FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
MESTRADO EM INFORMÁTICA APLICADA

Leonardo Ayres de Morais e Silva

OWLPREF: UMA REPRESENTAÇÃO DECLARATIVA DE PREFERÊNCIAS QUALITATIVAS PARA A WEB SEMÂNTICA

Fortaleza
2007

FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
MESTRADO EM INFORMÁTICA APLICADA

Leonardo Ayres de Morais e Silva

OWLPREF: UMA REPRESENTAÇÃO DECLARATIVA DE PREFERÊNCIAS QUALITATIVAS PARA A WEB SEMÂNTICA

Dissertação apresentada ao Curso de Mestrado em Informática Aplicada da Universidade de Fortaleza como requisito parcial para a obtenção do Título de Mestre em Informática.

Orientador: Prof. Dr. João José Vasco Peixoto Furtado

Fortaleza
2007

Leonardo Ayres de Morais e Silva

OWLPREF: UMA REPRESENTAÇÃO DECLARATIVA DE PREFERÊNCIAS QUALITATIVAS PARA A WEB SEMÂNTICA

Data de Aprovação: __/__/____

Banca Examinadora:

_____

Prof. João José Vasco Peixoto Furtado, D.Sc.

(Prof. Orientador – Universidade de Fortaleza - UNIFOR)

_____

Prof. Pedro Porfírio Muniz Farias, D. Sc.

(Membro – Universidade de Fortaleza - UNIFOR)

_____

Prof. Ana Cristina Bicharra Garcia, Ph.D.

(Membro – Universidade Federal Fluminense - UFF)

SILVA, LEONARDO AYRES DE MORAIS E

OWLPREF: Uma Representação Declarativa de Preferências Qualitativas para a Web Semântica [Fortaleza] 2007

xiii, 126p., (MIA/UNIFOR, M. Sc., Ciência da Computação, 2007)

1. Web Semântica
2. Preferências
3. Representação de Conhecimento
4. Ontologias

Aos meus pais e minhas irmãs.

"There are two kinds of people in the world, those who believe there are two kinds of people in the world and those who don't."

Robert Benchley, *Benchley's Law of Distinction*

AGRADECIMENTOS

Em primeiro lugar, aos meus pais, Rui e Maria, e às minhas irmãs, Flávia e Fernanda, pelo apoio e incentivo em todos os sentidos, que, mesmo com a distância, sempre estiveram ao meu lado, oferecendo todas as condições possíveis para que eu conseguisse meus objetivos. Tenho orgulho de ser filho deles e ter uma família como essa. A eles dedico este trabalho.

À minha namorada Renata, pela paciência e compreensão nos vários momentos em que estive ausente devido a esse trabalho. Amo você.

Aos meus amigos e colegas da UNIFOR da Informática, do Direito e de outros cursos, que compartilharam comigo estes anos de estudo e trabalho. Foram experiências e amizades com as quais tive oportunidade de aprender muito.

Aos amigos da célula de Engenharia do Conhecimento da UNIFOR, em especial ao casal 20 Juliana e Gustavo, Adriano e Daniel que conviveram e compartilharam comigo quase que diariamente o esforço dispensado a esse trabalho. Obrigado pela força!

Ao meu orientador e professor Vasco por sua competência e experiência. Sempre me deu um norte quando achava que estava sem saída. Um exemplo de profissional a ser seguido.

A todos colegas do BNB e ex-colegas de trabalho do SERPRO que conviveram comigo durante o período do mestrado, pelo conhecimento adquirido tanto no lado acadêmico quanto no lado pessoal.

A todos os professores do Mestrado da UNIFOR que, de uma forma ou de outra, contribuíram para o meu crescimento pessoal.

Aos meus amigos de Teresina, minha terra natal, os quais são muitos e por esse motivo seria impossível citar todos, pela amizade e pelas experiências de vida.

Aos amigos da turma do funil que mesmo nos momentos de lazer me incentivaram na conclusão deste trabalho.

A todas as outras pessoas que me apoiaram e ajudaram direta e indiretamente para a realização de meus objetivos.

Agradeço também às dificuldades surgidas, pois de uma forma ou de outra me fizeram crescer.

E por fim, a Deus, que sem Ele nada disso poderia ser possível.

Resumo da Dissertação apresentada ao Corpo Docente do Curso de Mestrado em Informática Aplicada da Universidade de Fortaleza como parte dos requisitos necessários para a obtenção do título de Mestre em Ciência da Computação (M.Sc.)

OWLPREF: UMA REPRESENTAÇÃO DECLARATIVA DE PREFERÊNCIAS QUALITATIVAS PARA A WEB SEMÂNTICA

Leonardo Ayres de Morais e Silva

Julho / 2007

Orientador: João José Vasco Peixoto Furtado, D.Sc.

Filtrar informações baseadas em preferências é uma característica essencial de sistemas multiagentes se estes necessitam acessar dados na Web Semântica. Entretanto, poucos trabalhos têm estudado a questão da representação genérica de preferências para uso por esses agentes. O objetivo deste trabalho é descrever nossa proposta de representação de preferências de maneira declarativa, independente de domínio e interpretável por máquinas usando OWL. OWLPREF será útil para o compartilhamento de conhecimento entre agentes de software, permitindo a representação parcial de ordem entre atributos, classes e valores de atributos. Neste trabalho descrevemos a interface de software que mapeia OWLPREF em consultas SPARQL e a exemplificamos em uma aplicação na Web Semântica, demonstrando sua capacidade de representação e compartilhamento de preferências.

Palavras-chave: Web Semântica. Representação do Conhecimento. Ontologias. Preferências. Inteligência Artificial.

Abstract of the dissertation presented to the board of faculties of the Master Program in Applied Informatics at the University of Fortaleza, as partial fulfillment of the requirements for the degree of Master of Computer Science (M.Sc.)

OWLPREF: A DECLARATIVE REPRESENTATION OF QUALITATIVE PREFERENCES FOR THE SEMANTIC WEB

Leonardo Ayres de Morais e Silva

July / 2007

Advisor: João José Vasco Peixoto Furtado, D.Sc.

Preference information filtering is a required feature of multi-agent systems if they need accessing a database on Semantic Web. However, few studies have paid attention to creating a generic representation of preference for agent consumption. In this work, we describe our proposal for representing preferences in a declarative, domain-independent and machine-interpretable way in OWL. The aim of OWLPREF is to be used for knowledge sharing among agents, allowing the representation of partial order between classes, attributes and attribute-values. We also describe the API that translates OWLPREF into SPARQL queries, as well as exemplify the use of the ontology and of the API in the development of multi-agent application in the Semantic Web context to demonstrate OWLPREF capabilities of sharing and representation of preferences.

Keywords: Semantic Web. Knowledge Representation. Ontologies. Preferences. Artificial Intelligence.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação e Problema

A representação e manipulação de preferências é um problema que vem sendo estudado em diversas áreas da computação e ganhou particular importância no contexto da Web Semântica. Nesse contexto, espera-se que agentes de software sejam capazes de compreender a semântica dos termos representados nas páginas web bem como das requisições e preferências dos usuários. A representação da semântica dos dados proporciona a interoperabilidade e acessibilidade por agentes de software. Uma definição de que *Carro* e *Automóvel* estão relacionados semanticamente pode, por exemplo, ampliar o poder de consulta na web e evitar ambigüidades. Para (Lukasiewicz & Schellhase, 2007), a idéia principal da Web Semântica é a de adicionar significado às paginas da Web que sejam compreensíveis por máquinas, utilizar ontologias para uma definição precisa de termos compartilhados dentro de recursos Web, fazer uso de representação de conhecimento e inferência para o raciocínio automático de recursos Web e utilizar agentes cooperativos para o processamento de informações da Web.

A forma atual de representação de preferências por meio de palavras-chaves se apresenta ambígua (polissemia) e insuficiente para compreensão de agentes de software. Ao se prover representações semânticas, atividades usuais na Web como a realização de buscas ou a realização de compras poderão gradativamente ser realizadas autonomamente com o auxílio, por exemplo, de agentes de software.

Poucos trabalhos têm proposto alternativas de representação de preferências para o contexto da web. O mais representativo trabalho nessa direção foi proposto recentemente por (Siberski, Pan, & Thaden, 2006) que estendeu a linguagem de consulta SPARQL para possuir cláusulas que representam ordem entre triplas RDF. Nesse enfoque a representação de preferências complexas requer a compreensão da semântica de uma linguagem de consulta e um completo domínio de suas estruturas que nem sempre são triviais de se obter. Em conseqüência, o compartilhamento de conhecimento entre agentes (quer seja de software ou

não) requer a interpretação da linguagem e não está explícita na representação. Ademais seria aconselhável que a representação usada fosse padronizada para que o impacto em termos de interoperabilidade fosse o maior possível.

Exemplos de como representação de preferências através de sentenças de linguagens de consulta podem ficar complexas podem ser visto abaixo.

A primeira representa uma consulta em uma tabela de carros da marca GM onde existe uma preferência por carros do tipo sedan (caso não exista nenhum sedan então, os que não sejam uma picape) , preço por volta de 40000 e motor com o maior valor possível. Menos importante que essas características, é que a cor seja vermelha. Por fim, menos importante que a cor é que a quilometragem seja a menor possível.

1. SELECT * FROM carros WHERE marca = 'GM' PREFERRING (categoria = 'sedan' ELSE categoria <> 'picape' AND preco AROUND 40000 AND HIGHEST(motor) CASCADE cor = 'vermelho' CASCADE LOWEST(quilometragem);

A segunda representa uma consulta em uma tabela de carros usados onde existe uma preferência por carros da BMW 7. Menos importante é que o preço fiquei entre 0 e 75000, a quilometragem esteja entre 0 e 30000, ano de fabricação entre 97 e 99, seja a diesel com airbag, automática e com arcondicionado.

2. SELECT * TOP(fabricante), TOP(modelo), TOP(preço), TOP(quilometragem), TOP(ano_fabricacao), TOP(diesel), TOP(airbag), TOP(transmissão_automatica), TOP(ar condicionado) FROM carros_usados PREFERRING fabricante = 'BMW' AND modelo = '7' CASCADE preço BETWEEN 0, 75000 AND quilometragem BETWEEN 0, 30000 AND ano_fabricacao between 1997, 1999 AND diesel = 'sim' AND airbag = 'sim' AND transmissão_automatica = 'sim' AND ar_condicionado = 'sim';

1.2 Proposta

Para mitigar essas deficiências propomos um enfoque diferente. Optamos por adotar uma representação declarativa para a representação de preferências e o fizemos com a definição de uma ontologia genérica que possui por base o conceito de preferências simples e que, a partir delas, pode-se construir preferências mais complexas. Tal representação

apresenta a flexibilidade e expressividade de uma linguagem de consulta, mas com a compreensibilidade natural das representações declarativas.

Abaixo, na Figura 1 mostramos um exemplo de uma representação de preferências por carros com características específicas (similar ao exemplo anterior) onde a forma de representação fica mais simples e intuitiva para as pessoas modelarem. No caso, a preferência de um usuário é representada pela instância de um conceito que representa uma preferência (linha 1 da Figura 1). Essa instância de prerência referencia sempre um contexto, (no caso o conceito Carro visto que trata-se de uma preferência relacionada a carros) e uma espécie de preferência (PREFERENCIA_ESPECIFICA) relacionada a este contexto. Na linha 2, mostramos que essa espécie de preferência é uma instância de um conceito de preferência específica (PREFERENCIA_DE_ORDEM) no qual representa uma ordem de importância entre dois conceitos de preferências diferentes (a PREFERENCIA_ESPECIFICA_2 tem mais prioridade que a PREFERENCIA_ESPECIFICA_3). Essas por sua vez referenciam-se a outras instâncias de preferências exemplificadas nas linhas 3 e 4 da Figura 1. A PREFERENCIA_ESPECIFICA_2 é uma instância da PREFERENCIA_IGUAL que representa que a preferência pela categoria *hatch* tem a mesma importância que a preferência pela cor diferente de azul. Por fim, a PREFERENCIA_ESPECIFICA_3 é uma instância da PREFERENCIA MINIMA que representa o menor valor de preço.

1. PREFERENCIA_USUARIO: Instância de PREFERENCIA= {Carro, PREFERENCIA_ESPECIFICA_1);
2. PREFERENCIA_ESPECIFICA_1:
   Instância de PREFERENCIA_DE_ORDEM= {PREFERENCIA_ESPECIFICA_2, PREFERENCIA_ESPECIFICA_3}
3. PREFERENCIA_ESPECIFICA_2:
   Instância de PREFERENCIA_IGUAL= {PREFERENCIA(Categoria=Hatch), PREFERENCIA(Cor!=Azul)}
4. PREFERENCIA_ESPECIFICA_3:
   Instância de PREFERENCIA_MINIMA={(Preco)}

Figura 1. Exemplo de Representação de Preferências com Ontologias

Resumindo temos a representação da preferência por carros *hatch* com cor diferente de azul prioritariamente seguido pelos carros com menor preço.

Além da questão da compreensibilidade e simplicidade representacional, outra vantagem do uso de ontologias para representação de preferências é a possibilidade utilizar raciocionadores que permite a inferência sobre novas assertivas de preferências. Através de uma ontologia comum, agentes podem, por exemplo, fazer consultas ou assertivas que serão

entendidas pelos outros agentes (Lamparter, Ankolekar, Studer, Oberle, & Weinhardt, 2006) e derivar conclusões que não estavam explícitas. Ou seja, é possível através de ontologias que determinada preferência que não esteja explicíta possa ser inferida pelos agentes. Vejamos um exemplo:

Suponha que João represente através de ontologias que tem preferência por músicos de Blues. Suponha também que um determinado músico (por exemplo, Eric Clapton) esteja associado a um conceito que é uma subclasse de Blues (por exemplo, Blues Britânico). Assim, é possível que um agente avalie que Eric Clapton atende à preferência de João, apesar de está associado direatamente a Blues Britânico e não a Blues. Ele pode fazer isso inferindo que na ontologia de estilos musicais, Blues Britânico é uma subclasse de Blues.

Outro problema é a representação de preferências que variam de acordo com determinada condição através de linguagens de consulta, como por exemplo:

- Prefiro músicos de Blues se não existem músicos de Jazz;
- Prefiro músicos de Blues caso eu esteja na Inglaterra;
- Prefiro músicos de Blues se for a noite, etc.

No caso dessas preferências condicionais, uma possível solução seria o uso de um conceito que represente a condição combinado com um conceito que represente a preferência. Isso permite que preferências estejam explicitamente definidas evitando ambigüidades, permitindo o compartilhamento da mesma entre diversos agentes e até mesmo a possibilidade de mapear esses modelos, em determinados casos, para diferentes linguagens de consultas.

Por fim, a adoção de OWL como linguagem para representar as ontologias facilita a interoperabilidade de agentes de software por se tratar de um padrão adotado pela W3C. Pode-se assim supor que a compreensão de instâncias de ontologias representadas em OWL sejam naturalmente reconhecidas por agentes de software do que se os mesmos tivessem que interpretar uma extensão de uma linguagem de consulta como SPARQL.

1.3 Organização

Este trabalho está organizado em seis capítulos, incluído este capítulo introdutório e dois apêndices, descritos conforme a seguir:

O Capítulo 2, **Web Semântica: conceitos básicos,** apresenta uma revisão do estado da arte sobre a Web Semântica e alguns conceitos relacionados.

O Capítulo 3, **Preferências e Trabalhos Correlatos**, discorre sobre o conceito de preferências, formas de modelagens e trabalhos correlatos.

O Capítulo 4, **Ontologia de Preferências: OWLPREF**, apresenta a ontologia de preferências proposta (OWLPREF), seus conceitos e formas de combinações, bem como uma API para o mapeamento de linguagens de consultas que já utilizam a noção de preferências.

O Capítulo 5, **Prova de Conceito**, mostra um exemplo de uso dos conceitos da ontologia em uma aplicação multiagente com intuito de demonstrar sua aplicabilidade em um domínio específico.

O Capítulo 6, **Conclusão**, aborda as considerações finais, ressaltando as contribuições do trabalho, bem como as limitações, dificuldades encontradas e as possibilidades de trabalhos futuros.

O Apêndice I, **Ontologia de Preferências OWLPREF em RDF/XML**, apresenta a sintaxe de OWLPREF.

O Apêndice II, **Exemplos do Mapeamento de OWLPREF para SPARQL *Preference***, apresenta sentenças SPARQL *Preference* geradas a partir de exemplos modelados em OWLPREF.

2 WEB SEMÂNTICA: CONCEITOS BÁSICOS

Neste segundo capítulo faz-se uma breve revisão dos conceitos básicos da Web Semântica, como sua definição e arquitetura, e de outros conceitos relacionados que nos levam a apresentar o modelo em camadas. Inicialmente será abordado o conceito da Web Semântica e seu modelo em camadas. Posteriormente, alguns conceitos necessários para a compreensão de nossa proposta serão brevemente descritos.

