Ontologias - OWL (*Web Ontology Language*)

Júnio César de Lima *Cedric Luiz de Carvalho*

Technical Report - RT-INF_004-05 - Relatório Técnico
June - 2005 - Junho

Instituto de Informática
Universidade Federal de Goiás
www.inf.ufg.br

Ontologias - OWL (*Web Ontology Language*)

Júnio César de Lima * junio@inf.ufg.br

Cedric L. de Carvalho † cedric@inf.ufg.br

Abstract. *An ontology defines the terms used to describe and represent a domain, that is, an ontology is a description of concepts and relationships that can be used by people or software agents that want to share information within a domain. For these characteristics, ontologies are one of the key technologies in the emerging Semantic Web. The main ontology language for publishing and sharing ontologies on the Web is OWL - Web Ontology Language, a W3C recommendation. The main features of an ontology and of the OWL language are discussed in this text.*

Keywords: Ontology, OWL, RDF Schema and Semantic Web.

Resumo. *Uma ontologia define os termos usados para descrever e representar um domínio, isto é, é uma descrição de conceitos e relacionamentos entre eles que pode ser usada por pessoas ou agentes de software para compartilhar informações dentro deste domínio. Por estas características, a ontologia é uma das tecnologias chaves para a implementação da Web Semântica. OWL é uma linguagem de marcação semântica para publicação e compartilhamento de ontologias, projetada para descrever classes e relacionamentos entre elas, que é atualmente recomendada pelo W3C. As principais características de uma ontologia e da linguagem OWL são discutidas neste texto.*

Palavras-Chave: Ontologia, OWL, RDF *Schema* e *Web* Semântica.

1 Introdução

A *World Wide Web*, ou simplesmente *Web*, é uma excelente fonte de pesquisa em razão da quantidade de informações que ela armazena. Porém, estas informações são disponibilizadas sem nenhum critério de organização ou padronização, dificultando o processo de busca e recuperação das mesmas por meio de mecanismos automatizados.

O conteúdo das informações é disponibilizado principalmente para o entendimento humano, havendo, portanto, uma grande dificuldade na execução dos serviços de recuperação processados por máquinas. Neste contexto surge a *Web* Semântica. Segundo Tim Berners-Lee, um dos seus idealizadores, a *Web* Semântica, a partir do uso intensivo de metadados, visa construir uma rede que seja capaz de reconhecer o significado dos documentos e, por meio de um processamento via máquina, inferir novos conhecimentos [10].

*Mestrando em Ciência da Computação - GEApIS/INF/UFG

†Orientador - GEApIS/INF/UFG

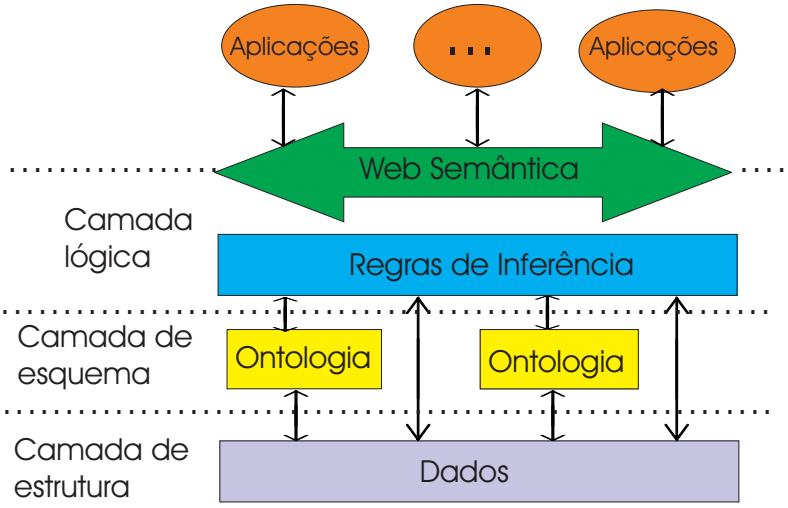

Figura 1: Uma das possíveis arquitetura da *Web* Semântica [10].

Os princípios da *Web* Semântica são implementados em camadas de tecnologias e padrões *Web*. Segundo Berners-Lee, Hendler e Lassila [10], uma das possíveis arquiteturas da *Web* Semântica pode ser dividida em três grandes camadas, como mostra a Figura 1. A camada de estrutura, é responsável por estruturar os dados e definir seu significado. Ela é implementada utilizando as tecnologias XML (*eXtensible Markup Language*) [2], RDF (*Resource Description Framework*) e RDF *Schema* [11]. A camada de esquema, é composta pelas ontologias que possibilitam a representação de conceitos através de uma taxonomia e um conjunto de regras de inferência. A taxonomia define as classes de objetos e as relações que se estabelecem entre elas. A camada lógica, é responsável por definir mecanismos para se fazer inferência sobre os dados.

Uma ontologia define um conjunto comum de termos que são usados para descrever e representar um domínio, como medicina, biblioteca etc.. Estas definições de conceitos básicos de domínios e seus relacionamentos são processáveis por computadores e são expressas em linguagens baseadas em lógica.

As ontologias podem ser usadas, dentre outras coisas, com o propósito de melhorar a exatidão de buscas na *Web*: um programa de busca baseado em ontologias poderá ser capaz de recuperar somente as páginas relevantes para o usuário.

Este texto tem por objetivo apresentar as ontologias no contexto da *Web* Semântica. Ele está organizado como segue: na Seção 2, se discute o significado da palavra ontologia e a definição de ontologia no contexto da Inteligência Artificial. Na Seção 3, é apresentada a linguagem de criação de ontologias para *Web*, chamada de OWL. Nesta Seção serão apresentadas as principais características da linguagem, entre elas a criação de classes, indivíduos e relacionamento entre eles. Finalmente, na Seção 4, são apresentadas as considerações finais.

2 Ontologias

O termo ontologia tem origem na filosofia grega [1]. Na filosofia, uma ontologia é uma teoria sobre a existência da natureza, sobre que tipos de coisas existem ou o que se dizer sobre o mundo. Segundo Ferreira [6], ontologia é uma "parte da filosofia que trata do ser enquanto

[1] Aristóteles 384-322 a.C.

ser, isto é, do ser concebido como tendo uma natureza comum que é inerente a todos e a cada um dos seres".

Os pesquisadores da Inteligência Artificial e da *Web* converteram este termo para o seu próprio jargão, sendo que, para eles, o que "existe" é aquilo que pode ser representado, ou seja, tudo que existe pode ser representado formalmente.

Segundo Gruber [7], uma ontologia é uma especificação explícita e formal de uma conceitualização compartilhada. Ou seja, uma ontologia conceitua um modelo abstrato de algum fenômeno do mundo em um conhecimento consensual, isto é, compartilhado por todos. Além disso, os conceitos, as propriedades, as funções, os axiomas devem ser especificados explicitamente e serem manipuláveis por computador.

Há várias definições na literatura. Entretanto, a relevância das ontologias para a *Web* deve ser tratada sobre três aspectos:

- Identificação de contexto: por exemplo, quando dois agentes de *softwares* trocam informações sobre `braço` é preciso assegurar em que contexto este termo está sendo referenciado. Isto é, um agente pode se referir ao termo `braço` no contexto de medicina, por exemplo, logo **braço** é um membro do corpo humano. O outro agente pode se referir ao termo **braço** no contexto de móveis, logo, ele está se referindo a um `braço` de sofá, por exemplo.

