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	Trabalho $\sqcap$ Faculdade
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	Trabalhador $\sqcup$ Empregador
`complementOf`	$\neg C$	$\neg$Pessoa
`oneOf`	$\{x_1, \ldots, x_n\}$	{PEDRO, MARIA}
`toClass`	$\forall P.C$	$\forall$temEmpregado.Operario
`hasClass`	$\exists P.C$	$\exists$temTrabalho.Manual
`hasValue`	$\exists P.\{x\}$	$\exists$temTrabalho.{SOLDADOR}
`minCardinalityQ`	$\geq n\ P.C$	≥ 2 hasEmpregado.Operario
`maxCardinalityQ`	$\leq n\ P.C$	≤ 1 hasEmprego.Trabalho
`cardinalityQ`	$= n\ P.C$	$= 1$ hasEmpregador.Empregador

Figura 4: Construtores de classe DAML+OIL.

O construtor `oneOf` permite a definição de classes através da enumeração de seus elementos. Essa criação de classe através da definição explícita de sua existencialidade não é utilizada nas LDs.

Os três primeiros construtores são apenas operadores "booleanos" de classes. Os construtores `hasClass` e `toClass` são restrições locais. A construção `toClass` restringe a aplicação da propriedade P sempre a elementos da classe C, para a classe especificada. Já a construção `hasClass` restringe a aplicação da propriedade P a pelo menos um elemento da classe C. O construtor `hasValue` é o mesmo que a aplicação de `hasClass` e `oneOf`.

Os construtores de cardinalidade são generalizações dos construtores `hasClass` e `hasValue`, especificando quantidades específicas dos elementos que são referenciados pela propriedade.

Além dos construtores de classe, a linguagem permite axiomas para afirmar generalização e equivalência de classes e propriedades, a disjunção de classes, equivalência ou não equivalência de indivíduos, e propriedades de propriedades. A figura 5 resume esses axiomas permitidos na linguagem.

A DAML+OIL foi tomada como base para a definição final de uma linguagem de ontologia da W3C. Veremos adiante essa linguagem muito semelhante a DAML+OIL, já que derivada

Construtor	Sintaxe DL	Exemplo
`subClassOf`	$C_1 \sqsubseteq C_2$	Trabalhador $\sqsubseteq$ Pessoa
`sameClassOf`	$C_1 \equiv C_2$	Homem $\equiv$ Pessoa $\sqcap$ Masculino
`subPropertyOf`	$P_1 \sqsubseteq P_2$	temSalario $\sqsubseteq$ temTrabalho
`samePropertyAs`	$P_1 \equiv P_2$	salario $\equiv$ ganhos
`disjointWith`	$C_1 \sqsubseteq \neg C_2$	Empregado $\sqsubseteq$ Desempregado
`sameIndividualAs`	$\{x_1\} \equiv \{x_2\}$	{PEDRO} $\equiv$ {PEDRO CARVALHO}
`differentIndividualFrom`	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{PEDRO} $\sqsubseteq \neg$ {MARIA}
`inverseOf`	$P_1 \equiv P_2^-$	temPatrao $\equiv$ temEmpregador$^-$
`transitiveProperty`	$P^+ \sqsubseteq P$	ancestral$^+ \sqsubseteq$ ancestral
`uniqueProperty`	$T \sqsubseteq \leq 1P$	$T \sqsubseteq \leq$ 1temEmpregador
`unambiguousProperty`	$T \sqsubseteq \leq 1P^-$	$T \sqsubseteq \leq$ 1temEmpregado$^-$

Figura 5: Axiomas DAML+OIL.

da mesma, e então daremos um exemplo prático na sintaxe RDF/RDFS.

3.4 OWL

A OWL, ou *Ontology Web Language* se propõe a ser uma definição padrão de linguagem ontológica para a teia, proposta pelo grupo W3C. Ainda se encontra em fase de definição, mas já com caminhos, estruturas e semântica bem definidas.