2.1 Web Semântica

Com a evolução e a popularização da Web, o volume de dados gerados e movimentados tornou-se gigantesco. Fazer consultas precisas e eficientes a milhões de dados não estruturados tornou-se uma tarefa custosa, sendo necessária sua automatização. Para isso, é necessário que computadores compreendam o objetivo das consultas e os próprios dados espalhados pela Web. O problema é que a maioria do conteúdo da Web atualmente foi projetada para a compreensão apenas dos seres humanos e não de máquinas (no caso, softwares). Ou seja, computadores ainda não possuem uma maneira confiável de manipular significados de conteúdos na Web. Com isso, existe uma grande dificuldade por parte dos agentes computacionais em tratar essas informações disponíveis na Web através das máquinas de buscas existentes - pelo fato de que a maioria delas efetua buscas baseadas em palavras-chave. Isso ocasiona baixos índices de eficácia e eficiência (baixa precisão nos resultados, bem como desempenho ruim) o que acaba levando a vários questionamentos:

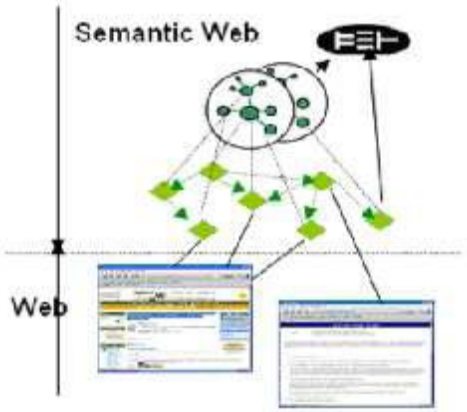
- Como recuperar a informação desejada com precisão?
- Como diminuir o esforço humano?
- Como coletar informações distribuídas que estão relacionadas entre si?
- Como recuperar informações confiáveis?

Para este fim, faz-se necessário que as informações estejam estruturadas e com semânticas bem definidas, possibilitando que os agentes computacionais sejam capazes de

processar e compreender esses dados. Isso facilitará a automação, integração e distribuição destes dados. É nesse contexto e com essa visão que surge a Web Semântica.

A Web Semântica visa fornecer essa infra-estrutura necessária para que o conteúdo da Web torne-se significativo tanto para os humanos quanto para os softwares. Ela é um esforço colaborativo liderado pela W3C[1] com a participação de um grande número de pesquisadores e indústrias.

Tim Berners-Lee, um dos principais pesquisadores do estudo e desenvolvimento dessa tecnologia e criador do termo Web Semântica, a define não como uma Web separada, mas como "uma extensão desta, na qual a informação dada é bem definida e dotada de significado, permitindo que computadores e pessoas trabalhem em cooperação" (Berners-Lee, Hendler, & Lassila, The Semantic Web, 2001). A sua visão da Web como o espaço para a cooperação entre pessoas e agentes computacionais com o intuito de realizar tarefas e inferir conhecimento só será alcançada quando os conceitos da Web Semântica estiverem amadurecidos.



.

Figura 2. Web Semântica como uma camada de alto nível sob a Web. Fonte: (IMG, 2005)

Na Figura 2 temos uma representação da Web Semântica como uma camada de mais alto nível sob a Web "tradicional". Assim, os conteúdos da Web, representados pelos sites na parte inferior da figura, são descritos semanticamente por meio de metadados (símbolos verdes da figura), nos quais são dados capazes de descrever outros dados. Essa

[1] Maiores informações sobre a W3C podem ser encontradas no endereço http://www.w3.org

figura representa bem um dos objetivos da Web Semântica que é fazer com que a Web deixe de ser uma rede de links e torne-se uma rede de conceitos. Desta forma, os dados passam a estar interligados semanticamente, facilitando a localização, automatização, integração e reuso através de várias aplicações diferentes.

2.1.1 Arquitetura

Hoje os principais objetivos dos grupos[2] de trabalhos que pesquisam Web Semântica é o desenvolvimento de padrões que, integrados, permitam aos agentes e aplicações compreender os dados disponíveis na Web e com isso, a Web Semântica se concretize.

(Antoniou & Harmelen, 2004) diz que o desenvolvimento da Web Semântica ocorre em etapas, onde cada etapa constitui uma camada em cima da anterior. A principal justificativa para essa abordagem é que é mais fácil obter um consenso a cada pequeno "passo" ou etapa. Porém, para isso, dois princípios devem ser seguidos:

- *Downward compatibility* – Aplicações sob uma determinada camada devem também estar aptas a interpretar e usar informações escritas em níveis mais baixos. Por exemplo, agentes que compreendem uma linguagem X que foi construída sob a camada Y podem tirar todas as vantagens de informações escritas nesta camada mais baixa.
- *Upward partial understanding* – Por outro lado, aplicações que compreendem totalmente uma camada, devem possuir pelo menos o conhecimento parcial de informações de níveis mais altos. Por exemplo, agentes que compreendem apenas semânticas de uma camada Y podem interpretar parcialmente conhecimento escrito em uma camada X, descartando elementos que não pertencem à camada Y.

Na Figura 3 temos o modelo "fatiado" da Web Semântica proposto por Tim Berners-Lee, a qual mostra as principais camadas seguindo os princípios acima citados. Na

[2] Os grupos de trabalho de pesquisa sobre a Web Semântica podem ser encontrados no endereço http://www.w3.org/2001/sw/

parte inferior da Figura 3, temos UNICODE[3] e URI[4] (*Uniform Resource Identifier)* como primeira camada que serve de base para todas as outras. Elas constituem-se como representação universal de padrão de caracteres e recursos, facilitando a legibilidade e o endereçamento de recursos/objetos na Web Semântica. Acima dessa camada, temos XML[5](*eXtensible Markup Language*) com a função de permitir a interoperabilidade sintática. Acima do XML, temos o RDF[6] como modelo de dados básico[7] com uma semântica formal que representa recursos Web.

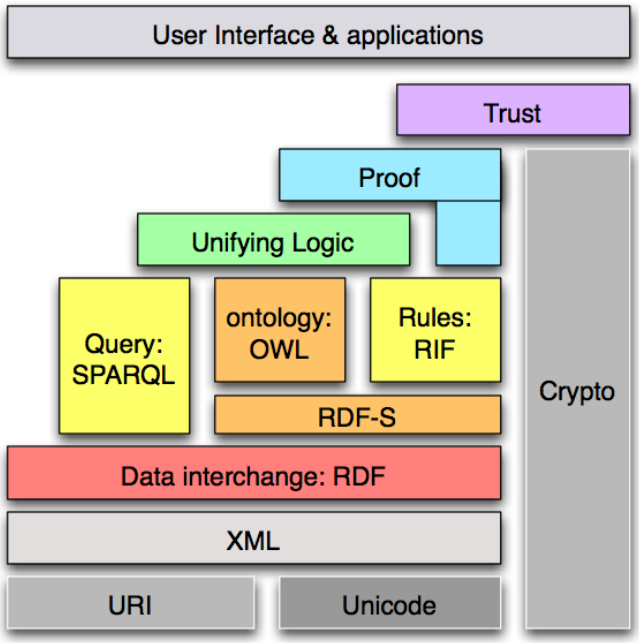


Figura 3. Modelo em camadas da Web Semântica. Fonte: (Berners-Lee, 2006)

Já o RDF *Schema*[8] (RDF-S) define mecanismos de modelagem que permitem a organização desses objetos Web (recursos) em hierarquias. O principal objetivo de RDF e RDF-S é a representação de metadados, sendo, portanto, base de outras linguagens para expressar semântica. Como evolução desta, temos a camada de ontologias que possui poder

[3] Unicode é um padrão que "fornece um único número para cada caractere, não importa a plataforma, não importa o programa, não importa a língua". Mais detalhes em: http://www.unicode.org/
[4] URI ou identificador de recursos uniforme é um conjunto de caracteres utilizado para identificar ou nomear um recurso da Web. Fonte: http://en.wikipedia.org/wiki/Uniform_Resource_Identifier
[5] Mais detalhes na Seção 2.1.2 deste Capítulo.
[6] *Resource Description Framework* é um *framework* para descrição de recursos/metadados. Mais detalhes na Seção 2.1.3 deste Capítulo.
[7] O modelo de dados definido pelo RDF é independente de sintaxe, porém possui uma sintaxe baseada em XML.
[8] Mais detalhes na Seção 2.4 deste Capítulo.

maior de representação de relacionamentos entre recursos da Web do que RDF e RDF-S. OWL[9] foi escolhido pela W3C como linguagem padrão para representação de ontologias.

No mesmo nível da camada de ontologias existem dois novos padrões que não existiam nos primeiros modelos de arquitetura da Web Semântica (Berners-Lee, 2000). O padrão de consultas e o padrão de regras foram incluídos pela necessidade de formas mais complexas de representação e consulta de dados que surgiu com a popularização de ontologias e dados RDF. SPARQL é atualmente candidato a recomendação[10] da W3C como padrão de linguagens para consultas de dados RDF. Já as regras surgiram como complemento (ou alternativas) às ontologias. Berners-Lee cita RIF[11] (*Rule Interchange Format)* (Boley & Kifer, 2007), cujo processo foi iniciado, recentemente, como possível padrão de regras para Web Semântica.

Finalizando, sob essas três camadas temos camadas que ainda estão em desenvolvimento, como a camada lógica (*unifying logic)* que visa à definição de semântica em linguagem formal, possibilitando a inferência através de agentes inteligentes e outros tipos de raciocinadores. Já a camada de provas (*proof)* utiliza os raciocínios e regras de inferências realizados na camada de lógica para avaliar ou explicar algo. Por último temos a camada de confiança (*trust)* em que existem mecanismos para confiar (ou não) em determinada prova ou até mesmo determinar o grau de confiança do conhecimento obtido. A utilização de criptografia e assinaturas digitais (*crypto)* garante a idoneidade e segurança das informações e por isso permeiam todas as camadas já citadas.

A seguir, apresentaremos uma visão geral dos padrões envolvidos nesse modelo de camadas da arquitetura da Web Semântica, enfocando as características que darão o suporte necessário para a viabilidade e a concretização da mesma.

---

[9] *Ontology Web Language* é uma linguagem de ontologia construída sob RDF. Mais detalhes sobre OWL na Seção 2.3 deste Capítulo.

[10] Última recomendação foi 14/07/2007. Mais detalhes em http://www.w3.org/TR/rdf-sparql-query/

[11] Já existe um grupo de trabalho fazendo pesquisas para um padrão comum de regras. Mais informações sobre este grupo de trabalho em http://www.w3.org/2005/rules/

2.2 XML

Uma solução proposta para a automatização e a viabilização da Web Semântica é o uso de metadados para descrever os dados contidos na Web. Esses metadados nada mais seriam do que dados que descrevem recursos Web. XML permite aos usuários adicionar uma estrutura arbitrária aos seus documentos (criando suas próprias *tags*), porém nada diz a respeito do que essas estruturas significam. Outra característica importante de XML é sua interoperabilidade. Isso porque permite que os dados sejam portáveis entre diferentes aplicações e sistemas, uma vez que XML é independente de plataforma. Além disso, tornou-se um padrão para a representação de dados tanto em sistemas fortemente acoplados como fracamente acoplados (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2006). XML encapsula os dados em *tags* assim como o HTML, porém a diferença é que em XML as *tags* se relacionam com o significado do texto encapsulado, enquanto que no HTML elas se relacionam na forma como eles serão apresentados ou visualizados. Na Figura 4 mostraremos um exemplo comparando HTML e XML.

Outra característica do XML é sua extensibilidade: é possível escrever suas próprias *tags* para descrever o conteúdo de um específico tipo de texto, por exemplo, bem como definir *schemas* que descrevem a estrutura de um tipo particular de documento XML. Em HTML, isso não ocorre: as *tags* são fixas. No XML são especificados em *schemas* quais *tags* podem ser usadas e onde elas podem ocorrer. Assim, diz-se que todo documento XML que segue essas especificações está conforme o determinado *schema*. Além disso, XML não inclui instruções de formatação/visualização. Assim, mantendo os dados separados das instruções de apresentação, podem-se expor os mesmos dados de diferentes maneiras. Em HTML, isso não ocorre visto que não existe essa separação entre dados e instruções de formatação/validação.

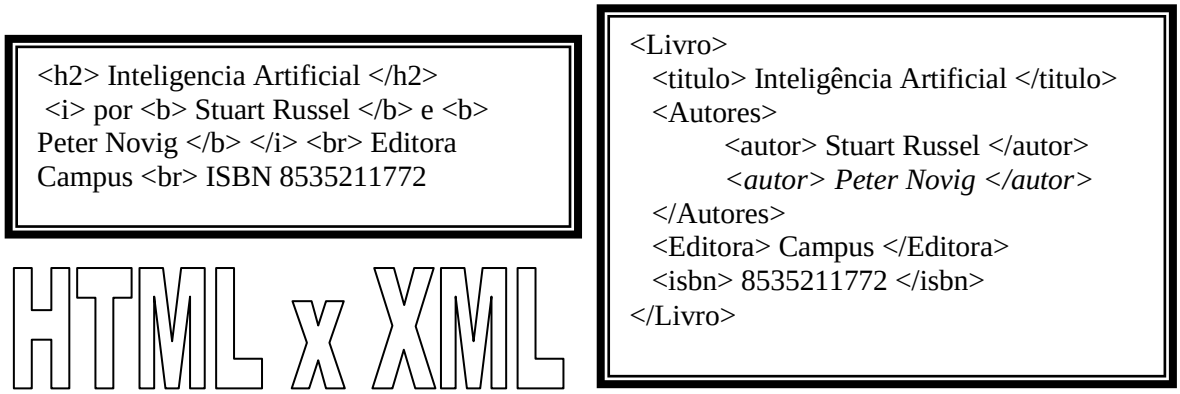


Figura 4. Comparativo dos mesmos dados em formatos diferentes: HTML x XML

2.3 RDF

Visto que o XML nada fornece sobre a semântica de seus dados (sua semântica só é compreensível para os humanos), faz-se necessário uma estrutura ou modelo que consiga melhor descrever dados, mesmo que de forma simples. RDF, sigla de *resource description framework,* como o próprio nome diz, trata-se de um modelo para descrição e intercâmbio de recursos, no caso metadados. É baseado em XML, como forma de maximizar sua interoperabilidade no heterogêneo mundo da Web. Assim, apoiando-se no XML, seu principal objetivo é definir um mecanismo para descrição de recursos sem fazer suposições sobre um domínio particular. Para atingir essa neutralidade, RDF possui um modelo de dados que é uma forma de representar expressões RDF, independente de sintaxe. Vale ressaltar que XML é apenas uma das sintaxes possíveis de RDF, outras podem surgir. Desse modo, duas expressões RDF são equivalentes "se somente se suas representações de modelos de dados forem iguais." (Lassila & Swick, 1999). Essa definição de equivalência permite alguma variação sintática em uma expressão sem, porém, alterar o significado. Ou seja, é possível ter dados RDF com sintaxes diferentes, porém seu modelo de dados é único, o que permite que sejam semanticamente equivalentes.

RDF se baseia na identificação de recursos através de identificadores Web, os URIs, e na descrição destes em simples termos de propriedades (atributo-valor) – uma coleção de triplas no modelo sujeito, objeto e predicado. Esse esquema pode ser entendido através da explicação dos seguintes conceitos:

- Recurso - Qualquer coisa descrita por uma expressão RDF é chamada de recurso. Estes podem ser uma página Web ou parte dela, um elemento XML ou qualquer outro tipo de recurso Web que possui um URI.
- Propriedade - é um recurso que possui um nome e pode ser usado como uma relação entre recursos. Cada propriedade tem um significado específico, definem seus valores permitidos, os tipos de recursos que ela pode descrever e suas relações com outras propriedades.
- Sentença - um rercurso específico combinado com uma propriedade mais o valor dessa propriedade consiste em uma sentença RDF (*statement*). Essas três partes são chamadas respectivamente de sujeito, predicado e

objeto. Lembrando que um objeto de uma sentença também pode ser outro recurso.

Uma forma de representação visual de uma sentença RDF é através de grafos. Na Figura 5 temos um exemplo de um grafo descrevendo sentenças. Elas relacionam-se com http://www.w3.org/People/EM/contact#me, que representa uma pessoa cujo nome é "*Eric Miller"*, possui como email "*em@w3.org"* e tem título de "*Dr"*. Note que cada relação segue as triplas mencionadas (sujeito, predicado, objeto). Os nós verdes representam recursos, os arcos representam as propriedades e os retângulos laranjas representam os literais. O arco sempre se direciona do sujeito ao objeto. No caso da sentença que representa o nome da pessoa, o recurso *#me* é o sujeito, o predicado é *#fullName* e o objeto é o valor "*Eric Miller*".

Figura 5. Exemplo de um grafo de sentenças RDF descrevendo uma pessoa: *Existe uma pessoa cujo nome é Eric Miller e email é em@w3.org e possui título Dr.* Fonte: (Manola & Miller, 2004).

Já na Figura 7 temos o mesmo exemplo da Figura 6 na sintaxe RDF/XML (Beckett & McBride, 2004). Trata-se de uma sintaxe baseada em XML para recursos RDF. Todos os nossos exemplos serão apresentados nessa sintaxe. No caso da Figura 7, o recurso http://www.w3.org/People/EM/contact#me representa uma pessoa através da linha 4. As linhas abaixo da linha 4 (5 a 7) representam a relação do recurso com outros atributos, tais como: nome, correio e titulação, respectivamente.

```
1<?xml version="1.0"?>
2<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3         xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
4  <contact:Person df:about="http://www.w3.org/People/EM/contact#me">
5   <contact:fullName>Eric Miller</contact:fullName>
6    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
7    <contact:personalTitle>Dr.</contact:personalTitle>
8  </contact:Person>
9</rdf:RDF>
```

Figura 6. Sintaxe RDF/XML do grafo RDF de exemplo da Figura 5. Fonte: (Manola & Miller, 2004).