- Fornecimento de definições compartilhadas: por exemplo, se uma aplicação **X** possui uma ontologia que define uma loja que vende `carros` e uma aplicação **Y** possui outra ontologia que define uma loja que vende `veículos`, logo se percebe o problema caso ambas queiram intercambiar informações. Este problema é bastante natural para o ser humano e difícil para uma máquina. Este tipo de confusão pode ser resolvido se as ontologias proverem relações de equivalência, ou seja, se uma ou as duas ontologias possuirem informações dizendo que `carro` da aplicação **X** é equivalente ao `veículo` da aplicação **Y**.

- Reuso de ontologias: por exemplo, se uma determinada pessoa **X** já tem construída uma ontologia (sobre medicina, por exemplo) que define um conjunto de termos que outra pessoa **Y** também necessita, então não há porque a pessoa **Y** criar outra ontologia, isto é, refazer o trabalho que já foi feito. Ela pode simplesmente fazer uso da ontologia já criada.

Quando as ontologias são construídas levando-se em consideração estes aspectos, elas permitem o compartilhamento de informações entre diferentes agentes de *software*. Assim, podem ser usadas em buscas baseadas em semântica, facilitando a comunicação entre estes agentes e capacitando a integração de dados de bases de dados heterogêneas.

O uso de ontologias, no contexto da *Web* Semântica, requer uma linguagem de ontologia compatível com a *Web*, com uma sintaxe e uma semântica bem definida, suporte ao raciocínio eficiente e expressiva. As linguagens de ontologia para *Web* são, geralmente, expressas em uma linguagem lógica, a lógica descritiva, garantindo as distinções entre as classes, propriedades e relações, evitando ambigüidades.

Diferentes linguagens de ontologias provêem diferentes facilidades. Dentre estas linguagens pode-se citar: a SHOE (*Simple HTML Ontology Extensions*) [15], XOL (*Ontology Exchange Language*) [18], OIL (*Ontology Inference Layer*) [14] e DAML (*DARPA Agent Markup Language*) [4]. Estas duas últimas foram combinadas e formaram a DAML+OIL [5]. Entretanto, a mais recente linguagem de ontologia desenvolvida é a OWL *Web Ontology Language* [8].

A OWL é uma revisão da linguagem DAML+OIL. Ela possui mais facilidades para expressar significados e semânticas do que XML, RDF e RDF *Schema*, embora seja baseada em RDF e RDF *Schema* e utilize a sintaxe XML. A OWL foi projetada para ser usada por aplicações que necessitem processar o conteúdo de informações, ao invés de somente apresentar a visualização destas informações.

O W3C [17] recomenda que as pessoas que queiram construir ontologias utilizem a linguagem OWL, com isso, se espera que esta linguagem se torne um padrão. Na Seção 3, é introduzida a linguagem OWL.

3 OWL - *Web Ontology Language*

A OWL é uma linguagem para definição e instanciação de ontologias *Web*. Uma ontologia OWL pode formalizar um domínio, definindo classes e propriedades destas classes, definir indivíduos e afirmações sobre eles e, usando-se a semântica formal OWL, especificar como derivar conseqüências lógicas, isto é, fatos que não estão presentes na ontologia, mas são vinculados pela semântica [16].

Segundo Harmelen e McGuinness [8], a OWL possui três sub-linguagens incrementais projetadas para serem usadas por diferentes comunidades de implementadores e usuários:

- OWL *Lite*: é uma sub-linguagen da OWL DL que usa somente algumas características da linguagem OWL e possui mais limitações do que OWL DL ou OWL *Full*.
- OWL DL: é usada por usuários que queiram o máximo de expressividade, com completude (todas as conclusões são garantidas serem computáveis) e decidibilidade (todas as computações terminarão em um tempo finito) computacional. Ela inclui todas as construções da linguagem OWL, mas estas construções somente podem ser usadas sob certas restrições. A sigla DL possui correspondência com a lógica descritiva (*description logics*), uma área de pesquisa que estuda um fragmento particular da lógica de primeira ordem.
- OWL *Full*: é usada por usuários que queiram o máximo de expressividade e independência sintática de RDF, sem nenhuma garantia computacional. A OWL *Full* e a OWL DL suportam o mesmo conjunto de construções da linguagem OWL, embora com restrições um pouco diferentes. Enquanto a OWL DL impõe restrições sobre o uso de RDF e requer disjunção de classes, propriedades, indivíduos e valores de dados, a OWL *Full* permite misturar OWL com RDF *Schema* e não requer a disjunção de classes, propriedades, indivíduos e valores de dados. Isto é, uma classe pode ser ao mesmo tempo uma classe e um indivíduo.

Cada uma destas sub-linguagens é uma extensão de sua predecessora, ou seja, cada ontologia válida em OWL *Lite* é uma ontologia válida em OWL DL, esta por sua vez é uma ontologia válida em OWL *Full* [8]. Vale lembrar que esta relação não é simétrica, isto é, uma ontologia válida em OWL *Full* pode ou não ser uma ontologia válida em OWL DL. Além disso, todo documento OWL (*Lite*, DL ou *Full*) é um documento RDF e todo documento RDF é um documento OWL *Full*, entretanto, somente alguns documentos RDF são documentos OWL *Lite* ou DL válidos.

Segundo Harmelen e McGuinness [8], a escolha de qual sub-linguagem OWL os desenvolvedores de ontologias devem usar vai depender das necessidades da ontologia. A escolha entre OWL *Lite* e OWL DL dependerá da necessidade das propriedades computacionais de

OWL *Lite* ou das construções mais expressivas providas pela OWL DL. A escolha entre OWL DL e OWL *Full* dependerá da necessidade de expressividade, decidibilidade e completude computacional da OWL DL ou da expressividade e das facilidades do meta-modelo RDF *Schema* sem a previsibilidade computacional de OWL *Full*.

Esta Seção é baseada no documento OWL *Web Ontology Language Guide* [16], que é uma recomendação do W3C. Ela é organizada como segue. Na Subseção 3.1, se discute a estrutura para definição de uma ontologia. Na Subseção 3.2, são apresentados os elementos básicos da OWL, ou seja, as classes, indivíduos e propriedades simples. Na Subseção 3.3, são apresentadas as características das propriedades. Na Subseção 3.4, se discute as restrições que podem ser impostas sobre as propriedades. Na Subseção 3.5, se discute o mapeamento de ontologias, ou seja, como se pode relacionar ontologias de fontes diferentes. Na Subseção 3.6, são apresentadas as classes complexas.

3.1 Estrutura das ontologias

A OWL foi projetada para prover uma linguagem de ontologia que pudesse ser usada para descrever, de um modo natural, classes e relacionamentos entre elas em documentos e aplicações *Web*. Os termos usados em uma ontologia devem ser escritos de uma maneira que eles possam ser interpretados sem ambigüidade e usados por agentes de *software*. Por isso, antes de escrever os termos, deve-se indicar qual vocabulário está sendo empregado.

Para indicar qual vocabulário está sendo usado, por recomendação de McGuinness, Smith e Welty [16], deve-se ter um componente inicial de uma ontologia incluindo um conjunto de *namespaces* XML[2] [3] contidos na *tag* de início `rdf:RDF`, como mostra o Código 1.