O documento [DCvH+02] dá uma visão geral da linguagem. No caminho da W3C para a Web Semântica, a OWL se encontra como passo atual do desenvolvimento. Primeiramente, o XML provê a sintaxe para documentos estruturados, mas sem qualquer imposição de restrições semânticas no significado dos documentos. O XML Schema restringe a estrutura de um documento XML. Surge então o RDF, um modelo de dados para descrever recursos e as relações entre eles, provendo uma semântica simples. O RDF é baseado na sintaxe XML. Logo então aparece o RDF Schema, um vocabulário para permitir a descrição de propriedades e classes com alguma expressividade limitada. Finalmente, o OWL acrescenta vocabulário e definições mais formais para a descrição de ontologias, como relações entre classes, cardinalidade, igualdade, tipos de propriedades complexos, etc.

A OWL é subdividida em três linguagens: *OWL Lite*, *OWL DL*, e *OWL Full*. De fato,

a OWL Lite é um subconjunto da OWL DL, que é subconjunto da OWL Full. Qualquer ontologia legalmente descrita em OWL Lite deve ser uma ontologia legalmente em OWL DL. Qualquer ontologia legal em OWL DL deve ser legal em OWL Full. Vejamos os detalhes gradativos das sub-linguagens.

3.4.1 OWL Lite

A OWL Lite é específica para necessidades básicas dos usuários, com restrições simples. Cardinalidade é suportada apenas com valores 0 ou 1. A OWL Lite é o mais simples a ser implementado, e pode ser uma boa alternativa para migração de tesauros e taxonomias.

Características herdadas do RDF Schema:

- `Class`: agrupamento de indivíduos que compartilham de propriedades semelhantes. Há uma classe genérica, super-classe de todas classes (e classe de todos indivíduos), chamada de Thing.

- `subClassOf`: permite a criação de hierarquia entre as classes. Pode-se inferir que um indivíduo pertence a outras classes, se a sua classe é sub-classes das mesmas.

- `rdfs:Property`: propriedades criam relações entre indivíduos (propriedades de objetos ou Object Properties) e entre indivíduos e tipos de dados (propriedades de dados ou Datatype Properties).

- `rdfs:subPropertyOf`: permite a criação de hierarquia entre as propriedades. Permite a mesma possibilidade de inferência, isto é, se um indivíduo está relacionado a outro por uma propriedade P, também está relacionado pelas super-propriedades de P.

- `rdfs:domain`: o domínio de uma propriedade limita a quais classes a propriedade pode ser aplicada, isto é, a parte esquerda da relação binária. Da mesma maneira que no RDFS, o domínio é uma restrição global, pois se aplica a uma propriedade como um

todo, e não especificamente para o caso da propriedade associada a uma determinada classe.

- `rdfs:range`: o alcance da propriedade limita os indivíduos que a propriedade pode ter como valor, isto é, a parte direita da relação binária. Como acima, essa restrição é global. Veremos adiante as restrições locais.

- `Individual`: os indivíduos são instâncias de classes, e as propriedades podem ser usadas para relacionar indivíduos.

Características de igualdade e desigualdade:

- `equivalentClass`: pode-se afirmar que duas classes são equivalentes. Equivalente ao `sameClassOf` do DAML+OIL (figura 4).

- `equivalentProperty`: também pode-se afirmar que duas propriedades são equivalentes. O mesmo que `samePropertyAs` do DAML+OIL.

- `sameIndividualAs`: pode-se explicitar que dois indivíduos são equivalentes.

- `differentFrom`: pode-se explicitar que dois indivíduos são diferentes, como no DAML+OIL (`differentIndividualFrom`). Importante para quando não há identificador único nas instâncias das classes.

- `allDifferent`: um conjunto de indivíduos pode ser declarado como diferentes entre si.

Características de propriedades e seus valores na OWL:

- `inverseOf`: uma propriedade pode ser explicitamente descrita como inverso de outra. Isso garante a inferência de que se um indivíduo X se relaciona a outro Y por uma propriedade P_1, e P_2 é o inverso de P_1, então Y se relaciona a X por P_2.

- `TransitiveProperty`: semelhante a mesma idéia do DAML+OIL. Podemos dizer que uma propriedade é transitiva. Se o par (x, y) é instância de uma propriedade transitiva P, e o par (y, z) também é instância de P, então o par (x, z) é instância de P, pela transitividade. A OWL Lite (e a OWL DL) impõe a condição às propriedades transitivas de restrição máxima cardinalidade igual 1. Sem isso, elas seriam linguagens sem decisão, que não é o objetivo do projeto (a OWL Full contém essa característica).