Como vantagens do RDF e dessa sintaxe podemos ressaltar a independência e a facilidade de intercâmbio desse modelo de dados, visto que, por estar baseado em XML, pode se beneficiar da interoperabilidade dessa linguagem. Além dessa troca e reuso de metadados estruturados, RDF fornece contribuições como a padronização de uma sintaxe que pode, por exemplo, ser usada para escrever ontologias. Outra vantagem está na simplicidade e flexibilidade das triplas combinadas com o uso de URIs para identificações globais únicas que permitem as buscas menos custosas e mais simples.

Uma desvantagem do RDF é seu baixo poder de expressividade (falta, por exemplo, conectivos de negação, conjunção e disjunção). Mais informações sobre RDF e seu modelo de dados podem ser encontradas em (Manola & Miller, 2004), (Brickley & Guha, 2004), (Beckett & McBride, 2004), (Lassila & Swick, 1999).

2.4 RDF-*Schema*

Para muitas aplicações, o RDF, com sua habilidade de transportar triplas através da Web (transportar levando em conta a interoperabilidade do XML), é suficiente. Porém, outros tipos de aplicações podem necessitar mais do que isso. Por exemplo, suponha que temos uma propriedade que represente um autor de livro e necessita-se restringir o valor dessa propriedade para que sempre referencie a uma pessoa e não qualquer tipo de objeto (uma casa, por exemplo, não faria sentido). Só utilizando RDF isso não seria possível.

RDF-*Schema* ou RDF-S (Brickley & Guha, 2004) serve para possibilitar alguns tipos de restrições que o modelo de representação de recursos de RDF não consegue realizar.

Ou seja, é através do RDF-S que essas validações de valores a certas propriedades podem ser alcançadas.

Segundo (Dodds, et al., 2001), enquanto a especificação do modelo de dados RDF e sua sintaxe definem como documentos XML podem ser construídos para poder se comunicar com RDF, RDF-*Schema* se preocupa em garantir que a estrutura de documentos RDF/XML possua um "correto significado". Entende-se este "correto significado" no sentido de que tanto as propriedades de um recurso quanto os valores usados nessas propriedades sejam coerentes (no caso do exemplo citado, fazer com que a propriedade autor de livro sempre se referencie a uma pessoa e não a uma casa).

Desta forma, RDF-S estende a linguagem RDF com um vocabulário para descrever classes e objetos, as suas propriedades e o tipo de dados dessas propriedades. Ou seja, ele permite que recursos descritos em RDF possam ser organizados em uma hierarquia de classes (a propriedade *rdf:subClassOf* denota esse relacionamento hierárquico, enquanto que a propriedade *rdf:type* indica as classes das quais um recurso é instância).

Pode-se, com isso, perceber que RDF *Schema* possui algumas características que possibilitam a construção de ontologias simples (basicamente na relação classe/subclasse). Porém, para a construção de ontologias mais complexas e com mais funcionalidades, ele fica limitado. Isso ocorre devido à sua expressividade limitada, pois se utiliza apenas de hierarquias de subclasses e propriedades com definições de domínio e faixa de valores para estas. Acontece que existem situações que é necessária maior expressividade para representar informações mais complexas no contexto da Web Semântica. Somente hierarquia de classes e restrições de valores não são suficientes. Para completar esta lacuna, surgiram as linguagens para definição de ontologias.

2.5 Ontologias

Termo originado na Filosofia, ontologia significa o estudo da natureza da existência ou do grego conhecimento do ser. Ela "se destina a descrever categorias ou relações de existência entre entidades..." (Wikipedia, Ontology, 2007). Ou seja, ela é utilizada para descrever conceitos e seus relacionamentos.

Geralmente uma ontologia define termos utilizados para descrever uma área de conhecimento. São utilizadas por pessoas, banco de dados e aplicações que necessitam "...compartilhar informação de domínio (sendo domínio uma área de conhecimento específica como medicina, ferramentas de manufatura, bens imobiliários, conserto de automóveis, gerenciamento financeiro, etc.)" (Heflin, 2004). Por ser utilizada em diferentes áreas de pesquisa, ela pode possuir algumas definições e características diferentes, porém, na computação, existe certo consenso.

Para (Gruber, 1995), uma ontologia é uma "...especificação explícita de uma conceitualização...". Evoluindo essa definição, (Borst, 1997) define ontologia como "uma especificação formal e explícita de uma conceitualização compartilhada". Entendemos o termo *formal* no sentido de ser apropriado para um processamento automático (seja por agentes ou aplicações). *Explícita* pelo fato de seus conceitos, propriedades, relações e restrições estarem claramente definidos. Por fim, *compartilhada* no sentido de um consenso sobre esses conceitos. Já para (Noy & McGuinness, 2001), uma ontologia é "uma descrição explícita e formal de conceitos de um domínio, propriedades descrevendo várias características e atributos de cada conceito e restrições."

De maneira mais sucinta, (Horridge, Knublauch, Rector, Stevens, & Wroe, 2004) dizem que uma ontologia descreve os conceitos de um domínio e o relacionamento que existe entre esses conceitos, sendo assim utilizada para "...capturar conhecimento sobre algum domínio de interesse".

(Noy & McGuinness, 2001) elencou algumas outras razões para o uso de ontologias. São elas:

- "Para compartilhar entendimento comum da estrutura de informações entre pessoas e agentes de software".
- "Para permitir o reuso do conhecimento de domínio".
- "Tornar explícitas as suposições de domínio."
- "Separar conhecimento de domínio de conhecimento operacional."
- "Analisar o conhecimento de domínio".

Compartilhar entendimento comum é um dos objetivos mais freqüentes no desenvolvimento de ontologias (Gruber, 1995). Por exemplo, uma única ontologia pode servir

de base para diferentes sites que, com isso, poderão ser consumidos por agentes que desejam extrair ou agregar informações desses diversos sites. Essa permissão de reuso do conhecimento de domínio está no sentido de que uma ontologia poderia ser reutilizada ou especializada em outros domínios totalmente diversos. Já o fato de tornar explícito facilita mudanças e o aprendizado por novas pessoas ou agentes. Por fim, a análise do conhecimento do domínio é extremamente valiosa na tentativa de reusar e especializar ontologias já existentes (McGuinness, Fikes, Rice, & Wilder, 2000).

Em termos gerais, ontologias fornecem um meio de capturar o conhecimento compartilhado de termos formais e explícitos que possam ser usados, tanto por pessoas quanto por máquinas, no auxílio da busca e troca de informações. Isso reflete a visão da Web Semântica de Tim Berners-Lee (Berners-Lee, Hendler, & Lassila, 2001) como uma infra-estrutura que permita essa cooperação e o desenvolvimento de ontologias, que fornece um mecanismo para a construção da semântica da Web Semântica. Para facilitar esse objetivo foi necessário o desenvolvimento de formas adequadas de expressar essas ontologias. Isso levou ao surgimento de algumas linguagens para definição de ontologias como OIL[12], DAML + OIL[13] e OWL.

2.6 OWL (*Ontology Web Language*)

OWL surgiu a partir de uma evolução de linguagens como DAML+OIL com a necessidade de um padrão de linguagem de ontologias para a Web Semântica. (McGuinness & Harmelen, 2004) ressalta que para a Web Semântica "a primeira camada acima de RDF requer uma linguagem de ontologia que possa descrever formalmente o significado das terminologias utilizadas em documentos Web. Se máquinas têm a expectativa de inferir ou executar tarefas úteis sobre esses documentos, essa linguagem deve ir além das semânticas básicas de RDF *Schema".*

Em fevereiro de 2004, OWL surgiu como uma recomendação de um grupo de

[12] Mais informações em http://www.ontoknowledge.org/oil/
[13] Mais informações em http://www.daml.org/2001/03/daml+oil-index.html

trabalho[14] da W3C como uma linguagem de ontologias padrão construída sob RDF e RDF-*Schema*, através da publicação de seis documentos. São eles:

- Visão Geral OWL (McGuinness & Harmelen, 2004) – fornece uma introdução mostrando suas características com uma breve descrição.
- Guia de OWL (Smith, Welty, & McGuinness, 2004) – demonstra o uso de OWL através de um exemplo.
- Referência OWL (Bechhofer, et al., 2004) – fornece uma descrição sistemática de todas as primitivas de modelagem de OWL.
- Semântica e Sintaxe OWL (Patel-Schneider, Hayes, & Horrocks, 2004) – apresenta a definição formal final da linguagem.
- Requisitos e Casos de Uso de OWL (Heflin, 2004) – apresenta um conjunto de casos de uso para uma linguagem de ontologia para a Web e compila um conjunto de requisitos para OWL.

Para (Kalyanpur, 2006), OWL é uma linguagem única, pois é a primeira linguagem de ontologia cujo projeto é baseado na arquitetura Web. Por exemplo, OWL é aberto (não é proprietário), utiliza URIs para identificar de forma não ambígua recursos na Web (similar a RDF e RDF-S), suporta a ligação de termos entre ontologias, tornando possível referências cruzadas e reuso de informações, e possui uma sintaxe XML, o RDF/XML (Beckett & McBride, 2004), para intercâmbio facilitado de dados.

A linguagem OWL fornece três sub-linguagens de expressividade progressiva com o objetivo de atender a diferentes necessidades dos usuários ou desenvolvedores. São elas (Smith, Welty, & McGuinness, 2004):

- OWL *Lite* – destinada àqueles usuários que necessitam apenas de uma hierarquia básica de classificação e restrições simples. Por exemplo, OWL *Lite* suporta restrições de cardinalidade, porém a cardinalidade máxima ou mínima assume apenas os valores 0 ou 1. Para (Freitas, 2003), OWL *Lite* é ideal para usuários iniciantes e desenvolvedores que preferem *frames* a lógica de descrições. É uma linguagem de fácil compreensão para usuários

[14] Mais informações sobre este grupo de trabalho em http://www.w3.org/2001/sw/WebOnt/

e fácil de implementar para desenvolvedores. Porém a desvantagem é a sua expressividade limitada.

- OWL DL – destinada àqueles usuários que querem o máximo de expressividade sem perder a completude computacional (a garantia de que todas as restrições serão computadas) e decidibilidade (que todas as computações irão terminar em tempo finito). Para isso, OWL DL (abreviação de Lógica Descritiva - *Description Logic*) inclui todos os construtores da linguagem OWL, porém só podem ser utilizados com restrições, tais como: separação de tipos (por exemplo, uma classe não pode ser instância e propriedade ao mesmo tempo). Ou seja, a expressividade é maior que OWL-*Lite,* pois classes podem ser construídas por união, interseção, complemento, etc. (Antoniou & Harmelen, 2004) cita que a principal vantagem do OWL DL é que ela "...permite um suporte ao raciocínio eficiente...", enquanto que a desvantagem é a "...perda da total compatibilidade com RDF...". Essa perda de compatibilidade ocorre visto que para tornar OWL decídivel algumas restrições sobre a linguagem RDF são necessárias. Isto significa que nem todo documento RDF é um documento OWL-DL, apesar de o inverso ser verdade.
- OWL *Full* – destinada aos usuários que querem o máximo de expressividade e liberdade sintática de RDF sem garantias computacionais. Não cabem, por exemplo, as restrições de separação de tipos da versão anterior e é possível manipular e modificar as meta classes. Ou seja, sua vantagem é a completa compatibilidade com RDF e é possível a combinação das primitivas da linguagem OWL de maneira livre com RDF e RDF-*Schema*, possibilitando a criação de novas metaclasses. Porém a desvantagem dessa liberdade de expressão é a possível perda da decidibilidade. (Antoniou & Harmelen, 2004) afirma que o problema OWL *Full* é "...que a linguagem se torna tão poderosa quanto indecidível, eliminando qualquer esperança de um suporte de raciocínio completo (ou eficiente)." Porém, recentemente, (Motik, 2005) mostrou que sob certas condições, OWL *Full* pode ser decidível.

Vale ressaltar que cada uma dessas sub-linguagens é uma extensão de seu predecessor mais simples. (Smith, Welty, & McGuinness, 2004) exemplifica essa característica assim:

- Toda ontologia OWL *Lite* é uma ontologia OWL DL;
- Toda ontologia OWL DL é uma ontologia OWL *Full;*
- Toda conclusão válida em OWL DL é uma conclusão válida em OWL *Full.*

2.6.1 Estrutura de Ontologias OWL

A sintaxe RDF/XML (Beckett & McBride, 2004) de uma ontologia típica em OWL inicia-se com uma declaração, em RDF, contendo os *namespaces* que serão utilizados na ontologia. *Namespaces* são mecanismos para qualificar elementos e nomes de atributos com intuito de resolver conflitos de nomeação de elementos de um documento XML (Bray, Hollander, Layman, & Tobin, 2006). Na Figura 7, temos um exemplo. As duas primeiras declarações identificam o *namespace* associado à ontologia que está sendo definida exemplificada. A primeira define o *namespace default*, indicando que nomes qualificados sem prefixo referem-se a essa ontologia. A segunda identifica o *namespace* da ontologia atual com o prefixo *mia*. As declarações restantes identificam os *namespaces* das declarações de OWL, RDF, RDF *Schema* e XML *Schema*.

```
<rdf:RDF
        xmlns="http://www.unifor.br/mia#"
        xmlns:mia ="http://www.unifor.br/mia#"
        xmlns:owl ="http://www.w3.org/2002/07/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/2001/XMLSchema#">
```

Figura 7. Exemplo de *namespace* padrão de uma ontologia OWL

Uma vez estabelecidos os *namespaces*, inclui-se um conjunto de assertivas sobre a ontologia (chamados de cabeçalhos OWL) agrupado sob o elemento *owl:Ontology*, como comentários, controle de versões, inclusões de outras ontologias, etc. Na Figura 8 temos um exemplo de um cabeçalho OWL com alguns desses exemplos. O primeiro elemento *rdfs:comment* refere-se a algum comentário sobre a ontologia modelada. Similar a esse

elemento existe a *rdf:label* utilizada para escrever um nome ou uma anotação. O elemento *owl:priorVersion* é utilizada para controle de versões, apontando para a última versão da ontologia atual. Já o elemento *owl:imports* representa a importação de uma ontologia externa. Essa importação permite que quaisquer definições existentes na ontologia importada possam ser utilizadas na ontologia atual. (Antoniou & Harmelen, 2004) ressalta ainda que *owl:imports* possui uma propriedade transitiva: se a ontologia X importa a ontologia Y e a ontologia Y importa a ontologia Z, então a ontologia X importa a ontologia Z.

```
<owl:Ontology rdf:about="">
    <rdfs:comment>Comentario sobre a ontologia</rdfs:comment>
     <owl:priorVersion rdf:resource="http://www.unifor.br/mia-old">
    <owl:imports rdf:resource="http://xmlns.com/foaf/0.1/">
    <rdf:label>Descricao sobre a ontologia</rdfs:label>
</owl:Ontology>
```

Figura 8. Exemplo de um cabeçalho OWL

Já em relação às classes[15], estas podem ser construídas de várias formas em OWL: por herança, união, interseção, complemento, pela enumeração de instâncias ou por restrições de propriedades (Smith, Welty, & McGuinness, 2004), (Freitas, 2003). Elas são definidas usando o elemento *owl:Class*, o qual é uma subclasse de *rdfs:Class* (Brickley & Guha, 2004). O elemento *owl:Class* define um grupo de indivíduos que "permanecem" juntos por compartilharem algumas propriedades. Por exemplo, João e Maria são membros da mesma classe *Pessoa*. Outra forma de organização é através de uma hierarquia especializada através do uso de *rdfs:subClassOf.*

Por exemplo, a classe Pessoa pode ser definida como uma subclasse da classe *Mamífero* e uma classe *Homem* é definida como uma subclasse de *Pessoa*. Deste modo, um raciocinador poderá deduzir que se *Homem* é subclasse de *Pessoa*, então ele também é subclasse de *Mamífero*. Na Figura 9 podemos visualizar esse exemplo.