Código 1 – Declaração de *namespace* em OWL.

```
1  <rdf:RDF
2   xmlns = "http://www.mindswap.org/2003/owl/swint/terrorism#"
3   xmlns:base = "http://www.mindswap.org/2003/owl/swint
4                                                  /terrorism#"
5   xmlns:terr = "http://www.mindswap.org/2003/owl/swint
6                                                  /terrorism#"
7   xmlns:owl = "http://www.w3.org/2002/07/owl#"
8   xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
9   xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
10  xmlns:xsd = "http://www.w3.org/2001/XMLSchema#"
11 >
```

Esta declaração identifica os *namespaces* associados com a ontologia `terrorism`. Na linha 2, se faz uso de um *namespace* padrão para a ontologia corrente. Na linha 3, se identifica um URI base para o documento. Na linha 5, se identifica o *namespace* da ontologia corrente com o prefixo `terr:`. Na linha 7, se identifica um *namespace* para o vocabulário OWL. Como a OWL é definida sobre RDF, RDF *Schema* e dados tipados da XML *Schema*, nas linhas 8, 9 e 10, respectivamente, são definidos *namespaces* para RDF (`rdf:`), RDF *Schema* (`rdfs:`) e XML *Schema* (`xsd:`).

Ainda seguindo a recomendação de McGuinness, Smith e Welty [16], é interessante o uso de entidadades XML, como mostra o Código 2.

[2] *Namespaces* são usados para prevenir colisões de nomes, isto é, para identificar unicamente os elementos. Eles identificam uma parte da *Web* (espaço) que atua como um qualificador para um conjunto específico de nomes.

Código 2 – Declaração de entidades XML em OWL.

```
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY terr "http://www.mindswap.org/2003/owl/swint
                                                /terrorism#">
  <!ENTITY pers "http://www.mindswap.org/2003/owl/swint
                                                /person#">
]>
```

Com a declaração desta entidade, pode-se substituir a declaração do *namespace* `xmlns:terr = "http://www.mindswap.org/2003/owl/swint/terrorism#"` por `xmlns:terr="&terr;"`. Além disso, quando em uma ontologia (na ontologia `terrorism`, por exemplo) se precisar referenciar o nome de uma classe chamada `Event`, por exemplo, ao invés de se utilizar a forma expandida, `"http://www.mindswap.org/2003/owl/swint/terrorism#Event"`, pode ser usada a forma mais compacta: `"&terr;Event"`.

As ontologias `terrorism` e `person` foram obtidas da **Universidade de *Mindswap*** em [13] e [12], respectivamente. Valer ressaltar que as construções apresentadas neste texto não são cópias fiéis das construções apresentadas pela Universidade de *Mindswap*, uma vez que estas ontologias não cobrem todas as construções propostas pela OWL e que o objetivo deste texto é mostrar os emprego de todas as construções OWL para criação de ontologias. Na Subseção 3.1.1, se discute a inclusão de cabeçalho na definição de uma ontologia.

3.1.1 Cabeçalho das ontologias

Em um documento descrevendo uma ontologia é interessante fornecer informações sobre ela própria. Como uma ontologia é um recurso, ela pode ser descrita usando-se as propriedades da OWL e *namespaces* XML. Estas informações devem ser agrupadas dentro da *tag* `owl:Ontology`, como mostra o Código 3.

Código 3 – Cabeçalho de uma ontologia.

```
<owl:Ontology rdf:about="">
 <owl:versionInfo> 05/01/2005 20:41:52 </owl:versionInfo>
 <owl:priorVersion
    rdf:resource="http://www.mindswap.org/2002/owl/swint
                                              /terrorism#"/>
 <rdfs:comment>This file specify a OWL ontology</rdfs:comment>
 <owl:imports rdf:resource="http://www.mindswap.org/2003
                                          /owl/swint/person#"/>
 <rdfs:label> Terrorism Ontology </rdfs:label>
  ...
 </owl:Ontology>
  ...
</rdf:RDF>
```

O elemento `owl:Ontology` é usado para fornecer informações sobre o documento OWL, ou seja sobre a ontologia sendo descrita. Assim, a linha 1 do Código 3 indica que o bloco descreve a onologia em questão, ou seja, indica que a ontologia é identificada pelo URI base do elemento `owl:Ontology`.[3]

O elemento `owl:versionInfo` indica a versão da ontologia. O elemento `owl:priorVersion` identifica a versão anterior da ontologia que está sendo criada. Este elemento permite fazer controle de versões de ontologias.

O elemento `rdfs:comment` é usado para inserir comentários sobre a ontologia.

A declaração `owl:imports` referencia outra ontologia OWL contendo definições usadas como parte do significado da ontologia que está sendo criada. A declaração `owl:imports` recebe um argumento simples, que consiste de uma referência URI, identificado pelo atributo `rdf:resource`. Além disso, as declarações `owl:imports` são trantisivas, isto é, se a ontologia A importa B e B importa C, então A importa B e C.

O elemento `rdfs:label` permite incluir um rótulo em linguagem natural para a ontologia. O cabeçalho é fechado usando-se a *tag* de fim `</owl:Ontology>`. Depois de definir o cabeçalho, são criadas as definições da ontologia que são fechados pela *tag* de fim `</rdf:RDF>`. Na Subseção 3.2, são apresentados os elementos básicos da linguagem OWL, isto é, a definição de classes, indivíduos (que são instâncias de classes) e propriedades.

3.2 Elementos básicos

Os elementos básicos para construção de uma ontologia OWL são as classes, as instâncias (também chamadas de indivíduos) das classes e os relacionamentos (as propriedades) entre estas instâncias. Nas próximas Subseções serão apresentados estes elementos.

3.2.1 Classes e Indivíduos

As classes provêem um mecanismo de abstração para agrupar recursos com características similares, ou seja, uma classe define um grupo de indivíduos que compartilham algumas propriedades. O conjunto de indivíduos que estão associados a uma classe, é chamado de **extensão de classe** [1]. Os indivíduos em uma extensão de classe são chamados de **instâncias** da classe.

Cada indivíduo na OWL é membro da classe `owl:Thing`. Deste modo, ela é superclasse de todas as classes OWL definidas pelos usuários. Além disso, existe a classe `owl:Nothing` (não possui instâncias) que é uma subclasse de todas as classes OWL. Uma classe é sintaticamente representada como uma instância nomeada da `owl:Class`, que é uma subclasse da `rdfs:Class`. Uma classe em OWL pode ser definida da seguinte maneira:

```
<owl:Class rdf:ID="Organization">
```

A sintaxe `rdf:ID="Organization"` é usada para nomear a classe. Segundo McGuinness, Smith e Welty [16], esta classe, dentro do documento em que ela foi definida, pode ser referenciada usando-se a expressão `#Organization`. Outras ontologias podem referenciar esta classe usando a forma completa `"http://www.mindswap.org/2003/owl/swint/terrorism#Organization"`. Entretanto, se forem definidos *namespaces* e entidades, como foi recomendado na Subseção 3.1,

[3] No caso deste exemplo, o URI `"http://www.mindswap.org/2003/owl/swint/terrorism\#"` foi definido como base na Subseção 3.1 (linha 3 do Código 1).

pode-se se referir à classe `Organization` usando-se a *tag* XML `terr:Organization` ou o valor do atributo `&terr;Organization`.

Hierarquias de classes podem ser criadas usando-se a construção `rdfs:subClasseOf`. O Código 4 mostra uma hierarquia de classes que usa esta construção.