- `SymmetricProperty`: as propriedades podem ser declaradas simétricas, isto é, se P é simétrica, e o par (x, y) é instância de P, então (y, x) também é instância de P. Essa característica impõe que o domínio e alcance de P não podem ser arbitrários.

- `FuncionalProperty`: propriedades funcionais têm um valor único. É equivalente a afirmar que a propriedade tem a restrição de mínima cardinalidade igual a 0 e máxima cardinalidade igual a 1.

- `InverseFuncionalProperty`: uma propriedade pode ser declarada como ser inversamente funcional. Isto é, o inverso da propriedade pode ter no máximo um valor (isto é, ter valor único).

Restrições de tipo nas propriedades (restrições locais):

- `allValuesFrom`: essa restrição é declarada para restringir todos os valores de uma propriedade, mas em respeito a determinada classe. Por exemplo, podemos dizer que a propriedade temTrabalho para a classe Operario está restrita aos valores TrabalhoManual. Assim, todos os operários têm trabalho manual, irrevogavelmente. Note que essa característica é equivalente à `toClass` do DAML+OIL (figura 4, $\forall P.C$ — conjunto de todos elementos pertencente a C, através da propriedade P). Entretanto, o nome da propriedade na OWL é bem mais intuitivo para o usuário comum.

- `someValuesFrom`: também é uma restrição local, pois restringe a propriedade em relação a determinada classe. Semelhante à `hasClass` do DAML+OIL (figura 4, $\exists P.C$).

Determina que um indivíduo da classe tem pelo menos um valor especificado pelo `someValuesFrom`, associado pela propriedade especificada.

Restrições de cardinalidade (um dos pontos em que a OWL Lite é limitada - os valores de cardinalidade podem ser apenas 0 ou 1):

- `minCardinality`: pode-se afirmar a cardinalidade mínima de uma propriedade em relação a uma classe. Note-se que é uma restrição local também. Afirmar que a cardinalidade mínima é 0 apenas explicita que a propriedade não é obrigatória. Dizer que a cardinalidade mínima é 1, afirma que a propriedade tem que ser aplicada pelo menos uma vez à classe.

- `maxCardinality`: também se pode afirmar a cardinalidade máxima de uma propriedade em relação a uma classe. Afirmar que a cardinalidade máxima é 0 equivale a dizer que propriedade não pode ser aplicada. Afirmar que a cardinalidade máxima é 1, implica em afirmar que só podemos ter no máximo um elemento para aquela propriedade.

- `cardinality`: é apenas o mesmo que afirmar cardinalidade mínima e máxima com os mesmo valores. Assim, se limita o número exato de relacionamentos da classe com aquela propriedade.

Os valores de cardinalidade são limitados a 0 e 1 para que a OWL Lite seja mais simples de se implementar. Essa restrição não existe na OWL DL, nem na OWL Full.

Características de classe (apenas intersecção para OWL Lite):

- `intersectionOf`: pode-se declarar classes como intersecção de classes e restrições. Por exemplo, podemos declarar que Empregado é a intersecção de Pessoa e ObjetoEmpregado, que pode ser uma classe com a restrição de cardinalidade mínima 1 para a propriedade temEmpregador).

3.4.2 OWL DL e OWL Full

A OWL DL e OWL Full têm o mesmo vocabulário (um superconjunto da OWL Lite), mas diferem nas restrições à linguagem. A OWL DL impõe a separação de tipos (uma classe não pode ser indivíduo ou propriedade, uma propriedade não pode ser um indivíduo ou classe). Portanto, restrições não podem ser aplicados aos elementos da própria linguagem. Já na OWL Full, essas flexibilidades são permitidas.

Outra diferença básica, é que na OWL DL temos a distinção de tipos de dados e objetos. As propriedades são ou ObjectProperties ou DatatypeProperties. Mais especificamente, DL se refere a *description logic*. A intenção da definição da OWL DL é deixá-la o mais próximo de uma lógica de descrição, para o aproveitamento de todas ferramentas e implementações relacionadas a esse campo de pesquisa.