[15] O termo conceito também é utilizado às vezes no lugar de classe. Classes geralmente são representações concretas de conceitos (Horridge, Knublauch, Rector, Stevens, & Wroe, 2004). Porém em nosso trabalho vamos utilizar apenas o termo classe.

```
<owl:Class rdf:ID="Pessoa">
    <rdfs:subClassOf rdf:resource="#Mamifero"/>
</owl:Class>
```

```
<owl:Class rdf:ID="Homem">
      <rdfs:subClassOf rdf:resource="#Pessoa"/>
</owl:Class>
```

Figura 9 Exemplo de hierarquia de classes em OWL

Outra forma de construir classes é através de enumerações e restrições de propriedades. Na Figura 11 temos alguns exemplos. No quadro esquerdo da Figura 10 temos uma classe representando uma coleção de continentes através do elemento *owl:oneOf.* No quadro direito temos a construção de uma classe com o uso de restrições de propriedades. No exemplo, representamos a classe dos carros que possuem cor azul. Ou seja, uma restrição (*owl:Restriction)* na propriedade *#temCor (owl:onProperty)* limita aos valores da classe *Azul* (*owl:AllValuesFrom).*

```
<owl:Class rdf:ID="Continentes">
  <owl:oneOf rdf:parseType="Collection">
      <owl:Thing rdf:about="#Eurasia"/>
      <owl:Thing rdf:about="#Africa"/>
      <owl:Thing rdf:about="#America"/>
      <owl:Thing rdf:about="#Australia"/>
      <owl:Thing rdf:about="#Antartica"/>
    </owl:oneOf>
</owl:Class>
```

```
<owl:Class rdf:ID="Carro_Azul">
   <rdfs:subClassOf rdf:resource =”#Carro”>
   <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty  rdf:resource="#temCor"/>
     <owl:allValuesFrom rdf:resource="#Azul"/>
   </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

Figura 10. Exemplos de novas classes criadas utilizando construtores OWL: uma enumeração e uma restrição

Existem outros tipos de restrições como *owl:someValuesFrom, owl:hasValue, owl:minCardinality, owl:MaxCardinality, owl:cardinality.* Além de enumerações, existem em OWL outras formas de combinações de classes como: união (*owl:unionOf),* interseção (*owl:intersectionOf*) e complemento (*owl:complementOf).* Maiores detalhes podem ser encontrados em (McGuinness & Harmelen, 2004), (Bechhofer, et al., 2004) e (Noy & McGuinness, 2001).

Os membros de uma classe OWL geralmente denominam-se indivíduos ou

instâncias. Já as propriedades[16] são geralmente relações binárias entre esses indivíduos (Horridge, Knublauch, Rector, Stevens, & Wroe, 2004), (Smith, Welty, & McGuinness, 2004) e se dividem em dois tipos:

- Propriedades de tipos de dados (*owl:DatatypeProperty)*: representam relações entre instância de uma classe e um literal representado em RDF ou um tipo de dados de XML *Schema*[17].
- Propriedade de objetos (*owl:ObjectProperty)*: representam relações entre instâncias de duas classes.

Na Figura 11 temos dois exemplos dessas propriedades OWL. Assim, temos *temNome* como uma propriedade onde os elementos *rdfs:domain* e *rdfs:range* representam respectivamente qual o domínio da propriedade (isto é, a que classe ela pertence) e qual a faixa de valores permitidos. No quadro esquerdo, temos *temNome* como uma propriedade de objetos dizendo que é uma relação entre instâncias da classe *#Pessoa* com instâncias da classe *#Nome.* Já no quadro direito *temNome* é uma propriedade de tipo de dados onde a faixa de valores referencia um tipo de dado (no caso *string).*

```
<owl:ObjectProperty rdf:ID="temNome">
  <rdfs:domain rdf:resource="#Pessoa" />
  <rdfs:range rdf:resource="#Nome" />
</owl:ObjectProperty>
```

```
<owl:DatatypeProperty rdf:ID="temNome">
   <rdfs:domain rdf:resource="#Pessoa" />
    <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

Figura 11. Exemplo de dois tipos de propriedades OWL

É possível também descrever em OWL propriedades inversas (*owl:inverseOf),* equivalentes (*owl:equivalentProperty),* funcionais (*owl:FunctionalProperty),* transitivas (*owl:TransitiveProperty*), simétricas (*owl:SymmetricProperty*), entre outras.

Por fim, existem outros dois elementos pré-definidos existentes em OWL chamados *owl:Thing* e *owl:Nothing*. O primeiro representa a classe mais geral, na qual contem todos os elementos, isto é, toda classe OWL é uma subclasse de *owl:Thing.* Já o segundo trata-se de uma classe vazia, o que significa que esta é uma subclasse de todas as outras classes OWL.

[16] Na Lógica Descritiva, propriedades também são conhecidas como papéis (*roles).* Mais detalhes no Capítulo 4 na Seção 4.2

[17] Maiores informações em http://www.w3.org/XML/Schema

2.7 SPARQL

SPARQL é uma linguagem de consulta (candidata a recomendação W3C) para recuperação de grafos RDF (Prud'hommeaux & Seaborne, 2007). Diz-se recuperação de grafos, pois sua busca segue um padrão similar às triplas do modelo de dados de RDF, porém com a opção de consultar variáveis nas posições de sujeito, objeto e predicado.

Atualmente SPARQL possui três especificações: uma especificação da linguagem de consulta básica (Prud'hommeaux & Seaborne, 2007), outra descrevendo um formato XML para a serialização dos resultados de uma consulta SPARQL (Beckett & Broekstra, 2007) e por último, uma especificação de um protocolo de acesso a dados utilizando WSDL 2.0[18] que permite acesso remoto a bases de dados descritos em RDF (Clark K. G., 2006).

Apesar de recente, já existem alguns trabalhos e estudos de caso em cima de SPARQL como (OpenLink Software, 2007); (ESW Wiki, 2007); (Web Clipboard, 2007). ARQ (ARQ, 2007), por exemplo, é um "motor de consultas" (*query engine*) que fornece suporte à linguagem SPARQL para o *framework* Jena (Jena, 2007). Joseki (Joseki, 2007) já é um servidor de consultas SPARQL que utiliza ARQ e o protocolo de SPARQL (Clark K. G., 2006).

2.7.1 Exemplo

(McCarthy, 2005) apresenta um exemplo da utilização de SPARQL sobre o site Planet RDF. Este site agrega diversos blogs com conteúdo relacionado à Web Semântica e armazena em um documento[19] RDF os autores destes blogs (Beckett, Biddulph, Dumbill, & McCarthy, 2007) periodicamente.

O modelo em grafo do documento, apresentado na Figura 12, utiliza duas ontologias comumente importadas em outras ontologias: FOAF (Brickley & Miller, 2007) e

[18] Mais informações em http://www.w3.org/TR/wsdl20/
[19] Disponível em http://journal.dajobe.org/journal/2003/07/semblogs/bloggers.rdf

Dublin Core (Hillmann, 2005). Elas são utilizadas para descrever o nome, o título do blog, a url do blog e uma descrição RSS[20] de cada autor de blog.

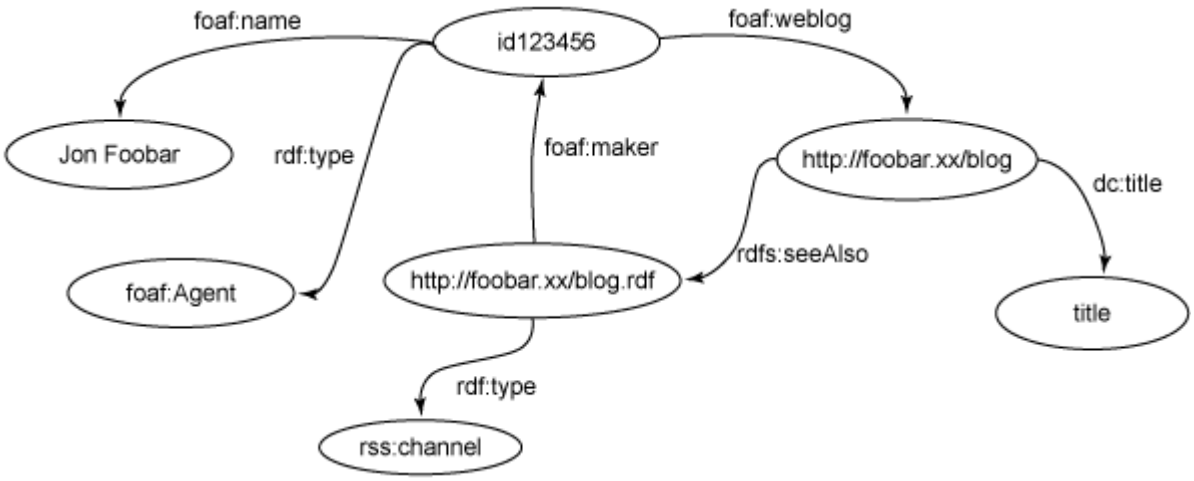


Figura 12. Estrutura básica do grafo de autor de blog agregado no site Planet RDF (Beckett, Biddulph, Dumbill, & McCarthy, 2007). Fonte: (McCarthy, 2005).

A Figura 12 mostra apenas o grafo básico de um autor de blog. O modelo completo do documento repete essa estrutura para cada blog que é incluído.

Na Figura 13 temos um exemplo de uma consulta SPARQL simples sobre esse modelo de dados, representando a busca pela URL do blog de "*Jon Foobar*".

```
1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 SELECT ?url
3 FROM  <bloggers.rdf>
4 WHERE  {
5    ?contributor foaf:name "Jon Foobar" .
6    ?contributor foaf:weblog ?url .
7 }
```

Figura 13. Consulta SPARQL para encontrar a URL de um autor de blog. Fonte: (McCarthy, 2005).

O termo PREFIX é equivalente a um *namespace* XML: ele associa uma nome abreviado para a URL da ontologia de FOAF (no caso, *foaf:*) – linha 1 da Figura 13.

O início da consulta propriamente dita é a cláusula SELECT na linha 2 da Figura 13. Similar a uma consulta SQL em banco de dados relacionais, essa cláusula é utilizada para

[20] Descrição ou fonte RSS (RSS *feeds*) são resumos de conteúdo em XML muito utilizados para acessar notícias ou blogs na Web. Mais informações em http://www.rssboard.org/rss-specification

definir que itens de dados serão retornados. No caso do exemplo, a URL do blog é representada pela variável *?url.* Variáveis SPARQL geralmente são prefixadas com o símbolo (*?).*

Do mesmo modo, a cláusula opcional *FROM* identifica qual a base de dados que será utilizada na consulta SPARQL. Podem existir diversas cláusulas *FROM* para associar a variadas bases de dados. No exemplo, está especificado o documento RDF do site Planet RDF (linha 3 da Figura 13).

Por fim, a cláusula *WHERE* contem duas triplas, no padrão RDF, utilizando a sintaxe Turtle (Beckett, 2006), na qual identificam o padrão de triplas de sentenças no formato RDF que se almeja encontrar. Note o padrão (sujeito, predicado, objeto) nas linhas 5 e 6 da Figura 13: (*?contributor*, *foaf*:*name*, "*Jon Foobar*") e (*?contributor, foaf:weblog, ?url*). No caso, um nó com a propriedade *foaf:name* com valor "Jon Foobar" será associado à variável *?contributor.* Assim, no exemplo, *?contributor* associa-se ao id123456 da Figura 12 (id123456, *foaf:name, "Jon Foobar").* Com isso, a segunda tripla (linha 6) associa o objeto da propriedade *foaf:weblog* a variável *?contributor,* levando à associação da variável *?url* com o valor da propriedae *foaf:weblog* (*?contributor, foaf:name, "http://foobar.xx/blog"),* resultando na solução da consulta.

SPARQL possui outras cláusulas como OPTIONAL, UNION, LIMIT, OFFSET, FILTER, ASK, DESCRIBE, GRAPH. Mais detalhes podem ser encontrados em (Prud'hommeaux & Seaborne, 2007), (Clark K. G., 2005) e (Beckett, 2006).

Apesar de SPARQL ter sido projetado apenas para ser uma linguagem de consulta e recuperação de dados, já existem alguns trabalhos que estendem SPARQL tornando-o uma linguagem também de atualização de dados como em (Seaborne & Manjunath, 2007).

3 PREFERÊNCIAS E TRABALHOS RELACIONADOS

Neste terceiro capítulo discorremos sobre o conceito de preferência e algumas formas de modelá-las que encontramos em nossa pesquisa bibliográfica. Além disso, mostramos como estão sendo realizados alguns dos trabalhos relacionados ao conceito de preferências, ressaltando a problemática destes que contribuiu para a proposta do modelo de representação de OWLPREF. Com isso, na Seção 3.1 abordamos o conceito e as formas de modelagens de preferências. Na Seção 3.2 fazemos um breve relato dos trabalhos relacionados. Na Seção 3.3 apresentamos algumas linguagens de consultas de preferências. Por fim, na Seção 3.4, finalizamos destacando os pontos que contribuíram para a nossa proposta.

3.1 Noções de Preferência

3.1.1 Conceito

O conceito de preferência faz parte do dia-a-dia de todas as pessoas. Às vezes preferências são expressas em termos de desejos ou objetivos, mas nem sempre elas podem ser totalmente satisfeitas.

Essa é a principal característica de preferências: quando não são atendidas totalmente, as pessoas geralmente estão preparadas para alternativas diferentes ou um pouco piores do que a ideal. Isso difere do conceito de restrições nas quais alternativas não são aceitas. Por exemplo:

- João tem que se formar em 2007;
- João gostaria de se formar antes de 2007.

Na primeira frase não existe alternativa, isto é, João só pode se formar até o ano de 2007 e não existe outra solução que o satisfaça. Ou seja, João, em hipótese nenhuma, pode se formar depois de 2007. Trata-se assim de uma restrição, também conhecida na literatura

como restrição rígida (*hard constraint).* Porém, na segunda frase, João deixou alternativas "abertas" uma vez que ele **gostaria** de se formar em 2007. Ou seja, sua preferência é se formar naquele ano, mas não necessariamente João **tem** que se formar em 2007. Nesse caso, a formatura poderia até ocorrer em 2008, mas a preferência de João é que se possível seja em 2007. Trata-se agora de uma preferência propriamente dita, onde João define uma situação ideal que pode ser relaxada. Preferência é também chamada por alguns autores de restrição suave (*soft constraint)* (Kießling, 2002).

Segundo (Wikipedia, Preference, 2007), "Preferência é um conceito utilizado nas ciências sociais, particularmente na Economia. Ele assume uma escolha real ou imaginária entre alternativas e a possibilidade de ordenar essas alternativas, baseado na felicidade, satisfação, gratificação, prazer ou utilidade que fornece. Em ciências cognitivas, preferências individuais permitem a escolha de objetivos". Baseado nesse conceito e em outros (Koutrika & Ioannidis, 2006), (Boutilier, 1994), podemos dizer que preferências são situações ideais baseadas em diferentes fatores mas que nem sempre podem ser alcançadas ou satisfeitas. Isso permite a possibilidade de ordenar alternativas à situação ideal representada pela preferência.

3.1.2 Modelagem de Preferências

A modelagem de preferências é uma atividade comum em diversos campos de pesquisa: Economia (Armstrong, 1948), Psicologia (Broome, 1991), Ciências Sociais (Barthélemy, Flament, & Monjardet, 1989), Inteligência Artificial (Doyle & Wellman, 1992) Programação Matemática (Perny & Spanjaard, 2005), Medicina (Carrano, 1988) entre outros (Oztürk, Tsoukià, & Vincke, 2005). Nota-se com isso que o campo de pesquisa sobre preferências é extenso e com diversas abordagens. Por ter permeado diferentes áreas, levou a uma variabilidade de modelagens com características próprias e vantagens/desvantagens de acordo com o que se pretende aplicar. Em termos gerais, existem basicamente duas formas de representar ou modelar preferências: a abordagem numérica ou quantitativa e a abordagem simbólica ou qualitativa.

Na abordagem numérica ou quantitativa, geralmente funções de utilidade calculam o grau ou ranking das alternativas e, com isso, a alternativa de maior valor calculado é escolhido como solução (Agrawal & Wimmers, 2000) (Hristidis, Koudas, & Papakonstantinou, 2001). Na Figura 14 temos um exemplo de preferência na abordagem

quantitativa. Esse método fornece um modelo simples para capturar e computar preferências (Bistarelli, Montanari, & Rossi, 1995), (Schiex, 1992), (Ruttkay, 1994).

A abordagem quantitativa também é chamada por alguns autores de ordem total. De acordo com (Oztürk, Tsoukià, & Vincke, 2005), ordem total consiste "de um arranjo de objetos ordenados do melhor para o pior...". Na literatura podemos encontrar outros termos associados à ordem total, como: ordem completa, ordem simples, ordem linear, entre outros.

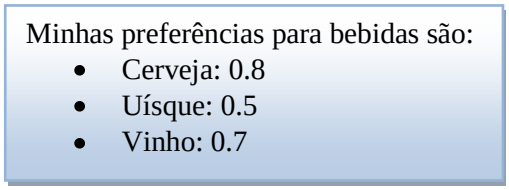


Figura 14. Exemplo de preferência quantitativa: pesos ou funções de utilidade são definidos para as alternativas.

A desvantagem do enfoque quantitativo é que muitas vezes além de não ser necessário expressar preferências sobre todos os elementos (Boutilier, Brafman, Domshlak, Hoos, & Poole, 2004), alguns elementos podem ser naturalmente incomparáveis e que, se a quantidade de elementos for muito grande, pode demandar um esforço até mesmo inviável (Doyle & Wellman, 1994), (Faltings, Torrens, & Pu, 2004), (Ha & Haddawy, 1999) (Boutilier, 1994), (Chomicki, 2003).

Já o enfoque simbólico ou qualitativo representa preferências em termos de ordem parcial e se assemelha mais como realmente as pessoas expressam preferências. Não existem funções de utilidade ou probabilidades associadas às alternativas. Somente se expressa que A é mais importante que B, ou "prefiro A a B", não mencionando preferência sobre todo o conjunto (Dubois, Prade, & Sabbadin, 2001), (Brafman & Tennenholtz, 1997), (Rossi, Venable, & Walsh, 2004). Na Figura 15 temos um exemplo da abordagem qualitativa. Note que a forma de expressão é mais simples e intuitiva do que o enfoque quantitativo. Para as pessoas, dizer que "Eu prefiro Cerveja a Vinho" é mais claro do que "Cerveja tem preferência 0.8 e vinho 0.7". Isso facilita a compreensão e a explicação das mesmas para as pessoas/usuários.