Código 4 – Hierarquia de classes.

```
1 <owl:Class rdf:ID="TerroristOrganization">
2   <rdfs:subClassOf rdf:resource="#Organization"/>
3   ...
4 </owl:Class>
```

Esta declaração diz se a classe `TerroristOrganization` é definida como uma subclasse da classe `Organization`, então, o conjunto de indivíduos da classe `TerroristOrganization` deve ser um sub-conjunto do conjunto de indivíduos da classe `Organization`.

Os indivíduos, que são instâncias de classes, são definidos com fatos. Um fato é uma declaração que é sempre verdade em um dado domínio. Indivíduos podem ser relacionados usando-se as propriedades da OWL, que são discutidas na Seção 3.2.2. Por exemplo, indivíduos podem ser declarados na forma:

```
<TerroristOrganization rdf:ID="Al_Qaeda"/>
```

Esta construção diz que o indivíduo `Al_Qaeda` é uma instância da classe `TerroristOrganization`. Além disso, esta construção declara um fato sobre a ontologia `Terrorism`, ou seja, `"Al Qaeda é uma organização terrorista"`.

Segundo Bechhofer et al. [1], uma classe tem um significado intensional que é relacionado mas não é igual à extensão de classe, ou seja, duas classes podem ter mesma extensão de classe e serem classes diferentes. Além disso, em OWL *Lite* e OWL DL, classes e indivíduos formam um domínio disjunto, diferentemente de OWL *Full*, que permite que uma classe seja instância de outra classe.

3.2.2 Propriedades

As propriedades, que são relações binárias, podem ser usadas para estabelecer relacionamentos entre indivíduos ou entre indivíduos e valores de dados. Estes relacionamentos permitem afirmar fatos gerais sobre os membros das classes e podem também especificar fatos sobre indivíduos [16]. Segundo McGuinness, Smith e Welty [16], a OWL distigue entre duas categorias principais de propriedades:

- Propriedades de dados tipados (*datatype properties*): relação entre indivíduos e valores de dados.
- Propriedades de objetos (*object properties*): relação entre indivíduos.

Uma propriedade de objeto é definida como instância da classe `owl:ObjectProperty`. Uma propriedade de dado tipado é definida como uma instância da classe `owl:DatatypeProperty`. Além disso, ambas são subclasses da classe RDF `rdf:Property`. Sendo assim, as propriedades do RDF *Schema* são herdadas, ou seja, `rdfs:subPropertyOf`, `rdfs:domain` e `rdfs:range`. Uma propriedade simples em OWL pode ser definida da seguinte maneira:

```
<owl:ObjectProperty rdf:ID="affiliate"/>
```

Esta declaração define uma propriedade com a restrição que seus valores devem ser indivíduos. Existem várias outras restrições que podem ser impostas, entre elas seu domínio (`domain`) e seus valores (`range`) [4]. Uma propriedade usando estas duas restrições é mostrada no Código 5.

Código 5 – Uso de `domain` e `range`.

```
1 <owl:ObjectProperty rdf:ID="government">
2   <rdfs:label>Governo</rdfs:label>
3   <rdfs:subPropertyOf rdf:resource="#affiliate"/>
4   <rdfs:domain rdf:resource="#Country"/>
5   <rdfs:range rdf:resource="#Government"/>
6 </owl:ObjectProperty>
```

A propriedade `government` tem como domínio (`domain`) a classe `Country`, ou seja, esta propriedade deve ser usada somente por indivíduos que são instâncias da classe `Country`. Seu valor (`range`) é a classe `Government`, ou seja, os valores desta propriedade devem pertencer à extensão de classe da classe `Government`.

O Código 6 mostra o uso de uma propriedade do tipo `DatatypeProperty`.

Código 6 – Uso de uma propriedade do tipo `owl:DatatypeProperty`.

```
1 <owl:DatatypeProperty rdf:ID="emailAddress">
2     <rdfs:label>Endereço de e-mail</rdfs:label>
3     <rdfs:domain rdf:resource="#Agent"/>
4     <rdfs:range rdf:resource="&xsd;string"/>
5 </owl:DatatypeProperty>
```

A propriedade `emailAdress` relaciona `Agent` à uma *string*, ou seja, um agente tem um endereço de email. O valor do recurso, `rdf:resource="&xsd;string"`, é um dado tipado definido na XML *Schema*. A referência para este e outros dados tipados estão em [16].

Hierarquias de propriedades podem ser criadas na OWL usando-se a declaração `rdfs:subPropertyOf`, ou seja, ela define que uma propriedade é uma sub-propriedade de alguma outra propriedade. O Código 7 mostra a declaração de uma sub-propriedade.

Código 7 Declaração de uma sub-propriedade.

```
1 <owl:ObjectProperty rdf:ID="member">
2   <rdfs:label>Membro</rdfs:label>
3   <rdfs:subPropertoyOf rdf:resource="#affiliate"/>
4   <rdfs:domain rdf:resource="#Organization"/>
5   <rdfs:range rdf:resource="#Agent"/>
6 </owl:ObjectProperty>
```

A propriedade `member` é uma sub-propriedade da propriedade `affiliate`, que só pode ser usada por indivíduos da classe `Organization` e o seu valor (`range`) deve ser os indivíduos da classe `Agent`. Como a propriedade `member` é uma sub-propriedade da propriedade

[4] Restrições herdadas do RDF *Schema* [11].

affiliate, qualquer classe com a propriedade member com o valor Al_Zarqawi também tem uma propriedade affiliate com o valor Al_Zarqawi.

Também se pode declarar propriedades sobre os indivíduos. As propriedades sobre indivíduos são ditas de fatos sobre os indivíduos. O Código 8 mostra uma declaração de propriedades sobre indivíduos.

Código 8 – Declaração de propriedade para indivíduos.

```
1 <Agent rdf:ID="Al_Zarqawi">
2     <memberOf rdf:resource="#Al_Qaeda">
3     <subordinateOf rdf:resource="#Bin_Laden">
4     <emailAddress rdf:datatype="&xsd;string">
5                         zarqawi@alqaeda.com </emailAddress>
6 </Agent>
```

Este trecho declara o indivíduo Al_Zarqawi e também três fatos sobre ele: 1) usando a propriedade memberOf, diz que o agente Al Zarqawi é membro da Al Qaeda; 2) usando a propriedade subordinateOf, diz que Bin Laden é o superior do agente Al Zarqawi; e 3) usando a propriedade emailAddress, diz que o agente Al Zarqawi possui a *string* zarqawi@alqaeda.com como seu endereço de *email*. Na Subseção 3.3, se discute as características das propriedades.

3.3 Características das Propriedades

É possível especificar características das propriedades [16]. A OWL suporta os seguintes mecanismos: owl:TransitiveProperty, owl:SymmetricProperty, owl:FunctionalProperty, owl:InverseFunctionalProperty e owl:inverseOf. Estes mecanismos são discutidos nas próximas Subseções.

3.3.1 *TransitiveProperty*

As propriedades podem ser ditas **transitivas**. Se a propriedade subordinate é dita ser transitiva e se é sabido que Mullah_Krekar é subordinado de Al_Zarqawi ((Mullah_Krekar, Al_Zarqawi) é uma instância da propriedade subordinate) e que Al_Zarqawi é subordinado de Bin_Laden ((Al_Zarqawi, Bin_Laden) é uma instância da propriedade subordinate), então se pode deduzir que Mullah_Krekar é subordinado de Bin_Laden ((Mullah_Krekar, Bin_Laden) é uma instância da propriedade subordinate). O Código 9, mostra este exemplo na sintaxe OWL.