O vocabulário de ambas se estende:

- `oneOf`: uma classe pode ser descrita como uma enumeração de indivíduos. Por exemplo, a classe diasDaSemana pode ser declarada como `oneOf` dos elementos Domingo, Segunda, Terça, Quarta, Quinta, Sexta, e Sábado.

- `hasValue`: pode-se especificar determinado indivíduo como valor de uma propriedade. Por exemplo, pode-se determinar uma classe C através da afirmação que a propriedade P têm valor X. Assim, qualquer indivíduo Y que estiver relacionado com X através de P pode ser inferido como pertencente a classe C.

- `disjointWith`: somente na OWL Full, pode-se declarar que duas classes são disjuntas.

- `unionOf, complementOf, intersectionOf`: pode-se declarar combinações booleanas arbitrárias de classes e restrições, na OWL Full e OWL DL.

- `minCardinality, maxCardinality, cardinality`: as restrições de cardinalidade são semelhantes ao OWL Lite, mas sem a limitação de 0 ou 1. A OWL DL e OWL Full permite qualquer valor não negativo.

- **classes complexas**: algumas construções da OWL Lite são limitadas em apenas um nome de classe (por exemplo em `subClassOf` e `equivalentClass`). Na OWL Full tais restrições não existem. Além disso, as classes podem ser usadas como instâncias na OWL Full, o que não pode ser feito na OWL DL, nem na OWL Lite.

Um exemplo de ontologia descrita em OWL pode ser bastante esclarecedor. A seguir o exemplo descreve um pequeno pedaço referente ao mesmo assunto dos exemplos anteriores.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:ns0="trabalho#" xmlns:owl="http://www.w3.org/??/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#">
    <owl:Ontology rdf:about="">
        <dc:title>Ontologia exemplo</dc:title>
        <dc:date>junho/2003</dc:date>
        <dc:creator>Marcio Carneiro</dc:creator>
        <dc:description></dc:description>
        <dc:subject>Trabalho</dc:subject>
        <owl:versionInfo>0.1</owl:versionInfo>
    </owl:Ontology>
    <owl:Class rdf:about="trabalho#Pessoa">
        <rdfs:label>Pessoa</rdfs:label>
    </owl:Class>
    <owl:Class rdf:about="trabalho#Empregador">
        <rdfs:label>Empregador</rdfs:label>
        <rdfs:subClassOf>
            <owl:Class rdf:about="trabalho#Pessoa"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:about="trabalho#Trabalho Manual">
        <rdfs:label>Trabalho Manual</rdfs:label>
        <rdfs:subClassOf>
            <owl:Class rdf:about="trabalho#Trabalho"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:about="trabalho#Trabalho">
        <rdfs:label>Trabalho</rdfs:label>
```

```
</owl:Class>
<owl:Class rdf:about="trabalho#Trabalhador">
    <rdfs:label>Trabalhador</rdfs:label>
    <rdfs:subClassOf>
        <owl:Class rdf:about="trabalho#Pessoa"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:about="trabalho#temEmpregador">
    <rdfs:label>temEmpregador</rdfs:label>
    <rdfs:domain>
        <owl:Class rdf:about="trabalho#Trabalhador"/>
    </rdfs:domain>
    <rdfs:range>
        <owl:Class rdf:about="trabalho#Empregador"/>
    </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="trabalho#temTrabalho">
    <rdfs:label>temTrabalho</rdfs:label>
    <rdfs:domain>
        <owl:Class rdf:about="trabalho#Trabalhador"/>
    </rdfs:domain>
    <rdfs:range>
        <owl:Class rdf:about="trabalho#Trabalho"/>
    </rdfs:range>
</owl:ObjectProperty>
<owl:FunctionalProperty rdf:about="trabalho#temTrabalho"/>
<owl:ObjectProperty rdf:about="trabalho#temEmpregado">
    <rdfs:label>temEmpregado</rdfs:label>
    <owl:inverseOf rdf:resource="trabalho#temEmpregador"/>
    <rdfs:domain>
        <owl:Class rdf:about="trabalho#Empregador"/>
    </rdfs:domain>
    <rdfs:range>
        <owl:Class rdf:about="trabalho#Trabalhador"/>
    </rdfs:range>
</owl:ObjectProperty>
<owl:Individual rdf:about="trabalho#SOLDADOR">
    <rdf:type>
        <owl:Class rdf:about="trabalho#Trabalho Manual"/>
    </rdf:type>
</owl:Individual>
```