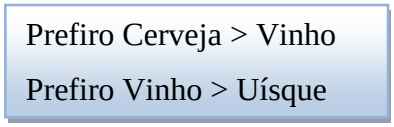


Figura 15. Exemplo de preferência qualitativa: "Eu Prefiro Cerveja a Vinho e Vinho a Uísque"

Além disso, é complicado representar preferências do tipo: “Eu prefiro vinho a cerveja se carne for servida” no enfoque quantitativo. É bem mais natural representar esses tipos de preferências na abordagem qualitativa (Rossi, Venable, & Walsh, 2004), (Boutilier, Brafman, Hoos, & Poole, 1999), (Boutilier, Bacchus, & Brafman, 2001). Essa falta de expressividade da abordagem quantitativa já é bem conhecida na teoria da utilidade (Chomicki, 2003), (Fishburn, 1999).

A representação de preferências de ordem parcial vem sendo pesquisada no contexto de banco de dados e recuperação de informações (Kießling, 2002), (Chomicki, 2003), (Lacroix & Lavency, 1987) onde são utilizadas representações baseadas em atributos e valores (Kießling & Köstler, 2002), (Kießling, Hafenrichter, Fischer, & Holland, 2001).

Alguns trabalhos chamam preferências com ordem parcial de “preferências independentes” (Boutilier, Brafman, Hoos, & Poole, 1999), (Keeney & Raiffa, 1977). Outros as definem como uma relação binária (Chomicki, 2003). Existem também outros tipos de ordem como: semi-ordem, ordem de intervalos, ordem fraca, entre outras (Oztürk, Tsoukià, & Vincke, 2005). Em nossa pesquisa focamos apenas nas ordens parciais e totais por entendermos que são as mais comuns.

3.2 Trabalhos Correlatos

Alguns trabalhos têm desenvolvido linguagens de consulta para domínios específicos que levam em consideração o conceito de preferências. Geralmente o contexto varia desde: banco de dados relacionais (Kießling & Köstler, 2002), (Chomicki, 2003), (Lacroix & Lavency, 1987), dados estruturados (Kießling, Hafenrichter, Fischer, & Holland, 2001), técnicas de relaxamento de consultas (Motro, 1988), pontos não dominados por outros (Börzsönyi, Kossmann, & Stocker, 2001), funções de preferências (Agrawal & Wimmers, 2000), visões ordenadas (Hristidis, Koudas, & Papakonstantinou, 2001).

Outra linha de pesquisa tem focado na representação de preferências, porém geralmente restrita a domínios específicos ou restritos como P3P[21] (Agrawal, Kiernan, Srikant, & Xu, 2005), XML (Kießling, Hafenrichter, Fischer, & Holland, 2001), Serviços Web (Balke & Wagner, 2003), aplicações multimídia (Tsinaraki & Christodoulakis, 2006), (Köhncke & Balke, 2007), (Mylonas & Wallace, 2006) e anti-spam (Kim, Dou, Liu, & Kwak, 2007). (Rossi, Venable, & Walsh, 2004), (Boutilier, Bacchus, & Brafman, 2001), (Boutilier, Brafman, Domshlak, Hoos, & Poole, 2004) utilizam um modelo gráfico qualitativo que capturam e organizam sentenças de preferências condicionais.

Outras pesquisas sobre preferências têm focado em modelos de fórmulas lógicas de preferências (Mantha, 1992), (Chomicki, 2003), (Hansson, 2001), representação visual de preferências condicionais (Rossi, Venable, & Walsh, 2004), (Boutilier, Bacchus, & Brafman, 2001), (Boutilier, Brafman, Hoos, & Poole, 1999).

3.2.1 Linguagens de Consultas de Preferências

Baseado nas duas abordagens da seção 3.1.2, identificamos como modelos mais proeminentes na abordagem qualitativa os que definem relações de preferências como fórmulas lógicas (Chomicki, 2003) e com construtores especiais (Kießling, 2002). Em termos gerais, esses trabalhos propuseram linguagens de consultas que através de um conjunto de construtores pré-definidos, chamados de preferências básicas, e, combinado com construtores de preferências complexas, permitem consultas do tipo: SELECT * FROM TABELAS WHERE <restrições> PREFERRING <preferências>. Essas consultas foram especializadas para dois contextos. São eles:

- Banco de dados relacionais definindo a linguagem Preference SQL (Kießling & Köstler, 2002) que estende a linguagem de consulta padrão SQL (Chamberlin & Boyce, 1974) com a noção de preferências.
- Banco de dados estruturados definindo a linguagem Preference XPATH (Kießling, Hafenrichter, Fischer, & Holland, 2001) que estende a linguagem de consulta de dados XML, o XPATH (Clark & DeRose, 1999), com a noção de preferências.

[21] Mais informações sobre a plataforma de preferências de privacidade em http://www.w3.org/P3P/

3.2.1.1 SPARQL *Preference*

Acontece que essas linguagens de consulta citadas na seção anterior não se adequavam ao contexto dos dados da Web Semântica. Elas não estão relacionadas a linguagens que permitem a descrição de semânticas como OWL/RDF. Contudo, (Siberski, Pan, & Thaden, 2006) e (Abel, Herder, Kärger, Olmedilla, & Siberski, 2007) estabeleceram uma extensão para a linguagem de consulta SPARQL com noção de preferências, seguindo o modelo proposto por (Chomicki, 2003). Eles, através da inclusão de um construtor (chamado de *PREFERRING*) com duas expressões atômicas de preferências e duas formas de combinação de preferências, formularam uma linguagem de consulta de dados semânticos com o conceito de preferências. As duas expressões de preferências definidas são:

- Preferências booleanas (*boolean preferences)* – das quais são especificadas como uma condição booleana onde os resultados que satisfazem a condição são preferidos sobre os resultados que não satisfazem.
- Preferências pontuadas (*scoring preferences)* – das quais são especificadas por dois operadores (*HIGHEST, LOWEST*) seguidos de uma expressão numérica. Assim, os resultados desta expressão que levam ao maior (*HIGHEST*) ou menor (*LOWEST*) valor são preferidos aos menores ou maiores.

Essas preferências atômicas podem ser combinadas em dois tipos de preferências multidimensionais:

- Preferência composta de Pareto (*pareto composed preference*)– que consiste de duas expressões de preferências conectadas por um operador (*AND*) e cada uma das expressões são avaliadas independentemente. Assim, um objeto é preferido se este é melhor em uma das preferências e pelo menos igualmente bom na outra preferência.
- Preferência em cascata (*cascading preference)* – que consiste de duas expressões de preferências conectadas por um operador (*CASCADE)* onde a primeira preferência é avaliada antes, enquanto que a segunda

preferência somente é considerada com os objetos dos quais são igualmente importantes com relação à primeira preferência.

Com isso, estenderam o ARQ (ARQ, 2007), um motor de busca para SPARQL que faz parte do *framework Jena* (Jena, 2007), com o construtor e as expressões de preferência do SPARQL *Preference.*

Vejamos um exemplo:

Busque carros. Eu gosto mais de carros pretos do que carros brancos. Se possível que não custe mais que 50.000 e que tenha os menores valores de quilômetros rodados e a maior quantidade de itens opcionais.

```
 1 PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 2 PREFIX car: <http://www.unifor.br/mia/carro.owl#>
 3 SELECT  *
 4 WHERE
 5  {    ?id rdf:type  car:Carro
 6       ?id car:temCor   ?cor
 7       ?id car:temPreco ?preco
 8       ?id car:temKM  ?km
 9       ?id car:numeroItensOpcionais ?numOpcionais }
10 PREFERRING
11       ?cor= 'PRETO'
12                CASCADE ?cor= 'BRANCO'
13 AND
14       ?preco<= '50000'
15 AND
16       LOWEST(?km)
17AND
18       HIGHEST(?numOpcionais)
```

Figura 16. Exemplo de sentença SPARQL com o operador de preferência (SPARQL *Preference)*

Na Figura 16 temos esse exemplo em uma sentença SPARQL *Preference* contendo as expressões de preferência já citadas. Na linha 10 temos o construtor modificador indicado que a partir daquela linha se tem expressões de preferência. Nas linhas 11 e 12 temos a primeira expressão de preferência (*Eu gosto mais de carros pretos do que carros brancos)* formulada por uma combinação de preferência em cascata de duas preferências booleanas (cor preta mais importante que cor branca). Na linha 14 temos uma preferência atômica booleana representando os preços menores que 50.000. Nas linhas 16 e 18 temos dois exemplos das preferências pontuadas onde são valorizadas as menores quilometragens e a maior quantidade de itens opcionais (*LOWEST* e *HIGHEST*, respectivamente). Por fim, nas

linhas 13, 15 e 17 temos o operador de preferência composta de Pareto (*AND)* combinando todas essas preferências independentes para que o princípio do ótimo de Pareto seja aplicado.

De maneira resumida, o princípio do ótimo de Pareto[22] reza que certos objetos dominam outros se eles são considerados melhores em uma das preferências e pelo menos equivalentes em todas as outras preferências (Börzsönyi, Kossmann, & Stocker, 2001), (Kießling, 2002), (Chomicki, 2003).

3.2.1.2 Semântica de SPARQL *Preference*

A semântica do modelo de preferências de SPARQL *Preference* define que preferências são expressas como relações binárias entre tuplas de uma mesma relação de banco de dados (Siberski, Pan, & Thaden, 2006). Baseando-se na formalização proposta por Chomicki (Chomicki, 2003) e na extensão da álgebra relacional, a semântica do modelo tem como conceito central a noção de dominação – definição abaixo.

"$Definição\ 1.\ Dada\ uma\ relação\ R(A_1, \dots, A_n)\ tal\ que\ U_i,$
$1 \leq i \leq n,$ é $o\ domínio\ do\ atributo\ A_i,$
$uma\ relação \succ$ é $uma\ relação\ de\ preferência\ sobre\ R,$
se é $um\ subconjunto\ de\ (U_1 \times \dots \times U_n) \times (U_1 \times \dots \times U_n).\ Uma\ tupla\ resultante\ t_1$ é $dita\ como\ dominada\ por\ t_2\ se\ t_1 \succ t_2$."

A partir dessa definição, os autores restringiram essa noção de relação mais genérica para expressões que batizaram de fórmulas de preferências intrínsecas (*intrinsic preference formulas).* Elas são expressões de lógica de primeira ordem com um conjunto limitado de operadores, conforme definição abaixo:

"$Definição\ 2.\ Dada\ uma\ relação\ R, uma\ fórmula\ de\ preferência\ intríseca\ C(t_1, t_2)$ é, $uma\ fórmula\ de\ primeira\ ordem\ sobre\ duas\ tuplas\ de\ R,$ $no\ qual\ usa\ apenas\ operadores\ de\ igualdade\ e\ ordem\ (<, >).\ Esta\ fórmula\ de\ preferência\ C\ define\ uma\ relação\ de\ preferência\ \succ_C$:

$$t_1 \succ_C t_2 \equiv C(t_1, t_2)$$"

Com intuito de uma notação mais conveniente, também definem um operador para denotar incomparabilidade entre duas tuplas como definição seguinte.

[22] Mais detalhes sobre esse princípio serão apresentados no Capítulo 4 na Seção 4.4.1

"$Definição\ 3.\ Dada\ uma\ fórmula\ de\ preferência\ C\ e\ duas\ tuplas\ t_1\ e\ t_2,$
$o\ operador\ de\ incomparabilidade\ \sim_C$ é $definido\ como$:
$t_1 \sim_C t_2 \equiv t_1 \not\succ_C t_2 \wedge t_2 \not\succ_C t_1$
$Se\ t_1\ domina\ t_2$ ou é $incomparável\ com\ ele, então\ denota\ que$:
$t_1 \gtrsim_C t_2 \equiv t_1 \succ_C t_2 \vee t_2 \sim_C t_1$."

A partir da Definição 3, eles puderam definir um novo operador, que chamam de *winnow operator*, que seleciona todos os objetos não-dominantes de um conjunto de tuplas. Abaixo encontra-se sua definição.

"$Definição\ 4. Se\ R$ é $uma\ relação\ e\ C\ uma\ fórmula\ de\ preferência,$
$que\ define\ uma\ relação\ de\ preferência\ sobre\ R,$
$winnow\ operator\ \omega_C$ é $definido\ como\ \omega_C(R), e\ para\ cada\ instância\ r\ de\ R$: $\omega_C(r) = \{t \in r \mid \neg\exists t' \in \mathrm{r}.\, t' \succ_{\mathrm{C}} t\}$

Com isso, as relações de preferência em SPARQL *Preference* se concentraram no conceito de expressões de preferências atômicas e métodos para combinação destas preferências atômicas (Siberski, Pan, & Thaden, 2006), (Abel, Herder, Kärger, Olmedilla, & Siberski, 2007).

As preferências atômicas em SPARQL *Preference* dividem-se em dois tipos:

- Preferências booleanas – especificadas por uma expressão booleana (EB) onde os resultados que satisfazem essa expressão têm preferência sobre os resultados que não a satisfazem.

  "$Para\ quaisquer\ soluções\ S_i\ e\ S_j,$
  $a\ relação\ de\ dominação\ para\ tal\ preferência, \succ_{C_{EB}}$ é $definida\ como$:

  $$S_i \succ_{C_{EB}} S_j \equiv EB(S_i) \wedge \neg EB(S_j)"$$

- Preferências de *scoring* – especificadas por uma expressão de valor onde os resultados desta expressão que tendem a valores maiores têm preferência sobre os resultados com valores menores.

  "$Para\ quaisquer\ soluções\ S_i\ e\ S_j,$
  $a\ relação\ de\ dominação \succ_{C_{Lowest,<}}\ e\ \succ_{C_{Highest,<}}$ $são\ definida\ como$:

  $S_i \succ_{C_{Lowest,<}} S_j \equiv S_i < S_j$ $\qquad$ $S_i \succ_{C_{Highest,<}} S_j \equiv S_j < S_i$

Para a combinação dessas relações de preferências, SPARQL *Preference* define duas composições multidimensionais também baseadas em (Chomicki, 2003). São elas:

- Preferências multidimensionais:

"$Para\ quaisquer\ soluções\ S_i\ e\ S_j,\quad a\ relação\ de\ dominação,$
$para\ combinar\ preferências\ independentes \succ_{[C_1 AND\ C_2]}$ é $definida\ como$:
$$S_i \succ_{[C_1 AND\ C_2]} S_j \equiv S_i \succsim_{C_1} S_j \ \wedge\ S_i \succsim_{C_2} S_j \ \wedge \left(S_i \succ_{C_1} S_j \ \vee S_i \succ_{C_2} S_j\right)$$
S_i não é $dominado\ por\ S_j\ nem\ em\ C_1 nem\ em\ C_2\ e\ S_i\ domina\ S_j\ tanto\ em\ C_1\ e\ C_2$"

- Preferências em cascata:

"$Para\ quaisquer\ soluções\ S_i\ e\ S_j, a\ relação\ de\ dominação,$
$para\ combinar\ preferências\ priorizadas \succ_{[C_1 CASCADE\ C_2]}$ é $definida\ como$:
$$S_i \succ_{[C_1 CASCADE\ C_2]} S_j \equiv S_i \succ_{C_1} S_j \ \vee \left(S_i \sim_{C_1} S_j \wedge S_i \sim_{C_2} S_j\right)$$"

3.2.2 Preferências e Ontologias

Recentemente, (Balke, Güntzer, & Lofi, 2007) seguiram uma linha similar a de nossa proposta ao permitir que usuários modelem preferências utilizando OWL para direcionar resultados de conjuntos *skylines* (Börzsönyi, Kossmann, & Stocker, 2001) através de informações adicionais fornecidas interativamente por esses usuários. Porém as relações de preferências restringem-se a: tem preferência (*isPreferredOver)* ou tem equivalência (*isEquivalentTo).*

Existem outros trabalhos nesse sentido de utilizar ontologias para expressar preferências. (Ngoc, Lee, & Lee, 2005), por exemplo, utiliza OWL para expressar preferências no domínio da computação ubíqua. (Kim, Dou, Liu, & Kwak, 2007) constrói uma ontologia para representar as preferências de usuários derivadas através de um processo de mineração de dados sobre e-mails e com isso prever a ação dos usuários na chegada de possíveis e-mails de *spam.* (Tsinaraki & Christodoulakis, 2006) permite a especificação explícita de preferências sobre conteúdos multimídias em OWL.

3.3 Problemática

3.3.1 Linguagens de consultas

SPARQL *Preference* é uma linguagem de consulta com noções de preferência para dados descritos em OWL e RDF. Contudo nós entendemos que não se trata de uma linguagem adequada para a representação de preferências na Web Semântica. Isso porque embora possua um formato de sentenças relativamente intuitivo para programadores, pois se assemelham a uma linguagem de consulta de banco de dados, SPARQL *Preference* não foi projetado para a representação explícita de preferências. Ou seja, trata-se de uma linguagem para recuperação de dados com preferências e não uma linguagem para representação declarativa de preferências.

Essa característica não-declarativa de SPARQL *Preference* não é adequada para um contexto onde agentes necessitem compreender e compartilhar o significado de preferências. Já uma representação declarativa de preferências evita a ambigüidade de interpretações, facilitando a compreensão mais correta de sua semântica.