Código 9 – Declaração de uma propriedade transitiva, sintaxe 1.

```
<owl:ObjectProperty rdf:ID="subordinate">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Agent"/>
  <rdfs:range rdf:resource="#Agent"/>
</owl:ObjectProperty>

<Agent rdf:ID="Mullah_Krekar">
  <subordinate rdf:resource="#Al_Zarqawi"/>
</Agent>

<Agent rdf:ID="Al_Zarqawi">
  <subordinate rdf:resource="#Bin_Laden"/>
</Agent>
```

Sendo que `Mullah_Krekar` é subordinado de `Al_Zarqawi`, então ele também deve ser subordinado de `Bin_Laden`, uma vez que `subordinate` é transitiva. A propriedade `owl:TransitiveProperty` é uma subclasse de `owl:ObjectProperty`, logo se pode declarar que uma propriedade é transitiva usando-se uma sintaxe equivalente ao Código 9, mostrada no Código 10.

Código 10 – Declaração de uma propriedade transitiva, sintaxe 2.

```
<owl:TransitiveProperty rdf:ID="subordinate">
  <rdfs:domain rdf:resource="#Agent"/>
  <rdfs:range rdf:resource="#Agent"/>
</owl:TransitiveProperty>
```

As outras características de propriedades, `owl:SymmetricProperty`, `owl:FunctionalProperty`, `owl:InverseFunctionalProperty` e `owl:inverseOf`, também são subclasses `owl:ObjectProperty`.

3.3.2 *SymmetricProperty*

As propriedades podem ser ditas **simétricas**. Se a propriedade `married` é dita ser simétrica e se é sabido que `Bin_Laden` é casado com `Najwa_Ghanem`, então se pode deduzir que `Najwa_Ghanem` também é casada com `Bin_Laden`. O Código 11, mostra este exemplo em OWL.

Código 11 – Declaração de uma propriedade simétrica.

```
<owl:SymmetricProperty rdf:ID="married">
  <rdfs:domain rdf:resource="http://www.mindswap.org/2003/owl
                                    /swint/person#Person"/>
  <rdfs:range rdf:resource="http://www.mindswap.org/2003/owl
                                    /swint/person#Person"/>
</owl:SymmetricProperty>

<Agent rdf:ID="Bin_Laden">
  <married rdf:resource="#Najwa_Ghanem"/>
</Agent>
```

No exemplo, indivíduo `Bin_Laden` é `casado` com o indivíduo `Najwa_Ghanem` e o indivíduo `Najwa_Ghanem` é `casado` com o indivíduo `Bin_Laden`.

3.3.3 *FunctionalProperty*

Uma propriedade **funcional** é aquela em que se tem um único valor de `y` para cada instância de `x`. Portanto, não tem mais do que um único valor para cada indivíduo. Uma propriedade com esta característica é dita ter cardinalidade miníma 0 e cardinalidade máxima 1. O Código 12, mostra a propriedade `husband` que é declarada como funcional.

Código 12 – Declaração de uma propriedade funcional.

```
<owl:FunctionalProperty rdf:ID="husband">
  <rdfs:domain rdf:resource="http://www.mindswap.org/2003/owl
                                    /swint/person#Woman"/>
  <rdfs:range rdf:resource="http://www.mindswap.org/2003/owl
                                   /swint/person#Man"/>
</owl:FunctionalProperty>
```

A propriedade `husband` só pode ter um único valor para o indivíduo, ou seja, uma mulher tem no máximo um marido. A `owl:FunctionalProperty` é uma subclasse de `owl:DatatypeProperty` e também é uma subclasse de `owl:ObjectProperty` [1].

As propriedades podem ser ditas **funcionais inversas**. Se uma propriedade é funcional inversa, então, o inverso da propriedade é funcional [8]. Deste modo, o inverso da propriedade tem, no máximo, um valor para cada indivíduo. Esta característica pode também ser usada para dizer que uma propriedade é não ambígua, como mostra o Código 13.

Código 13 – Declaração de uma propriedade funcional inversa.

```
<owl:InverseFunctionalProperty rdf:ID="biologicalMother">
    <rdfs:domain rdf:resource="http://www.mindswap.org/2003
                                    /owl/swint/person#Woman"/>
    <rdfs:range rdf:resource="#Agent"/>
</owl:InverseFunctionalProperty>
```

A definição da propriedade `biologicalMother` como sendo `owl:InverseFunctionalProperty`, garante que um agente terrorista (`#Agent`)

tenha uma única mãe biológica. Isto é, para cada objeto da declaração `biologicalMother` (um agente como **Bin Laden**, por exemplo) deve ser possível identificar unicamente o sujeito da declaração (a mãe biológica de Bin Laden, **Hamida al-Attas**). Os elementos do `range` em uma propriedade funcional inversa podem ser visualizados como chaves únicas em banco de dados [16].

3.3.4 *InverseOf*

Uma propriedade pode ser o inverso de uma outra propriedade. Por exemplo, se a propriedade `subordinateOf` é o inverso da propriedade `subordinate` e se `Ayman_AlZawahiri` é subordinado de `Bin_Laden`, então se pode deduzir que `Bin_Laden` é o subordinador de `Ayman_AlZawahiri`. Este exemplo é mostrado no Código 14.

Código 14 – Declaração de uma propriedade inversa.

```
1  <owl:TransitiveProperty rdf:ID="subordinateOf">
2    <owl:inverseOf rdf:resource="#subordinate"/>
3  </owl:TransitiveProperty>
4
5  <Agent rdf:ID="Ayman_AlZawahiri">
6     <subordinateOf rdf:resource="#Bin_Laden"/>
7  </Agent>
```

Na Subseção 3.4, se discute restrições em propriedades.

3.4 Restrições em propriedades

A OWL permite que sejam impostas restrições sobre propriedades. Uma restrição é um tipo especial de descrição de classe, isto é, descreve uma classe anônima de indivíduos que satisfazem as restrições. As restrições podem ser de valores (`allValuesFrom`, `someValuesFrom` e `hasValue`) ou de cardinalidade (`maxCardinality`, `minCardinality` e `Cardinality`). Estas restrições serão apresentadas nas próximas Subseções.

3.4.1 *allValuesFrom*

A restrição `allValuesFrom` é declarada em uma propriedade com respeito a uma classe, isto é, a restrição requer que, para cada instância da classe que tenha instâncias da propriedade especificada, os valores da propriedade devem ser todos os membros da classe indicada pela cláusula `allValuesFrom` [16]. Esta é uma restrição local, diferente das restrições de `domain` e `range`, discutidas na Seção 3.2.2, que são restrições globais. O Código 15 mostra como usar a restrição `allValuesFrom` na OWL.

Código 15 – Restrição dos valores de uma propriedade usando `allValuesFrom`.

```
1 <owl:Class rdf:ID="MarriedAgent">
2  ...
3   <owl:Restriction>
4     <owl:onProperty rdf:resource="#haveChild"/>
5     <owl:allValuesFrom rdf:resource="&pers;Person"/>
6   </owl:Restriction>
7 </owl:Class>
```

Todas as instâncias da classe `MarriedAgent` que usam a propriedade `haveChild` devem ter como valores associados a esta propriedade instâncias da classe `&pers;Person`. Esta restrição é análoga ao quantificador universal da lógica de predicados. Isto é, não é requerido que os `MarriedAgent` tenham filhos, mas se tiverem, estes devem ser `&pers;Person`.