```
    <owl:Individual rdf:about="trabalho#PEDRO">
        <rdf:type rdf:resource="http://www.w3.org/??/owl#Thing"/>
        <ns0:temTrabalho rdf:resource="trabalho#SOLDADOR"/>
    </owl:Individual>
</rdf:RDF>
```

3.5 Tesauros e criação de ontologias

A descrição de uma área de conhecimento numa linguagem de ontologia não é uma tarefa simples. Somente essa área de aplicação já se faz uma grande área de pesquisa e possibilidades.

Os tesauros, por serem mais simples e já conhecido dos usuários comuns — não necessariamente desenvolvedores e estudiosos da computação, podem ser um bom ponto de partida para a criação de uma ontologia, de maneira semelhante ao que fizemos nos exemplos. Entretanto, a tarefa de transformar um tesauro numa ontologia é bastante complexa por si só. Várias decisões têm que ser tomadas quanto a criação de propriedades. Muitas vezes, alguns conceitos do tesauros podem ser transformados em propriedades. Além disso, a decisão entre a transformação de um conceito da realidade em classe, propriedade ou mesmo instância da ontologia, é bastante delicada e não trivial (o artigo [WSWS01] traz uma boa discussão sobre o assunto).

Um ponto interessante é a possibilidade de se descrever um tesauro utilizando uma linguagem de ontologia. O relacionamento hierárquico já se encontra na hierarquia de classes (`subClassOf` equivalente ao TE). Podemos criar uma propriedade chamada `TR`, e restringir localmente a aplicação dela para cada conceito, somente aos termos que realmente são relacionados àquele conceito no tesauro (como o `allValuesFrom`). Os sinônimos podem ser descritos com o `equivalentClass`. Essa é uma boa maneira de se obter um caminho gradual para a criação de uma ontologia completa.

4 Aplicações

Após entendermos o que são ontologias e Web Semântica e olharmos as linguagens relacionadas a essa nova tecnologia, devemos agora verificar como essas idéias se encaixam praticamente na teia e sistemas semelhantes, e quais ferramentas sáo necessárias para a criação, manutenção e aplicação de ontologias aos sistemas atuais.

4.1 OnToKnowledge

O projeto OnToKnowledge tem como objetivo o desenvolvimento de um ambiente utilitário, além das linguagem e especificações. As principais ferramentas desenvolvidas pelo projeto são:

- **RDF*ferret***: uma ferramenta de busca por textos com consultas RDF. Basicamente se assemelha a um sistema de busca comum, como os conhecidos da Web, mas pode-se fazer refinamentos das buscas com as classes trazidas das descrições ontológicas dos documentos.
- **OntoShare**: ferramenta de compartilhamento de conhecimento e melhores práticas baseada em ontologias.
- **OntoEdit**: ambiente de desenvolvimento e edição de ontologias, permitindo a exportação das ontologias em diversas linguagens, além da inserção de diversos *plug-ins*.
- **Sesame**: sem dúvida o mais interessante, tanto por ser base para outras ferramentas, quanto por sua disponibilização com código aberto. Trata-se de uma camada de persistência e inferência para RDF/RDFS (e conseqüentemente para outras linguagens baseadas nessa sintaxe). Veja secção 4.3.
- **CORPORUM**: conjunto de duas ferramentas semelhantes (**OntoExtract** e **OntoWrapper**) para extração de ontologias a partir de linguagens livres e para encapsulamento de textos livres no formato de ontologias.

4.2 Editores de ontologia

Infelizmente as ferramentas do OnToKnowledge, exceto pelo Sesame, são proprietárias e não compartilhadas com a comunidade científica, pelo menos no momento. Uma ferramenta essencial para iniciar a empreitada pelo desenvolvimento de ontologias é um editor gráfico que esconda a horrível sintaxe RDF/RDFS/XML (é consenso que a possibilidade de ler o XML é um efeito colareral, pois pode até facilitar a depuração de códigos e arquivos de configuração, mas também induz a sugestão de se editar diretamente esses arquivos, tarefa nada agradável).

Duas ferramentas de desenvolvimento aberto são bastante interessantes:

- **OILed**: desenvolvido na Universidade de Manchester, e sob licença GPL. Inicialmente projetado para ontologias descritas em OIL, mas pode exportar os dados para outros formatos. Permite conexão com o FaCT, para realização de inferências e deduções. Veja http://oiled.man.ac.uk/.

- **Protégé-2000**: desenvolvido em Stanford, no departamento de informática médica da faculdade de medicina. Tem a DARPA como patrocinador, entre outros institutos de pesquisa e fomento. Provê uma interface mais intuitiva que o OILed. Veja http://protege.stanford.edu/.

4.3 Sesame

Uma parte das mais importantes, senão a mais importante, de um ambiente de aplicação de ontologias e Web Semântica, é uma camada de persistência e inferência. Com uma ferramenta bem desenvolvida para essa tarefa, podemos pensar em escalabilidade e desempenho para a aplicação de ontologias à Web.

O Sesame (http://sesame.aidministrator.nl/), desenvolvido inicialmente pelo projeto OnToKnowledge, mas em conjunto com as empresas Aidministrator e OntoText, se trata de uma boa proposta de camada de persistência e inferência para ontologias.

A princípio, apenas RDF e RDFS são utilizados para armazenagem e consulta. Entretanto, qualquer outra linguagem se utiliza dessa sintaxe, o que torna fácil a adaptação do sistema à elas, seja DAML+OIL, seja OWL. O código é Java, permitindo fácil migração para a maioria dos sistemas operacionais (o projeto é bem modularizado, com intenção de ser utilizado em PDAs também). A licença de desenvolvimento e uso é o LGPL.

A arquitetura do sistema é modular. Há um módulo principal para o encapsulamento do armazenamento, chamado de RAL (*Repository Abstract Layer*), módulos funcionais para extração, segurança, consultas e administração dos dados, e separadamente há as interfaces para acesso a esses módulos. Esses módulos funcionais são:

- módulo para administração de dados (adição e remoção).
- módulo de exportação, para extração de dados em diversos formatos de documentos.
- um sistema de consultas RQL, para consultas nessa linguagem.
- um sistema de consultas RDQL, para consultas também nessa linguagem (desenvolvida por um grupo na HP).
- módulo de segurança
- módulo de versão para versões de ontologias.

4.3.1 Camada de repositório

A camada de repositório foi projetada para realizar o armazenamento do conteúdo RDF e também as inferências sobre os dados armazenados. A partir de uma interface independete e pré-definida, podemos apenas alterar o módulo de armazenamento, permitindo a troca de banco de dados ou mesmo de meio (memória, sistema de arquivos, etc), transparentemente.

Outra característica da RAL é que ela pode ser empilhada, isto é, pode-se realizar operações em uma camada, e repassar a requisição para outra camada de armazenamento.

Assim, pode-se criar camadas que realizam consultas em outros repositórios de dados, para integração com os dados do RDF, entre outras possibilidades interessantes.

5 Conclusão

A Web Semântica parece ser uma idealização muito poderosa e útil para a vida das pessoas. O esforço corrente da academia e dos órgãos como W3C é evidente, e bastante sincronizado no momento, o que sugere um rápido desenvolvimento e adoção pelo mercado. Entretanto, ainda estamos no começo, com pouco mais de 4 anos de inicio desse movimento ao mundo semântico.

Mesmo com o trabalho dos desenvolvedores e especificadores, há que se esperar o bom aceitamento do mercado e usuários finais. Por mais que um usuário não compreenda nada por trás da usa interface, se bem feita e realmente útil, realmente inovadora, ele certamenta a adotará, e "comprará" a idéia da Web Semântica.

Uma dos maiores problemas para a boa difusão dessa nova tecnologia é o esforço que é exigido para a criação de ontologias. Provavelmente haverá muitas ontologias divergentes sobre o mesmo assunto. Entretanto, muitas vezes essas ontologias serão locais, e não necessariamente deverão seguir padrões mundiais. Logicamente, para a comunicação global de agentes humanos e computacionais de acordo com uma ontologia, é necessário um acordo comum por uma única versão da mesma, ou ao menos uma boa maneira de mapeamento entre as ontologias diferentes, mas semelhantes em conteúdo e semântica.