Portanto, entendemos que é mais apropriado o uso de linguagens de ontologias como OWL para a representação de preferências, pois possuem estruturas e construtores que permitem a representação explícita da semântica dessas preferências. Isso porque o fato dessas linguagens, como OWL, terem sido construídas com esse propósito de representação de conhecimento facilita o compartilhamento, bem como a "compreensão" por parte dos agentes do que realmente significam essas preferências. Outras vantagens do uso de OWL como linguagem para representação declarativa de preferências também percebidas por (Balke, Güntzer, & Lofi, 2007) são citadas abaixo. São elas:

- Padrão aceito - OWL é o padrão atual proposto pela W3C para modelagem de ontologias em aplicações da Web Semântica. Isso permite uma integração mais simples com diversas aplicações já existentes ou em desenvolvimento.
- Extensibilidade – OWL possui construtores simples para descrição de relações entre instâncias e classes. Isso permite o uso destes construtores

para definição de novos conceitos e, mais especificamente, a definição de novos tipos de preferências.

- Flexibilidade – Reuso através de diferentes domínios.
- Validação – Por ser uma linguagem estruturada, OWL permite uma maior consistência em relação a outras linguagens não-estruturadas. Além disso, OWL também foi projetado para o uso junto com raciocionadores que podem inferir novas assertivas. Vejamos um exemplo: suponha uma ontologia de domínio que descreva um estilo musical de cantores como na Figura 17. Se uma pessoa define que gosta de *blues* e *Eric Clapton* é uma instância de *Blues Britânico*, um raciocionador pode inferir que, como *Blues Britânico* é uma subclasse de *Blues*, então essa pessoa também gosta de *Eric Clapton*.

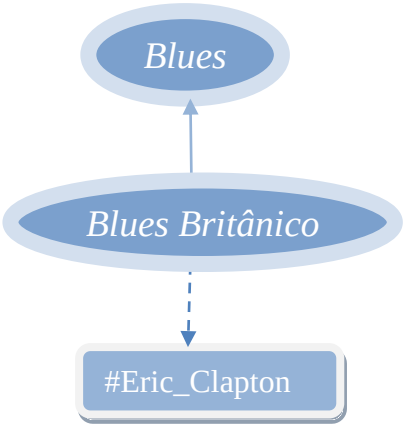


Figura 17. Exemplo de ontologia representando um estilo musical.

Outra linguagem de consulta de preferências como Preference-SQL (Kießling & Köstler, 2002) possui, além dos problemas citados de SPARQL *Preference*, o fato de só aplicar-se ao contexto de banco de dados relacional. O mesmo acontece com Preference XPath (Kießling, Hafenrichter, Fischer, & Holland, 2001), pois restringe-se a dados XML.

Portanto, acreditamos que linguagens de consultas não foram projetadas para a representação declarativa de preferências, mas somente para a recuperação de informação. E que o uso de OWL como forma de representar explicitamente variados tipos de preferências torna-se mais sensato visto que possui estruturas apropriadas para a explicitação deste tipo de conhecimento.

Além disso, uma linguagem que represente declarativamente preferências através de ontologias permite que ela torne-se uma espécie de “metalinguagem” para representação de preferências onde novos conceitos podem ser criados ou redefinidos a partir dos já existentes.

3.3.2 Preferências e Ontologias

Apesar de alguns trabalhos estarem relacionados com dados OWL e ontologias, a maioria deles está relacionada a domínios específicos (multimídia, *spam*, privacidade, serviços Web, etc.). Ou seja, eles definem um modelo de representação específico para determinado domínio que, muitas vezes, não é possível de ser utilizado em outro tipo de aplicação voltada para outro contexto. Por exemplo, um modelo de preferências definido para o contexto de conteúdos multimídias não pode ser reutilizado em um contexto diferente (por exemplo, pacotes de viagens).

O trabalho descrito em (Balke, Güntzer, & Lofi, 2007) não está relacionado a um domínio específico. Acontece que os tipos de relações de preferências definidos pelos autores (tem preferência a ou tem equivalência à) no nosso entendimento são bastante simples. Essa simplicidade impede a modelagem de preferências mais complexas como nos exemplos citados dos construtores de (Chomicki, 2003) e (Kießling, 2002). Com o uso dos construtores definidos por esses dois trabalhos, é possível a combinação de preferências para permitir, por exemplo, a representação de preferências hierarquizadas. Além disso, a forma de modelagem de preferências proposta pelos autores exige alterações na ontologia de domínio que podem tornar inviáveis em ontologias muito extensas, além do fato do reuso ser prejudicado. Isso ocorre porque é necessário definir que a classe do domínio cuja preferência está sendo modelada é subclasse de um conceito (*Predicate)* proposto pelos autores. Ou seja, isso ocasiona modificações na ontologia de domínio, o que prejudica sua reutilização em outras aplicações que por ventura já a utilizam.

Portanto, nossa proposta de representação declarativa de preferência segue esses direcionamentos, onde utilizando uma ontologia desenvolvida em OWL pretendemos representar explicitamente preferências sobre dados descritos em RDF/OWL e ao mesmo tempo permitir a combinação destas preferências e até mesmo a criação de novos conceitos

com intuito de ampliar a expressividade e permitir o reuso e compartilhamento de conhecimento, bem como o suporte a inferências.

4 ONTOLOGIAS DE PREFERÊNCIAS: OWLPREF

Conforme apresentado no Capítulo anterior, alguns trabalhos, apesar de proporem uma formalização para preferências como (Chomicki, 2003), (Kießling, 2002), (Kießling, Hafenrichter, Fischer, & Holland, 2001), não são apropriados para a representação de preferências na Web Semântica. Eles não utilizam linguagens de ontologias como OWL e, desta forma, não aproveitam seu poder de representação e inferência para o uso de agentes e aplicações no contexto da Web Semântica. Nosso trabalho segue essa direção, com a proposta de um modelo que chamamos de OWLPREF. Ele utiliza-se da praticidade de consultas SPARQL *Preference* com a expressividade e a semântica de OWL para a representação de preferências, visto que OWL se tornou padrão para a representação de conhecimento na Web e possui potência representacional suficiente para a definição de preferências. Além disso, OWL permite descrever conceitos de domínios e a relação entre os mesmos através de um conjunto de operadores (interseção, união, complemento, propriedades transitivas e inversas etc.). Com isso, OWLPREF pode assim utilizar esse conjunto de operadores para combinação de preferências e até mesmo a definição de novos conceitos de preferências.

4.1 Modelagem de OWLPREF

Modelar OWLPREF em OWL nos leva a alguns problemas citados em (Horrocks, Patel-Schneider, & Harmelen, 2003). Esses problemas estão relacionados principalmente porque na modelagem de ontologias em OWL, geralmente se faz uso da sintaxe RDF/XML (sintaxe oficial).

Descrever OWLPREF com essa sintaxe seria pouco claro, como já atestou (Horrocks, Patel-Schneider, & Harmelen, 2003), pois consideram a sintaxe RDF/XML muito prolixa. Isso é facilmente comprovado fazendo um comparativo de uma representação nesta sintaxe e em uma sintaxe de lógica descritiva. Por exemplo, em lógica descritiva temos

$MAE = FEMEA \sqcap \geq 1 temFilhos$. O mesmo exemplo em OWL com RDF/XML sintaxe teríamos algo assim:

```
<owl:Class rdf:ID= “MAE”>
   <owl:intersectionOf rdf:parsetype= “Collection”>
         <owl:Class rdfs:about= “#FEMEA”>
         <owl:Restriction>
               <owl:onProperty rdf:resource = “temFilhos” />
               <owl:minCardinality rdfs:datatype= “&xsd;Integer”>
                     1
               </owl:minCardinality>
         </owl:Restriction>
   </owl:intersectionOf>
</owl:Class>
```

Ou seja, mesmo utilizando uma sintaxe de uma linguagem formal, a legibilidade não é tão boa quanto seria se utilizássemos a sintaxe de lógica descritiva. Outro problema decorre de RDF trabalhar sempre em triplas. Com isso, algo simples em lógica descritiva como $\exists temFilhos. Person$ ficaria em RDF/XML assim:

```
_:x owl:onProperty temFilhos .
_:x someValuesFrom Person .
```

Baseado nesses problemas e em outros também elencados por (Horrocks, Patel-Schneider, & Harmelen, 2003), decidimos formalizar OWLPREF em lógica descritiva com fins didáticos, porém a ontologia propriamente dita foi implementada na sintaxe RDF/XML.

No processo da modelagem de OWLPREF, baseamos-nos em alguns princípios elencados por (Noy & McGuinness, 2001) para a construção de ontologia, visto que os consideramos suficiente para os nossos objetivos. São eles:

- “Não existe uma maneira correta de modelar um domínio – existem sempre alternativas viáveis. A melhor solução quase sempre depende da aplicação que se tem em mente e as extensões que são vislumbradas”.
- “O desenvolvimento de ontologias é necessariamente um processo iterativo”.
- “Conceitos em uma ontologia devem ser próximos aos objetos (físicos ou lógicos) e relações dentro do domínio de interesse. Eles provavelmente

tendem a ser substantivos (objetos) ou verbos (relações) em sentenças que descrevem o domínio."

Basicamente, o processo proposto por (Noy & McGuinness, 2001) resume-se a revisão e refinamento progressivo dos detalhes da ontologia. Com isso, modelamos os conceitos das preferências de modo simples e em um processo iterativo foi-se refinando os detalhes até chegarmos a OWLPREF.

Antes de apresentarmos os conceitos, faz-se necessária uma pequena introdução sobre os conceitos de lógica descritiva.

4.1.1 Lógica Descritiva

A lógica descritiva trata-se de um tipo de formalismo para representação de conhecimento baseado em conceitos/classes (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2002). Caracterizam-se pelo uso de diversos construtores para elaborar classes mais complexas a partir das mais simples. A lógica descritiva tem forte influência em OWL, principalmente na formalização da semântica, na escolha dos construtores de linguagem e na integração dos tipos de dados com os valores dos mesmos (Horrocks, Patel-Schneider, & Harmelen, 2003).

Com isso, lógica descritiva consiste de um alfabeto de nomes de conceitos distintos - *CN*, nomes de papéis (*roles*) – R*N* e nomes de indivíduos – *IN*, combinados com um conjunto de construtores para construção de conceitos e expressões de papéis (Pothipruk & Governatori, 2005). Ou seja, temos em lógica descritiva três conceitos básicos:

- Conceitos atômicos – denotam a um conjunto ou classe de objetos. Por exemplo, *Pessoa*, *Homem*, *Mulher*, etc.
- Papéis atômicos – denotam relações entre objetos. Por exemplo, *temFilho*, *éUm*, *temValor*, etc.
- Indivíduos – denotam objetos dentro do domínio: João, Maria,etc.

E como construtores, um conjunto de operadores:

- Conjunção ($\sqcap$) – $Pessoa \sqcap Homem$
- Disjunção ($\sqcup$) – $Homem \sqcup Mulher$

- Negação ($\neg$) - $\neg$Mulher
- Existencial ($\exists$) - $\exists temFilhos.Homem$
- Universal ($\forall$) - $\forall temFilhos.Mulher$
- Axioma ($\sqsubseteq$) - $Pai \sqsubseteq Pessoa$

Formalmente, uma base de conhecimento de lógica descritiva é um par $K=(\mathcal{T},A)$, onde $\mathcal{T}$ é um *Tbox* e A é um *Abox*. De forma geral, *Tbox* contem sentenças descrevendo conceitos hierárquicos (relações entre conceitos), enquanto que *Abox* contem sentenças que expressam o que, numa hierarquia, certos indivíduos fazem – relações entre indivíduos e conceitos. Vejamos um exemplo:

(1) Toda Mulher é uma Pessoa
(2) Maria é uma Mulher

No exemplo citado, a sentença (1) pertence ao *Tbox*, enquanto que a sentença (2) ao *Abox*.

Com isso, podemos dizer que *Tbox* contem um conjunto finito de axiomas de assertivas, nos quais estão na forma:

$$C \sqsubseteq D \mid C \doteq D,$$

onde C e D são expressões de conceito.

Expressões de conceito estão na forma:

$$A \mid \mathcal{T} \mid \bot \mid C \mid C \sqcap D \mid C \sqcup D \mid \exists \mathrm{R}.C \mid \forall \mathrm{R}.C,$$

onde A é um conceito atômico ou um nome de conceito em CN, R é um nome de papel em RN, $\mathcal{T}$ (topo ou todo domínio) é o conceito mais geral, e $\bot$ (base ou conjunto vazio) é o mínimo conceito geral.

Segundo (Horrocks, Patel-Schneider, & Harmelen, 2003), a técnica padrão para especificação de significado de uma lógica descritiva é através de modelos (*model theoretic semantics*) os quais se propõem a explicar a relação entre a sintaxe e o modelo intencional do domínio. Este modelo consiste de um *domínio* ($\Delta^{\mathcal{I}}$) e uma *função de interpretação* ($\cdot^{\mathcal{I}}$), onde o

domínio é um conjunto de objetos e a função de interpretação é um mapeamento de nomes de indivíduos para elementos do domínio, de classes para subconjuntos do domínio e propriedades (papéis) para relações binárias dentro do domínio. Portanto, para um indivíduo *Maria*, $Maria^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, para a classe *Pessoa*, $Pessoa^{\mathcal{I}} \sqsubseteq \Delta^{\mathcal{I}}$, e para a propriedade ser, $ser^{\mathcal{I}} \sqsubseteq \Delta^{\mathcal{I}} x \Delta^{\mathcal{I}}$. Propriedades de tipos de dados (*datatypes)* como inteiros (integer) são interpretadas como um subconjunto de $\Delta^{\mathcal{I}}_{\mathcal{D}}$, que representa uma interpretação adicional de domínio para valores de dados, e valores como o inteiro "35" são interpretados como elementos de $\Delta^{\mathcal{I}}_{\mathcal{D}}$. Elas também são conhecidas como propriedades concretas. Outro tipo de propriedade são as consideradas propriedades abstratas. Elas são interpretadas como relações binárias em $\Delta^{\mathcal{I}}$ (por exemplo, subconjuntos de $\Delta^{\mathcal{I}} x \Delta^{\mathcal{I}}$), enquanto que as propriedades de tipos de dados são interpretadas como relações binárias ente $\Delta^{\mathcal{I}} x \Delta^{\mathcal{I}}_{\mathcal{D}}$. Na formalização de OWLPREF batizamos propriedades abstratas como *R* e propriedades concretas (ou tipo de dados) como *U* – ver Tabela 1.

Tabela 1. Sintaxe e Semântica de propriedades

Nome	**Sintaxe**	***Semântica***
Propriedades Abstratas	*R*	$R^{\mathcal{I}} \sqsubseteq \Delta^{\mathcal{I}} x \Delta^{\mathcal{I}}$
Propriedades Concretas	*U*	$U^{\mathcal{I}} \sqsubseteq \Delta^{\mathcal{I}} x \Delta^{\mathcal{I}}_{\mathcal{D}}$

Do ponto de vista de modelagem e semântica, OWL possui uma forte correspondência com a lógica descritiva em muitos construtores lógicos como os apresentados na Tabela 2 extraída de (Kalyanpur, 2006). Em termos de OWL, podemos dizer que um conceito(*C)* em lógica descritiva equivale a uma classe (*owl:Class),* um papel (*R)* é equivalente a uma propriedade de objetos (*owl:ObjectProperty),* um papel concreto a uma propriedade de tipo de dados (*owl:DatatypeProperty),* e um indivíduo ou objeto a um individuo OWL (*owl:Individual).* Perceba que na Tabela 2, *C* e *D* referem-se a classes OWL, *P* refere-se a propriedades OWL, I_1 e I_2 referem-se a indivíduos OWL e *n* são inteiros não negativos.

Tabela 2. Correspondência entre OWL e Lógica Descritiva

Construtor OWL	Representação em Lógica Descritiva	*Exemplo*
owl:equivalentTo(C,D)	$C \equiv D\ (C \sqsubseteq D\ e\ D \sqsubseteq D)$	$Humano \equiv Pessoa$
rdfs:subClassOf(C,D)	$C \sqsubseteq D$	$Pais \sqsubseteq Pessoa$
owl:complementOf(C,D)	$C \equiv \neg D$ *(negação)*	$Homem \equiv \neg$Mulher
owl:disjointWith(C,D)	$C \sqsubseteq \neg D$	$Pai \sqsubseteq \neg Mae$
owl:intersectionOf(C,D)	$C \sqcap D$ *(conjunção)*	$Parente \sqcap Homem$
owl:unionOf(C,D)	$C \sqcup D$ *(disjunção)*	$Pai \sqcup Mae$
owl:oneOf (I_1,I_2)	$\{I_1\} \sqcup \{I_2\}$	$\{Joao\} \sqcup \{Maria\}$
owl:someValuesFrom (P,C)	$\exists P.C$ *(existencial)*	$\exists temCrianca.Filha$
owl:allValuesFrom(P,C)	$\forall P.C$*(universal)*	$\forall temCrianca.Filho$
owl:hasValue (P,I_1)	$\exists P.\{I_1\}$	$\exists temCrianca.\{Maria\}$
owl:cardinality(P,n)	$= n.P$	$= 2.temPais$
owl:minCardinality(P,n)	$\geq n.P$	$\geq 1.temFilha$
owl:maxCardinality(P,n)	$\leq n.P$	$\leq 2.temCrianca$

Fonte: (Kalyanpur, 2006)

4.2 Formalizando Preferências em OWLPREF

Quando alguém revela que possui preferência por algo, por exemplo, "Prefiro bolas vermelhas", na verdade está dizendo que prefere bolas vermelhas em relação a alguma outra coisa que geralmente encontra-se no mesmo contexto que está inserido. Isto é, se João prefere bolas vermelhas e o contexto em que se encontra é C_1*={bolas vermelhas, bolas azuis e bolas pretas}* então podemos concluir que para João as bolas vermelhas têm mais importância que as bolas azuis e as bolas pretas. Entendemos assim que existe uma relação de ordem entre o conjunto das bolas vermelhas e o complemento deste (no caso do exemplo: bolas que não sejam vermelhas). Esse complemento sempre vai depender do contexto em que esteja inserido. Por exemplo, se o contexto fosse o universo de todas as bolas então para esse caso, as bolas vermelhas teriam mais importância que todas as outras bolas que tivessem cor diferente de vermelhas. Em (Chomicki, 2003), é definido uma relação binária de preferência entre tuplas de uma mesma relação de um banco de dados. Adaptando essa definição e a definição de (Kießling, 2002) para nossos conceitos, definimos a relação de preferência como uma relação de ordem parcial estrita entre elementos de um conjunto.