As definições das linhas 3 a 6 definem uma classe não nomeada que representa o conjunto de coisas que possuem a propriedade `haveChild` cujo valor são instâncias da classe `&pers;Person`. Esta construção é chamada de **classe anônima**. Deve-se observar que `owl:Restriction` é uma subclasse da `owl:Class`. Como esta classe anônima foi definida dentro do escopo de definição da classe `MarriedAgent`, então os membros desta classe são também membros desta classe anônima. A definição de classes anônimas evita a necessidade de "inventar" nomes para cada classe e, além disso, fornece uma poderoso mecanismo de definição de classes baseado na satisfação de uma propriedade [16].

3.4.2 *someValuesFrom*

A restrição `someValuesFrom` descreve a classe de indivíduos `x` para os quais existe pelo menos um `y`, tal que o par `(x,y)` seja uma instância da propriedade, isto é, a restrição requer que, pelo menos para uma instância da classe que tenha instâncias da propriedade especificada, os valores da propriedade devem ser membros da classe indicada pela cláusula `someValuesFrom` [16]. Esta é uma restrição local, diferente das restrições de `domain` e `range`, discutidas na Seção 3.2.2, que são restrições globais. O Código 16 mostra como usar a restrição `someValuesFrom` na OWL.

Código 16 – Restrição dos valores de uma propriedade usando `someValuesFrom`.

```
1 <owl:Class rdf:ID="Agent">
2  ...
3   <rdfs:subClassOf rdf:resource="&pers;Person"/>
4     <owl:Restriction>
5       <owl:onProperty rdf:resource="#location"/>
6       <owl:someValuesFrom rdf:resource="#Location"/>
7     </owl:Restriction>
8   </rdfs:subClassOf>
9 </owl:Class>
```

No exemplo, pelo menos uma instância da classe `Agent` que tenha a propriedade `Location` deve ter como valor associado a esta propriedade instâncias da classe `Location`. Esta restrição é análoga ao quantificador existencial da lógica de predicados, ou seja, para cada instância da classe sendo definida, existe pelo menos um valor que satisfaz a restrição. Todas as instâncias da classe `Agent` têm pelo menos uma localização que é da classe `Location`.

3.4.3 *hasValue*

A propriedade `hasvalue` permite que uma propriedade tenha um certo indivíduo como valor. Em outras palavras, `hasvalue` permite especificar classes baseadas na existência de valores de propriedades particulares, assim, um indivíduo será membro de tal classe sempre que pelo menos um dos valores das propriedades for igual ao valor do recurso `hasvalue` [16]. Esta restrição é uma restrição local. O Código 17 mostra o uso desta restrição.

Código 17 – Restrição dos valores de uma propriedade usando `hasValue`.

```
1  <owl:Class rdf:ID="TerroristBigEvent">
2   ...
3   <owl:Restriction>
4       <owl:onProperty rdf:resource="#organizer"/>
5       <owl:hasValue rdf:resource="#TerroristLeader">
6   </owl:Restriction>
7  </owl:Class>
```

Instâncias da classe `TerroristBigEvent` podem ser caracterizadas pelos eventos que têm líderes terroristas como valor para a propriedade `organizer`.

3.4.4 Restrições de cardinalidade

A OWL permite utilizar restrições de cardinalidade, uma vez que qualquer instância de uma classe pode ter um número arbitrário de valores para uma propriedade. As restrições de cardinalidade são restrições locais, uma vez que elas são declaradas em propriedades com respeito a uma classe. Segundo Bechhofer et al. [1], a OWL provê três construções para cardinalidade:

- `owl:maxCardinality`: descreve uma classe de todos os indivíduos que têm, no máximo, `N` valores semanticamente distintos.
- `owl:minCardinality`: descreve uma classe de todos os indivíduos que têm, no mínimo, `N` valores semanticamente distintos. Esta restrição é um meio para dizer que uma propriedade é requerida ter um valor para todas as instâncias da classe.
- `owl:cardinality`: descreve uma classe de todos os indivíduos que têm exatamente `N` valores semanticamente distintos.

Código 18 – Restrigindo a quantidade de jogadores de um time.

```
1  <owl:Class rdf:ID="Agent">
2   ...
3    <rdfs:subClassOf rdf:resource="&pers;Person"/>
4      <owl:Restriction>
5        <owl:onProperty rdf:resource="#affiliate"/>
6        <owl:mincardinality rdf:datatype=
7                  "&xsd;nonNegativeInteger">1</owl:cardinality>
8      </owl:Restriction>
9    </rdfs:subClassOf>
10 </owl:Class>
```

O Código 18, usando a `owl:mincardinality`, diz que um agente possui no mínimo `1` afiliação. A declaração na linha 6, `owl:mincardinality`, pode ser substituída por `owl:cardinality` ou `owl:maxCardinality`.

Na Subseção 3.5, se discute o uso de definições de ontologias de fontes diferentes.

3.5 Mapeamento de ontologias

Uma das principais dificuldades no desenvolvimento de uma ontologia é juntar classes e propriedades, de ontologias diferentes, da melhor maneira possível para se fazer inferência. Nas próximas Subseções serão apresentadas construções que permitem fazer afirmações sobre classes, indivíduos e propriedades de fontes diferentes.

3.5.1 Equivalência entre Classes

Duas classes podem ser equivalentes, ou seja, podem possuir exatamente a mesma extensão de classe. Esta construção pode ser usada para criar classes sinônimas e para ligar duas classes na mesma ontologia ou em ontologias diferentes. O Código 19 mostra um exemplo de classes equivalentes.

Código 19 – Declaração da construção `owl:equivalentClass`.

```
1 <owl:Class rdf:ID="TerroristLeader">
2     <owl:equivalentClass>
3         <owl:Restriction>
4             <owl:onProperty rdf:resource="#organizerOf"/>
5             <owl:hasValue rdf:resource="#TerroristEvent">
6         </owl:Restriction>
7     </owl:equivalentClass>
8 </owl:Class>
```

Este código diz que um indivíduo que é o organizador de um evento terrorista é também instância da classe `TerroristEvent` e vice-versa. A construção `owl:equivalentClass` não implica em igualdade de classes, ou seja, duas classes são iguais se elas possuem o mesmo significado intensional [1]. Além disso, esta construção possui uma diferença significativa em relação a construção `rdfs:subClassOf` mostrada no Código 20.

Código 20 – Comparação das construções `owl:equivalentClass` e `rdfs:subClassOf`.

```
1 <owl:Class rdf:ID="TerroristLeader">
2     <rdfs:subClasseOf>
3         <owl:Restriction>
4             <owl:onProperty rdf:resource="#organizerOf"/>
5             <owl:hasValue rdf:resource="#TerroristEvent">
6         </owl:Restriction>
7     </rdfs:subClasseOf>
8 </owl:Class>
```

No exemplo mostrado no Código 20, usa-se a construção `rdfs:subClassOf` (no lugar da construção `owl:equivalentClass`, mostrada no Código 19). Nesta construção,

os indivíduos que são instâncias da classe `TerroristEvent` também são organizadores de eventos terroristas, mas organizadores de eventos terroristas não são necessariamente instâncias da classe `TerroristEvent`. Esta construção difere da construção do Código 19 (que faz uso da construção `owl:equivalentClass`) que diz que um indivíduo que é um organizador de um evento terrorista é também instância da classe `TerroristEvent` e vice-versa.