Há vários campos ainda de pesquisa e desenvolvimento. A melhoria das ferramentas e camadas de armazenamento. Pesquisas no processo de desenvolvimento e extração de ontologias de maneira automática ou ao menos semi-automática.

O assunto desse texto como um todo se encaixa no meu projeto de mestrado, cuja intenção é o desenvolvimento de alguma aplicação baseada em ontologia. A princípio, partiremos do princípio de termos uma ontologia, mesmo que simples e pequenas, e um objetivo específico

da aplicação da ontologia dentro de um sistema. Duas opções surgem:

- Utilização de ontologia para criação de contatos (relacionamentos direcionados) entre objetos de um sistema de cadastro de pessoas. Esses contatos teriam categoria e outros dados, e podem se encaixar bem numa descrição ontológica.

- Sistema de controle de acesso baseado em ontologias. Certamente mais interessante que o primeiro, mas em contrapartida menos certo de ser funcional. Controle de acesso geralmente se aplica a recursos e sempre está ligado a ações, como *acessar*, *alterar* ou *ler*. Com uma boa modelagem em ontologias, podemos tirar proveito das possibilidades de inferência e consultas complexas (nesse caso o desejado deve ser algo como verificação da veracidade das afirmações) que uma camada como o Sesame pode oferecer.

Todo o assunto aqui pesquisado deve ser base para uma qualificação, assim que o problema a ser resolvido (isto é, a aplicação a ser desenvolvida) estiver bem definido.

Referências

[BHS03] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In Dieter Hutter and Werner Stephan, editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer, 2003. To appear.

[BKvH02] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF schema. In I. Horrocks and J. Hendler, editors, *Proceedings of the First Internation Semantic Web Conference*, number 2342 in Lecture Notes in Computer Science, pages 54–68. Springer Verlag, July 2002.

[BL00] Tim Berners-Lee. *Weaving the Web*. Texere Publishing, NA, 2000. ISBN 1-58-799018-0.

[DCvH+02] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, and D. McGuinness. Web ontology language (OWL) reference version 1.0. *World Wide Web Consortium Working Draft*, November 2002.

[FSvH02] Christiaan Fluit, Marta Sabou, and Frank van Harmelen. Ontology-based information visualisation. In Vladimir Geroimenko, editor, *Visualising the Semantic Web*. Springer Verlag, 2002.

[GG95] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, 1995.

[Hay03] Patrick Hayes. RDF semantics. *World Wide Web Consortium Working Draft*, January 2003.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[PSHvH02] P. Patel-Schneider, I. Horrocks, and F. van Harmelen. Reviewing the design of DAML+OIL: An ontology language for the semantic web. In R. Dechter, M. Kearns, and R. Sutton, editors, *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, July 2002.

[SAB+03] Y. Sure, H. Akkermans, J. Broekstra, J. Davies, Y. Ding, A. Duke, R. Engels, D. Fensel, I. Horrocks, V. Iosif, A. Kampman, A. Kiryakov, M. Klein, Th. Lau, D. Ognyanov, U. Reimer, K. Simov, R. Studer, J. van der Meer, and F van Harmelen. On-to-knowledge: Semantic web enabled knowledge management. In N. Zhong, J. Liu, and Y. Yao, editors, *Web Intelligence*, pages 277–300. Springer-Verlag, 2003.

[vH02a] F. van Harmelen. How the semantic web will change KR: challenges and opportunities for a new research agenda. *The Knowledge Engineering Review*, 17(1), 2002.

[vH02b] Frank van Harmelen. The complexity of the web ontology language. *IEEE Intelligent Systems*, 17(March/April), 2002.

[WSWS01] B. J. Wielinga, A. Th. Schreiber, J. Wielemaker, and J. A. C. Sandberg. From thesaurus to ontology. *Internation Conference on Knowledge Capture, Victoria, Canada*, October 2001.