Definição 1. Sejam x e y elementos de um conjunto qualquer, dizemos que $x \succ_o y\ se\ x\ tem\ preferência\ a\ y$. Propriedades típicas dessa relação binária de ordem incluem:

[transitividade] $x \succ_o y \wedge y \succ_o z \Rightarrow x \succ_o z$

[assimetria] $x \succ_o y \Rightarrow y \prec_o x$

[irreflexibilidade] $\neg(x \succ_o x)$

Transitividade no sentido de que se *x* tem preferência a *y* e *y* tem preferência a *z* implica que *x* tem preferência a *z.* Por exemplo: "se prefiro leite integral ao leite desnatado e prefiro leite desnatado ao semi-desnatado, então posso concluir que prefiro leite integral ao leite semi-desnatado. Com relação à assimetria, dizemos que se *x* tem preferência a *y* então *y* tem menos preferência que *x.* Com o mesmo exemplo anterior, podemos inferir que: "se prefiro leito integral ao leite desnatado, então o leite desnatado é menos importante que o leite integral." Por último, temos a propriedade de irreflexibilidade na qual um elemento não pode ter preferência sobre ele mesmo. Por exemplo: "não posso preferir leite integral a ele mesmo."

Dizemos também que se *x* não têm preferência sobre *y,* então $x \nsucc_o y$. Pelo princípio da assimetria implica que se $x \nsucc_o y \Rightarrow y \nsucc_o x$, concluindo-se que nesse caso, *x* e *y* têm a mesma importância.

Com a definição de relação de ordem de importância juntamente a essas propriedades, podemos conceituar preferências em OWLPREF como uma relação de ordem parcial estrita entre conjuntos de elementos de OWL, como segue abaixo:

Definição 2. Sejam *C* e *D* duas classes de OWL, tal que $C, D \sqsubseteq owl{:}Thing$, diz-se que *C* tem preferência sobre *D* quando quaisquer indivíduos de *C* têm preferência sobre quaisquer indivíduos de *D.*

$$C\ Pref\ D\ iif\ (\forall x\ \in C) \succ_o (\forall y\ \in D)$$

A fórmula acima pode ser lida como: "*C tem preferência a D".* Em muitos casos omitimos o conjunto *D* quando o mesmo representa o complemento do conjunto C como segue na definição abaixo.

Definição 3. Uma preferência por uma classe C em OWLPREF equivale a preferência de C em relação ao complemento de sua classe.

$$Pref\ C \equiv C\ Pref\ D \mid D = \neg \mathrm{C}\text{, onde } C, D \sqsubseteq owl{:}Thing\ e\ D = owl{:}Thing \backslash C$$

Com isso se voltarmos ao exemplo citado no início desta seção: *João prefere Bolas Vermelhas.* Podemos formalizá-lo segundo nossa conceituação do seguinte modo com as seguintes premissas:

I – $bolas\ vermelhas, bolas\ azuis, bolas\ pretas\ \in Bola$

II -$Bola\ \sqsubseteq owl{:}Thing$
III- $Contexto\ Cx = Bola$

$Pref\ bolas\ vermelhas\ no\ contexto\ Cx \equiv (\ bolas\ vermelhas \succ_o\ bolas\ azuis) \wedge$
$bolas\ vermelhas\ \succ_o\ bolas\ pretas)$

4.3 Conceitos de OWLPREF

A partir da definição do conceito de preferência na seção anterior, podemos agora falar sobre os conceitos definidos em OWLPREF. Todos eles foram formalizados em lógica descritiva baseados nas definições já anteriormente relatadas e nos conceitos definidos em (Horrocks & Patel-Schneider, 2003), (Horrocks, Patel-Schneider, & Harmelen, 2003), (Kießling, 2002). Na Figura 18 temos uma representação gráfica dos conceitos da ontologia com intuito de facilitar a visualização e compreensão de sua hierarquia (classes e subclasses) e o relacionamento entre os mesmos. Os conceitos de *Preferência Simples* foram modelados de forma que possam ser utilizados como construtores de novos conceitos de preferências. Isso é possível através da própria linguagem OWL e pela combinação de preferências com as *Preferências Compostas* (ver Seção 4.4.2 deste capítulo sobre a construção de novos conceitos).

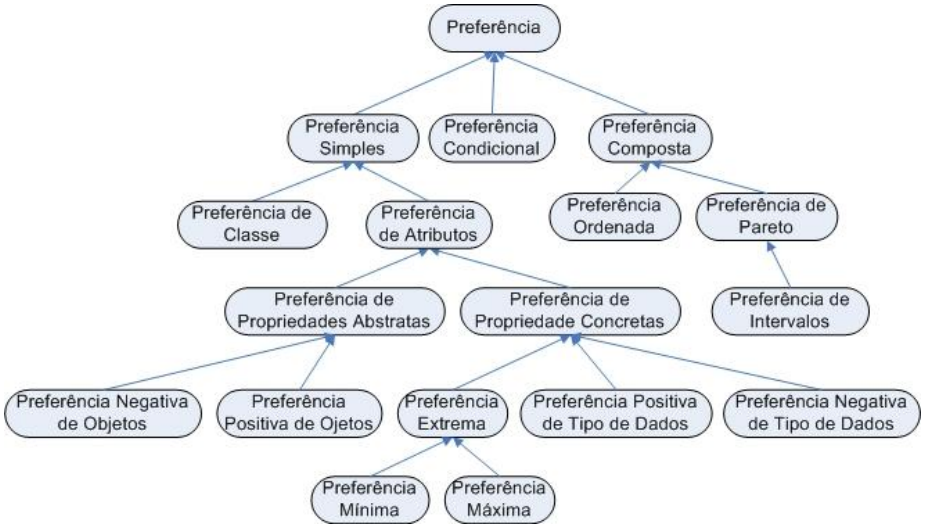


Figura 18. Hierarquia de conceitos de OWLPREF

O conceito básico é o de *Preferência (Preference)* o qual todos os outros conceitos são especializações do mesmo. Uma preferência em OWLPREF por ser dividida em *Preferências Simples*, *Preferências Compostas* e *Preferências Condicionais.*

As *Preferências Simples* referem-se ao conjunto de preferências relacionadas a classes e a atributos, sendo que atributos podem ser relacionados a tipo de dados ou a objetos. Criamos esses subtipos de preferências levando em consideração as diferentes formas de modelagem que uma ontologia pode ter.

Vejamos um exemplo. Suponha uma ontologia que represente o domínio das cores seja modelada em termos de classes, onde cada cor é uma subclasse da classe *Cor* e as instâncias de cada classe representam cores mais específicas como no exemplo da Figura 19. Nela temos uma classe *Cor* com três subclasses representando tipos de cores: *Branco*, *Azul* e *Verde*. As instâncias de cada uma dessas subclasses representam variedades de cores diferentes (branco_gelo, azul_escuro, verde_claro). Representar preferências sobre uma ontologia modelada dessa forma necessita de referências às classes propriamente ditas: *Prefiro a classe Azul à classe Verde no contexto das cores (subclasses de Cor)*

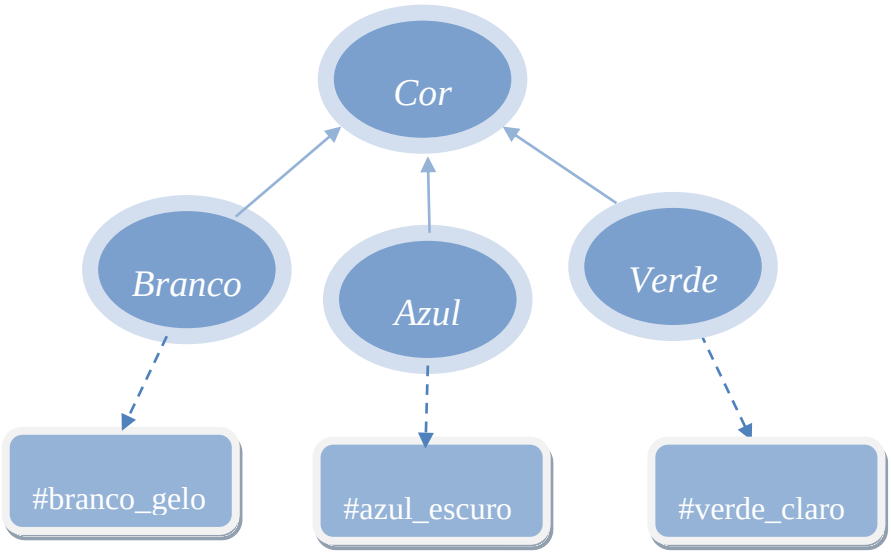


Figura 19. Exemplo de ontologia de cores baseada em classes

Porém essa mesma ontologia de cores poderia ter sido modelada de outra forma. Isto é, ao invés dos tipos de cores estarem modelados como classes, eles poderiam ter sido modelados como atributos de classes. Por exemplo, suponha um atributo *temCor* que possui como valores possíveis instâncias da classe *Cor* da Figura 19. Neste caso, ao invés de utilizar preferências sobre classes, utilizamos preferências sobre atributos, por exemplo: *Prefiro temCor(#azul_escuro) à temCor(#verde_claro)*. O mesmo atributo *temCor* poderia ter sido modelado com uma propriedade de tipo de dados, sendo também adequado nesses casos o uso de *Preferências de Atributos*. Exemplo: *Prefiro temCor("azul"),* sendo *"azul"* uma *string.*

Portanto, levando em consideração essas duas possibilidades de modelagens de ontologias, definimos *Preferências Simples* como as preferências de OWLPREF relacionadas a classes (*Preferências de Classes)* e às preferências sobre propriedades, sejam elas relacionadas a objetos (propriedades abstratas) ou a tipo de dados (propriedade concretas). Apesar de não existir instâncias diretas de *Preferências Simples*, modelamos esse e alguns outros conceitos com o objetivo de agrupar preferências que possuem características comuns entre si e com isso obter uma melhor organização da ontologia.

Preferências de Classe (*ClassPreference)* são as preferências sobre tipos específicos de conceitos/classes, aonde todos os indivíduos/instâncias desses conceitos têm preferência em relação a todos os outros. Por exemplo: *prefira Notebook a Desktop no contexto de Computadores* significa que todas as instâncias da classe *Notebook* têm preferência às instâncias da classe *Desktop* e qualquer outra instância de classe que seja

diferente de *Notebook* e seja subclasse da classe dos *Computadores.* Com isso temos a seguinte definição:

Definição 4. Uma *Preferência de Classe* em OWLPREF é uma *Preferência Simples* em que todos os indivíduos de uma determinada classe, expressado na relação *hasOWLClass*, têm preferência sobre todos os outros que não fazem parte desta classe. O contexto em que a preferência se aplica é representado pela relação *hasContext.*

Sintaxe

$$ClassPreference \sqsubseteq SimplePreference \sqcap (\leq 1 \forall hasOWLClass.C) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$$Seja\ C_2\ uma\ classe\ que\ representa\ o\ contexto\ tal\ que\ C_2 \subseteq owl{:}Thing,$$
$$C_1\ uma\ classe\ tal\ que\ C_1 \subseteq C_2, ClassPreference \text{ é } definido\ como$$

$$x \succ_o y\ iff\ x \in C_1 \wedge y \in C_2 \backslash C_1$$

Exemplo

$$ClassPreference\ com\ hasOWLClass.Carro\ e\ hasContext.Veículos\ implica\ que$$

$$Carro \succ_o Y\ iff\ Carro \subseteq Veículo \wedge Y \in Veículo \backslash Carro$$

O exemplo acima representa que todas as instâncias da classe *Carro* têm preferências sobre quaisquer outras instâncias que sejam subclasses da classe *Veículo* uma vez que a preferência refere-se ao contexto de veículos. Por exemplo, instâncias de motos, instância de barcos, etc.

Relacionando com o exemplo da Figura 19, poderíamos definir uma *Preferência de Classe* sobre a classe *Azul* no contexto das cores (classe *Cor*), o que significaria que todas as instâncias da classe *Azul* (no caso cores azuis) têm preferências sobre as instâncias de outras classes (nas quais representam outras cores). Na Figura 20, temos esse exemplo na sintaxe RDF/XML em OWLPREF.

```
<ClassPreference rdf:ID= “PreferenciaCorAzul”>
        <hasOWLClass rdf:resource= “#AZUL” />
        <hasContext rdf:resource= “#Cor” />
</ClassPreference>
```

Figura 20. Exemplo de Preferência de Classe em OWLPREF

Já as preferências relacionadas a atributos são as preferências por todos os indivíduos que possuem determinado atributo ou propriedade. Elas podem está relacionadas a propriedades abstratas ou propriedades concretas. Conforme já foi falado anteriormente, definimos esses dois tipos de propriedades com intuito de permitir que OWLPREF fique compatível com os dois tipos de propriedades existentes em OWL (*owl:ObjectProperty* e *owl:DatatypeProperty*, respectivamente. Assim como as *Preferências Simples*, definimos o conceito de preferências de atributos (*PropertyPreference)* para agrupar as preferências relacionadas a essas propriedades.

Preferências por propriedades abstratas (*ObjectPropertyPreference)* são as preferências relacionadas a valores específicos de propriedades as quais denotam relações entre objetos. Isto é, preferências por instâncias que possuem objetos (outras instâncias) como valores de atributos específicos. Por exemplo, supondo uma propriedade *temCor* que possui como possíveis valores instâncias das classes exemplificadas na Figura 19. Nesse caso, uma preferência por propriedades abstratas seria a forma mais adequada de modelagem de uma preferência to tipo: *temCor(#verde_claro).*

As preferências por propriedades abstratas dividem-se *Preferências Positivas* (*PositiveObjectPreference)* ou *Preferências Negativas* (*NegativeObjectPreference)* com o intuito de representar valores desejáveis e não desejáveis sobre objetos.

No caso da *Preferência Positiva por Propriedades de Objetos* (*PositiveObjectPreference),* é especificado qual valor do atributo é preferido em relação ao outros. Ou seja, o objeto em questão tem maior preferência. Por exemplo: *prefira pessoas que conheçam o João,* temos uma *Preferência Positiva* pelas instâncias da classe *Pessoa* que possuam a propriedade *conhece* cujo valor é o objeto *João.* O oposto acontece na *Preferência Negativa*, onde é especificado qual objeto não é desejado. Para o mesmo contexto, estaríamos falando de representar a preferência por pessoas que **não** conheçam João.

Definição 5. Uma *Preferência Positiva por Propriedades de Objetos* é uma *Preferência de Propriedades de Objetos* (pois se relaciona a propriedades abstratas) que expressa uma preferência sobre os indivíduos cuja propriedade, especificada na relação *hasOWLProperty*, possui o valor especificado na relação *hasObjectPropertyValue* sobre os indivíduos que não possuem este valor.

Sintaxe

$$PositiveObjectPreference$$
$$\sqsubseteq ObjectPropertyPreference \sqcap (\leq 1\ \forall hasOWLProperty.R)$$
$$\sqcap (\leq 1 \forall hasObjectPropertyValue.\text{o}) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$$Seja\ C_1\ uma\ classe\ que\ representa\ o\ contexto\ tal\ que\ C_1 \subseteq owl{:}Thing,$$
$$C_2\ uma\ classe\ tal\ que\ C_2 \subseteq C_1,$$
$$P\ uma\ classe\ que\ representa\ uma\ propriedade\ abstrata\ onde\ P \subseteq R,$$
$$o_1\ um\ valor\ de\ P\ onde\ o_1 \in owl{:}Thing,$$
$$PositiveObjectPreference\ \text{é}\ definida\ como\ sendo$$
$$x \succ_o y\ iff\ x \in C_2 \wedge y \in C_1 \backslash C_2, onde\ C_2 = \{\ o\ |(o, o_1) \in P\ \}$$

Exemplo

$$PositiveObjectPreference\ \ com\ hasOWLProperty.temCor,$$
$$hasObjectPropertyValue.Azul\ e\ hasContext.Veículo, implica\ que$$
$$x \succ_o y\ iff\ x \in C \wedge\ y \in Veículo \backslash C, onde\ C = \{\ o\ |(o, azul) \in temCor\}\ e\ C \subseteq \text{Veículo}$$

O exemplo acima representa que todas as instâncias que possuem a propriedade *temCor* com valor *azul* tem preferência sobre outras instâncias desde que todas essas instâncias em questão sejam instâncias de subclasses da classe *Veículo* (visto que a preferência refere-se ao contexto de veículos). Por exemplo, instâncias de motos azuis, barcos azuis ou carros azuis, têm preferência sobre motos pretas, barcos amarelos ou carros vermelhos.