3.5.2 Equivalência entre Propriedades

Duas propriedades podem ser ditas equivalentes usando-se a construção `owl:equivalentProperty`. Segundo [1], sintaticamente, a construção `owl:equivalentProperty` é a definição de uma propriedade com o mesmo domínio (`domain`) e valores (`range`) de outra propriedade. Esta construção pode ser usada para criar propriedades sinônimas e para ligar duas propriedades na mesma ontologia ou em ontologias diferentes. O Código 21 mostra o emprego de propriedades equivalentes.

Código 21 – Declaração da construção `owl:equivalentProperty`.

```
1 <owl:DatatypeProperty rdf:ID="name">
2    <owl:equivalentProperty rdf:resource="&pers;name"/>
3 </owl:DatatypeProperty>
```

Neste código, se pode deduzir que `name` é uma sub-propriedade de `&pers;name` (está definida em uma outra ontologia) e `&pers;name` é uma sub-propriedade de `name`. Vale observar que a equivalência entre propriedades não é como igualdade de propriedades, isto é, propriedades equivalentes possuem os mesmos valores, mas podem ter significado intensional diferentes [1].

3.5.3 Igualdade de Indivíduos

Dois indivíduos podem ser ditos iguais usando-se a construção `sameAs`. Esta construção pode ser usada para criar vários nomes diferentes que se referem a um mesmo indivíduo [8]. O Código 22 mostra uma declaração de indivíduos iguais.

Código 22 – Declaração da construção `owl:sameAs`.

```
1 <TerroristLeader rdf:ID="NumberOneTerrorist">
2     <owl:sameAs rdf:resource="#Osama_Bin_Laden"/>
3 </TerroristLeader>
```

O exemplo diz que o indivíduo `NumberOneTerrorist` é idêntico ao indivíduo `Osama_Bin_Laden`. Um uso típico da `owl:sameAs` é para unificação de ontologias, ou seja, para dizer que dois indivíduos, definidos em documentos diferentes, são iguais.

3.5.4 Indivíduos Diferentes

Uma vez que, em uma ontologia OWL, não se assume que nomes diferentes se referem a indivíduos diferentes, ela permite que se diga explicitamente que certos indivíduos são diferentes. Existem duas construções para isso:

- `owl:differentFrom`: esta construção diz que um indivíduo é diferente de outro.

- `owl:AllDifferent`: esta construção pode ser usada para dizer que certos indivíduos são mutuamente distintos uns dos outros.

Código 23 – Declaração da construção `owl:differentFrom`.

```
<Terrorist rdf:ID="Bin_Laden">

<Terrorist rdf:ID="Al_Zarqawi">
  <owl:differentFrom rdf:resource="#Bin_Laden"/>
</Terrorist>

<Terrorist rdf:ID="Mullah_Krekar">
  <owl:differentFrom rdf:resource="#Bin_Laden"/>
  <owl:differentFrom rdf:resource="#Al_Zarqawi"/>
</Terrorist>
```

O Código 23 diz que os três indivíduos (`Bin_Laden`, `Al_Zarqawi` e `Mullah_Krekar`) são mutuamente distintos. Entretanto, existe um mecanismo mais conveniente para definir um conjunto de indivíduos mutuamente distintos. O Código 24 diz que três indivíduos são diferentes.

Código 24 – Declaração da construção `owl:AllDifferent`.

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Terrorist rdf:about="#Bin_Laden"/>
    <Terrorist rdf:about="#Al_Zarqawi"/>
    <Terrorist rdf:about="#Mullah_Krekar"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```

Segundo Bechhofer et al. [1], `owl:distinctMembers` é uma construção especial que deve ser sempre usada como sujeito da `owl:AllDifferent`. Na Subseção 3.6, são apresentados os construtores para definição de classes complexas.

3.6 Classes Complexas

A OWL oferece construtores adicionais que podem ser usados na construção de classes, as chamadas ***class expressions***. OWL suporta um conjunto básico de operações: união, interseção e complemento. Adicionalmente, as classes podem ser enumeradas. Também é possível afirmar que as extensões de classes são disjuntas. Estes construtores são discutidos nas próximas Subseções.

3.6.1 Conjunto de Operadores

A OWL permite combinações *booleanas* arbitrárias para manipulações das extensões de classes, chamados de conjunto de operadores. Os membros das classes construídas são especificados pelo conjunto de operadores. Este conjunto de operadores podem ser visualizados como uma representação dos operadores **AND**, **OR** e **NOT**, usados na Lógica Descritiva, para serem usados em classes [1]. Segundo Bechhofer et al. [1], são três estes operadores:

- `owl:intersectionOf`: declara uma classe cuja extensão de classe contém somente indivíduos que são membros da extensão de classe de todas classes descritas em uma lista. Este operador é análogo à conjunção lógica, **AND**. O Exemplo 25 mostra o uso deste operador.

- `owl:unionOf`: declara uma classe anônima cuja extensão de classe contém indivíduos que ocorrem em pelo menos uma extensão de classe das classes descritas em uma lista. Este operador é análogo a disjunção lógica, **OR**. O Exemplo 27 mostra o uso deste operador.

- `owl:complementOf`: declara uma classe cuja extensão de classe contém exatamente os indivíduos que não pertencem à extensão de classe de outra classe, que é o objeto da declaração. Este operador é análogo a negação lógica, **NOT**. O Exemplo 28 mostra o uso deste operador.

Código 25 – Construção 1 usando o operador de interseção.

```
1 <owl:Class rdf:ID="WomanAgent">
2   <owl:intersectionOf rdf:parseType="Collection">
3     <owl:Class rdf:about="#Agent">
4     <owl:Class rdf:about="&pers;Woman"/>
5   </owl:intersectionOf>
6 </owl:Class>
```

A classe `WomanAgent` é exatamente a interseção da classe `agent` e da classe `&pers;Woman`. A declaração do Código 25 pode também ser escrita de outra forma, como mostra o Código 26. Este código diz que a classe `WomanAgent` é exatamente a interseção da classe `Agent` e do conjunto de coisas que possuem a propriedade **sexo** com valor **feminino**. A vantagem desta declaração é que ela não obriga a criação de uma classe chamada, por exemplo, de `Woman`. O Código 27 mostra o uso do operador de união. Este exemplo diz que a classe `Person` é formada pela união de indivíduos da classe `Man` e da classe `Woman`. Já o Código 28 mostra o emprego do operador de complemento.

Código 26 – Construção 2 usando o operador de interseção.

```
1 <owl:Class rdf:ID="WomanAgent"/>
2   <owl:intersectionOf rdf:parseType="Collection">
3     <owl:Class rdf:about="#Agent"/>
4     <owl:Restriction>
5       <owl:onProperty rdf:resource="#sex"/>
6       <owl:hasValue rdf:resource="&pers;Female">
7     </owl:Restriction>
8   </owl:intersectionOf>
9 </owl:Class>
```

Código 27 – Construção usando o operador de união.

```
<owl:Class rdf:ID="Person"/>
    <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Man">
        <owl:Class rdf:about="#Woman"/>
    </owl:unionOf>
</owl:Class>
```

Código 28 – Construção usando o operador de complemento.

```
<owl:Class rdf:ID="Man"/>
    <owl:complementOf>
        <owl:Class rdf:about="#Woman">
    </owl:complementOf>
</owl:Class>>
```

O Código 28, diz que a classe `Man` contém todos os indivíduos que não pertencem à classe `Woman`.

3.6.2 Classes Enumeradas

A OWL permite que as classes sejam descritas pela enumeração dos indivíduos que são instâncias da classe [8]. Os membros da classe são exatamente o conjunto de indivíduos enumerados. O tipo de enumeração é definido pela propriedade `owl:oneOf` e a lista de indivíduos é representada com ajuda do construtor RDF `rdf:parseType="Collection"` [1]. O Código 29 mostra a declaração de uma classe enumerada.

Código 29 – Construção de uma classe enumerada.

```
<owl:Class rdf:ID="BaseCountry"/>
    <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Iraq">
        <owl:Thing rdf:about="#Afghanistan">
        <owl:Thing rdf:about="#Saudi_Arabia">
        <owl:Thing rdf:about="#Pakistan">
    </owl:oneOf>
</owl:Class>
```

O Código 29 diz que nenhum outro indivíduo pode ser um indivíduo da classe `BaseCountry`, além dos indivíduos declarados pela propriedade de enumeração. Todos os elementos da construção `owl:oneOf` devem ser indivíduos válidos declarados, ou seja, um indivíduo deve pertencer a alguma classe [16]. No exemplo, todos os indivíduos são instâncias da classe `owl:Thing`, por definição.

3.6.3 Classes Disjuntas

A OWL permite dizer que a extensão de classe de uma classe não tem membros em comum com a extensão de classe de uma outra classe, ou seja, ambas extensões de classe são

disjuntas [1]. A disjunção de um conjunto de classes pode ser expresso usando-se o construtor `owl:disjointWith`. O Código 30 mostra a criação de classes disjuntas.

Código 30 – Construção de classes disjuntas.

```
1 <owl:Class rdf:ID="TerroristOrganization" />
2     <rdfs:subClassOf rdf:resource="#Organization"/>
3     <owl:disjointWith rdf:resource="#NonGovermentOrganization">
4     <owl:disjointWith rdf:resource="#GovermentOrganization">
5     <owl:disjointWith rdf:resource="#MilitaryOrganization">
6 </owl:Class>
```

O Exemplo 30, afirma que a classe `TerroristOrganization` é disjunta de todas as outras classes. Entretanto, a construção mostrada no exemplo, não afirma que `NonGovermentOrganization` e `GovermentOrganization` são classes disjuntas. Para afimar que todas as classes, do Exemplo 30, são mutuamente disjuntas, é necessário criar uma construção com `owl:disjointWith` para cada par de classes que são disjuntas.

4 Considerações Finais

A *Web* Semântica é uma visão do futuro da *Web* atual, onde as informações terão um significado explícito e, com isso, as máquinas e os agentes de *softwares* poderão fazer inferências mais precisas e interagirem de maneira mais confiável com o seres humanos [10]. Para que a visão do futuro em torno da *Web* Semântica seja possível, uma linguagem de ontologia para *Web* é essencial. A OWL é uma destas linguagens.

Uma ontologia escrita em OWL define formalmente um conjunto comum de termos que são usados para descrever e representar um domínio específico. Assim, a OWL pode ser usada por ferramentas automatizadas para melhorar os serviços avançados como buscas na *Web* e gerenciamento de conhecimento [9].

OWL também permite a unificação do conhecimento e a formação de consensos sobre certos conceitos, através do reuso de ontologias. Esta característica tem levado à uma integração maior de usuários dentro domínios específicos do conhecimento.

Por todas estas vantagens e pelo trabalho que vem sendo feito pelo W3C, a OWL está se tornando a linguagem padrão para criação de ontologias para *Web*, principalmente no contexto da *Web* Semântica.

5 Agradecimento

A Profa. Dra. Ana Paula Laboissière Ambrósio, pela avaliação do presente texto e pelas sugestões feitas, as quais muito contribuiram para a melhoria do texto original.

Referências

[1] BECHHOFER, S; HARMELEN, F. V; HENDLER, J; HORROCKS, I; MCGUINNESS, D. L; PATEL-SCHNEIDER, P. F; STEIN, L. A. **OWL Web Ontology Language Reference**. http://www.w3.org/TR/2004/REC-owl-ref-20040210/, acessado em Abril 2005, Fevereiro 2004.

[2] BRAY, T; AND; C. M. SPERBERG-MCQUEEN, J. P; YERGEAU, F. **Extensible Markup Language (XML) 1.0**. http://www.w3.org/TR/2004/REC-xml-20040204/, acessado em Abril 2005, Fevereiro 2004.

[3] BRAY, T; HOLLANDER, D; LAYMAN, A; TOBIN, R. **Namespaces in XML 1.1**. http://www.w3.org/TR/2004/REC-xml-names11-20040204/, acessado em Abril 2005, Fevereiro 2004.

[4] DAML. **Home Page**. http://www.daml.org/ , acessado em Abril de 2005.

[5] DAML+OIL. **Home Page**. http://www.daml.org/ , acessado em Abril de 2005.

[6] FERREIRA, A. B. H. **Dicionário Aurélio Básico da Língua Portuguesa**. Nova Fronteira, 1ª edition, 1988.

[7] GRUBER, T. **What is an Ontology?** http://www-ksl.stanford.edu/kst/what-is-an-ontology.html, acessado em Abril de 2005.

[8] HARMELEN, F. V; MCGUINNESS, D. L. **OWL Web Ontology Language Overview**. http://www.w3.org/TR/2004/REC-owl-features-20040210/, acessado em Abril de 2005, Fevereiro 2004.

[9] HEFLIN, J. **OWL Web Ontology Language Use Cases and Requirements**. http://www.w3.org/TR/2004/REC-webont-req-20040210/, acessado em Abril 2005, Fevereiro 2004.

[10] HENDLER, J; BERNERS-LEE, T; LASSILA, O. **The Semantic Web**. Issue of Scientific American, Maio 2001.

[11] MANOLA, F; MILLER, E. **RDF Primer**. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, acessado em Abril de 2005, Fevereiro 2004.

[12] MINDSWAP. **Ontology Person**. http://www.mindswap.org/2003/owl/swint/person, acessado em Abril de 2005.

[13] MINDSWAP. **Ontology Terrorism**. http://www.mindswap.org/2003/owl/swint/terrorism, acessado em Abril de 2005.

[14] OIL. **Home Page**. http://www.ontoknowledge.org/oil/ , acessado em Abril de 2005.

[15] SHOE. **Home Page**. http://www.cs.umd.edu/projects/plus/SHOE/ , acessado em Abril de 2005.

[16] SMITH, M. K; WELTY, C; MCGUINNESS, D. L. **OWL Web Ontology Language Guide**. http://www.w3.org/TR/2004/REC-owl-guide-20040210/, acessado em Abril de 2005, Fevereiro 2004.

[17] W3C. **Home Page**. `http://www.w3.org/`, acessado em Abril de 2005.

[18] XOL. **Home Page**. `http://www.ai.sri.com/pkarp/xol/` , acessado em Abril de 2005.