Definição 6. Uma *Preferência Negativa por Propriedades de Objetos* é uma *Preferência de Propriedades de Objetos* que expressa uma preferência sobre os indivíduos cuja propriedade, especificada na relação *hasOWLProperty*, possui valor que difere do especificado na relação *hasObjectPropertyValue*.

Sintaxe

$$NegativeObjectPreference$$

$$\sqsubseteq ObjectPropertyPreference \sqcap (\leq 1\, \forall hasOWLProperty.R)$$

$$\sqcap (\leq 1 \forall hasObjectPropertyValue.o) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$$Seja\ C_1\ uma\ classe\ que\ representa\ o\ contexto\ tal\ que\ C_1 \subseteq owl{:}Thing,$$
$$C_2\ uma\ classe\ tal\ que\ C_2 \subseteq C_1,$$
$$P\ uma\ classe\ que\ representa\ uma\ propriedade\ abstrata\ onde\ P \subseteq R,$$
$$o_1\ um\ valor\ de\ P\ onde\ o_1 \in owl{:}Thing,$$
$$NegativeObjectPreference\ \text{é}\ definida\ como\ sendo$$

$$x \prec_o y\ iff\ x \in C_2 \wedge y \in C_1 \backslash C_2, onde\ C_2 = \{\, o \mid (o, o_1) \in P \,\}$$

Exemplo

$$NegativeObjectPreference\ com\ hasOWLProperty.temCor,$$
$$hasObjectPropertyValue.Azul\ e\ hasContext.Veículo, implica\ que$$

$$x \prec_o y\ iff\ x \in C \wedge y \in Veículo \backslash C, onde\ C = \{\, o \mid (o, azul) \in temCor \,\}\ e\ C \subseteq \text{Veículo}$$

Similar ao exemplo anterior, este exemplo também representa preferência sobre cores de tipos de veículos. Acontece que nas preferências negativas, as instâncias que terão preferências são aquelas que possuem a propriedade *temCor* com valor diferente de *azul*. Neste caso, instâncias de motos pretas, barcos amarelos ou carros vermelhos, por exemplo, têm preferência sobre motos azuis, barcos azuis ou carros azuis.

Na Figura 22, temos esses dois exemplos citados de *Preferências de Propriedade de Objetos* em RDF/XML. No primeiro quadro (parte superior) temos um exemplo de *Preferência Positiva por Propriedade de Objetos*, representando a preferência por cores azuis e no segundo quadro (parte inferior) temos um exemplo de *Preferência Negativa por Propriedade de Objetos* representando a preferência por cores que não sejam amarelas. Ambas no contexto de veículos.

```
<PositiveObjectPreference rdf:ID= “PreferenciaCorAzul”>
        <hasOWLProperty rdf:resource= “#temCor />
        <hasObjectPropertyValue rdf:resource= “#azul” />
        <hasContext rdf:resource= “#Veiculo” />
</PositiveObjectPreference>
```

```
<NegativeObjectPreference rdf:ID= “PreferenciaCorNaoAzul”>
        <hasOWLProperty rdf:resource= “#temCor />
        <hasObjectPropertyValue rdf:resource= “#amarelo” />
        <hasContext rdf:resource= “#Veiculo” />
</NegativeObjectPreference>
```

Figura 21. Exemplo de *Preferência por Propriedade de Objetos*: *Positiva* representando preferências por cor azul e *Negativa* representando preferências por cores diferentes de amarelo

Similar às *Preferências de Propriedade de Objeto*s temos as *Preferências de Propriedade de Tipos de Dados* (*DatatypePreference)* que representam as preferências por instâncias que possuem propriedades concretas cujo valor relacionam a tipos de dados primitivos (como inteiro, *float*, string, etc.).

Assim como as *Preferências de Propriedade de Objetos*, também definimos subtipos de preferências para as *Preferências de Tipo de Dados*: além das preferências positivas e negativas definimos também as preferências máximas e mínimas.

As preferências positivas e negativas relacionadas a propriedades de tipos de dados (*PositiveDatatypePreference*, *NegativeDatatypePreference* respectivamente) são bastante similares às preferências positivas/negativas sobre propriedades de objetos. A principal diferença é o próprio fato de elas estarem relacionadas às propriedades concretas ou de tipos de dados (em OWL, *owl:DatatypeProperty)* ao invés de propriedades abstratas. Outra diferença está no fato serem propriedades concretas, o que permite a utilização de alguns tipos de operadores para expressar tipos de preferências mais genéricas do tipo: *prefira carros com ano maior que 2005; prefira cds com preço menor ou igual a 50,00; prefira livros com autor igual a “Dan Brown”,* entre outras. Restringimo-nos apenas aos operadores: *Maior que*, *Maior ou Igual que*, *Igual que*, *Menor que* e *Menor ou Igual que*. Seguem suas definições:

Definição 7. Uma *Preferência Positiva de Tipo de Dados* é uma *Preferência de Propriedade de Tipo de Dados* (pois se relaciona a propriedades concretas) que expressa uma preferência sobre os indivíduos cuja propriedade, especificada na relação *hasOWLProperty*, possui valor

que se enquadra dentro da combinação entre o valor especificado na relação *hasDatatypePropertyValue* e o operador especificado na relação *hasOperator*.

Sintaxe

$$PositiveDatatypePreference \sqsubseteq DatatypePropertyPreference \sqcap (\leq 1\ \forall hasOWLProperty.U) \sqcap (1 \leq \forall hasDatatypePropertyValue.v) \sqcap ((\exists hasOperator.<) \sqcup (\exists hasOperator.\leq) \sqcup (\exists hasOperator.=) \sqcup (\exists hasOperator.>) \sqcup (\exists hasOperator.\geq)) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$$Seja\ C_1\ uma\ classe\ que\ representa\ o\ contexto\ tal\ que\ C_1 \subseteq owl{:}Thing,$$
$$C_2\ uma\ classe\ tal\ que\ C_2 \subseteq C_1,$$
$$P\ uma\ classe\ que\ representa\ uma\ propriedade\ concreta\ onde\ P \subseteq U,$$
$$D\ uma\ classe\ que\ representa\ o\ domínio\ de\ valores\ de\ tipo\ de\ dados,$$
$$v_1\ um\ literal\ que\ representa\ um\ valor\ da\ propriedade\ P\ onde\ v_1 \in D,$$
$$PositiveDatatypePreference\ é\ definida\ como\ sendo$$

$$x \succ_o y\ iff\ x\ \in C_2\ \wedge\ y \in C_1 \backslash C_2, onde$$

$$C_2 = \begin{cases} \{o \mid (o, v_1) \in P\}\ \text{se}\ (hasOperator.=) \\ \{o \mid (o, v) \in P \wedge\ v < v_1\}\ \text{se}\ (hasOperator.<) \\ \{o \mid (o, v) \in P \wedge\ v \leq v_1\}\ \text{se}\ (hasOperator.\leq) \\ \{o \mid (o, v) \in P \wedge\ v > v_1\}\ \text{se}\ (hasOperator.>) \\ \{o \mid (o, v) \in P \wedge\ v \geq v_1\}\ \text{se}\ (hasOperator.\geq) \end{cases}$$

Exemplo

$$PositiveDatatypePreference\ com\ hasOWLProperty.temCor,$$
$$hasDatatypePropertyValue.50000,$$
$$hasOperator.<\ e\ hasContext.Veículo, implica\ que$$

$$x \succ_o y\ iff\ x \in C \wedge\ y \in Veículo \backslash C\ onde\ C = \{o \mid (o, v) \in temPreco \wedge v < 50000\}, C \subseteq \text{Veículo}$$

O exemplo acima representa a preferência por todas as instâncias que possuem a propriedade *temPreço* cujo valor é menor que 50000 e que sejam instâncias de subclasses da classe *Veículo* uma vez que a preferência refere-se ao contexto de veículos.

Definição 8. Uma *Preferência Negativa de Propriedade de Tipos de Dados* é uma *Preferência de Tipo de Dados* que expressa uma preferência sobre os indivíduos cuja propriedade, especificada na relação *hasOWLProperty*, possui valor que não se enquadra dentro da combinação entre o valor especificado na relação *hasDatatypePropertyValue* e o operador especificado na relação *hasOperator*. Note que sintaticamente equivale à *Preferência Positiva de Propriedades de Tipos de Dados*, porém diferem semanticamente.

Sintaxe

$$
\begin{aligned}
&NegativeDatatypePreference \\
&\quad \sqsubseteq DatatypePropertyPreference \\
&\quad \sqcap (\leq 1\ \forall hasOWLProperty.U) \\
&\quad \sqcap (1 \leq \forall hasDatatypePropertyValue.v) \\
&\quad \sqcap \big((\exists hasOperator.<) \sqcup (\exists hasOperator.\leq) \\
&\quad \sqcup (\exists hasOperator.=) \sqcup (\exists hasOperator.>) \\
&\quad \sqcup (\exists hasOperator.\geq)\big) \sqcap (\leq 1 \forall hasContext.C)
\end{aligned}
$$

Semântica

$$
\begin{aligned}
&Seja\ C_1\ uma\ classe\ que\ representa\ o\ contexto\ tal\ que\ C_1 \subseteq owl{:}Thing, \\
&\quad C_2\ uma\ classe\ tal\ que\ C_2 \subseteq C_1, \\
&\quad P\ uma\ classe\ que\ representa\ uma\ propriedade\ concreta\ onde\ P \subseteq U, \\
&\quad D\ uma\ classe\ que\ representa\ o\ domínio\ de\ valores\ de\ tipo\ de\ dados, \\
&\quad v_1\ um\ literal\ que\ representa\ um\ valor\ da\ propriedade\ P\ onde\ v_1 \in D, \\
&\quad NegativeDatatypePreference\ \text{é}\ definida\ como\ sendo
\end{aligned}
$$

$$x \prec_o y\ iff\ x\ \in C_2\ \wedge\ y \in\ C_1 \backslash C_2, onde$$

$$
C_2 = \begin{cases}
\{\, o \mid (o, v_1) \in P\} \text{ se } (hasOperator.=) \\
\{o \mid (o, v) \in P \wedge\ v < v_1\} \text{ se } (hasOperator.<) \\
\{o \mid (o, v) \in P \wedge\ v \leq v_1\} \text{ se } (hasOperator.\leq) \\
\{o \mid (o, v) \in P \wedge\ v > v_1\} \text{ se } (hasOperator.>) \\
\{o \mid (o, v) \in P \wedge\ v \geq v_1\} \text{ se } (hasOperator.\geq)
\end{cases}
$$

Exemplo

$$
\begin{aligned}
&NegativeDatatypePreference\ \ com\ hasOWLProperty.temPreco, \\
&\quad hasDatatypePropertyValue.50000, \\
&\quad hasOperator.>\ \ e\ hasContext.Veículo, implica\ que
\end{aligned}
$$

$$x \prec_o y \; iff \; x \in C \wedge y \in Veículo \backslash C, onde \; C = \{o | (o, v) \in temPreco \wedge v > 50000\}, C \subseteq \text{Veículo}$$

O exemplo acima se trata justamente do inverso do exemplo anterior das preferências positivas. Ele representa a preferência por todas as instâncias que possuem a propriedade *temPreço* cujo valor não sejam maior que 50000 e que sejam instâncias de subclasses da classe *Veículo*, uma vez que a preferência refere-se ao contexto de veículos.

Na Figura 23 temos alguns exemplos dessas preferências já citadas na sintaxe RDF/XML de OWLPREF. O primeiro exemplo representa a preferência instâncias que tenham a propriedade cor, cujo valor está especificado em termos de uma *string*, no caso "azul". No segundo exemplo, temos uma representação de uma preferência por preços menores que 50.

```
<PositiveDatatypePreference rdf:ID= "PreferenciaCorAzul">
        <hasOWLProperty rdf:resource= "#temCor />
        <hasOperator rdf:resource= "#equal" />
        <hasDatatypePropertyValue rdf:datatype= "&xsd;string"> azul
        </hasDatatypePropertyValue>
        <hasContext rdf:resource= "#Veiculo" />
</PositiveDatatypePreference>
```

```
<PositiveDatatypePreference rdf:ID= "PreferenciaPrecoMenor50">
        <hasOWLProperty rdf:resource= "#temPreco />
        <hasOperator rdf:resource= "#lessThan" />
        <hasDatatypePropertyValue rdf:datatype= "&xsd;int">50
        </hasDatatypePropertyValue>
        <hasContext rdf:resource= "#Veiculo" />
</PositiveDatatypePreference>
```

Figura 22. Exemplo de preferências positivas sobre propriedades concretas com operadores: "Prefiro cor azul" e "Prefiro Preços Menores que 50" (ambas no contexto de veículos)

Outro subtipo de preferências relacionadas às propriedades concretas são as chamadas preferências extremas (*ExtremalPreference)*. Nesses tipos de preferências, o objetivo sempre é maximizar ou minimizar os valores de uma determinada propriedade concreta. Ou seja, diferente das positivas e negativas nas quais se especifica os valores desejáveis ou não desejáveis, nas preferências extremas somente declara-se que o desejável é o máximo ou mínimo valor de uma propriedade.

Em uma *Preferência Máxima* (*MaximumPreference)*, o maior valor encontrado da propriedade é preferível em relação aos outros menores. Vejamos um exemplo: *prefira o*

processador mais veloz. Neste caso, a preferência é por indivíduos do conceito Processador que tenha na propriedade velocidade o maior valor encontrado. Já a *Preferência Mínima* o menor valor encontrado é o preferível em relação aos outros maiores, tratando-se, portanto, do inverso da *Preferência Máxima*. Por exemplo: *prefira o mouse mais barato,* equivale a uma *Preferência Mínima* no atributo preço resultando na preferência por indivíduos que tenham o menor valor no atributo preço. Suas definições:

Definição 9. Uma *Preferência Máxima* é uma preferência extrema onde existe a preferência pelos indivíduos que possuem o maior valor possível na propriedade especificada na relação *hasOWLProperty* dentro do contexto especificado na relação *hasContext.*

Sintaxe

$$MaximumPreference \sqsubseteq ExtremalPreference \sqcap (\leq 1\, \forall hasOWLProperty.U) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$Seja\ C_1\ uma\ classe\ que\ representa\ o\ contexto\ tal\ que\ C_1 \subseteq owl{:}Thing,$
$C_2\ uma\ classe\ tal\ que\ C_2 \subseteq C_1,$
$P\ uma\ classe\ que\ representa\ uma\ propriedade\ concreta\ onde\ P \subseteq U,\ \ P_{ord_d}\ um\ conjunto\ ordenado\ de\ P\ em\ ordem\ decrescente\ tal\ que\ P_{ord_d} = \{(o, v_i), (o, v_{i+1}), \dots, (o, v_n)\}\ tal\ que\ (v_i > v_{i+1} >, \dots, > v_n),$
$MaximumPreference\ é\ definida\ como\ sendo$

$$x \succ_o y\ iff\ x \in C_2 \wedge y \in C_1 \backslash C_2, onde\ C_2 = \{\, o \mid (o, v_1) \in P_{ord_d}\}$$

Exemplo

$MaximumPreference\ com\ hasOWLProperty.temVelocidade\ e\ hasContext.Veículo,$
$implica\ que$

$$x \succ_o y\ iff\ x \in C \wedge y \in Veículo \backslash C\ onde\ C = \{o | (o, v_1) \in temVelocidade_{ord_d}\}, C \subseteq \text{Veículo}$$

O exemplo acima representa a preferência pelos indivíduos que possuem o maior valor possível da propriedade *temVelocidade* e que sejam instâncias de subclasses da classe *Veículo,* uma vez que a preferência refere-se ao contexto de veículos. Em outras palavras podemos dizer que existe uma preferência por veículos que possuam a maior velocidade.

Definição 10. Uma *Preferência Mínima* é uma preferência extrema onde existe a preferência pelos indivíduos que possuem o menor valor possível na propriedade especificada na relação *hasOWLProperty* dentro do contexto especificado na relação *hasContext.*

Sintaxe

$$MinimumPreference \sqsubseteq ExtremalPreference \sqcap (\leq 1\ \forall hasOWLProperty.U) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$$Seja\ C_1\ uma\ classe\ que\ representa\ o\ contexto\ tal\ que\ C_1 \subseteq owl{:}Thing,$$
$$C_2\ uma\ classe\ tal\ que\ C_2 \subseteq C_1,$$
$$P\ uma\ classe\ que\ representa\ uma\ propriedade\ concreta\ onde\ P \subseteq U,$$
$$P_{ord_c}\ um\ conjunto\ ordenado\ de\ P\ em\ ordem\ crescente\ tal\ que\ P_{ord_c} = \{(o, v_i), (o, v_{i+1}), \dots, (o, v_n)\}\ tal\ que\ (v_i < v_{i+1} <, \dots, < v_n),$$
$$MinimumPreference\ é\ definida\ como\ sendo,$$

$$x \succ_o y\ iff\ x \in C_2 \wedge y \in C_1 \backslash C_2, onde\ C_2 = \{\, o \mid (o, v_1) \in P_{ord_c}\}$$

Exemplo

$$MinimumPreference\ com\ hasOWLProperty.temPreco\ e\ hasContext.Veículo,$$
$$implica\ que$$

$$x \succ_o y\ iff\ x \in C \wedge y \in Veículo \backslash C\ onde\ C = \{o | (o, v_1) \in temPreco_{ord_c}\}, C \subseteq \text{Veículo}$$

Já no exemplo acima temos o inverso da preferência máxima. Neste exemplo existe uma preferência por todos os indivíduos que possuem a propriedade *temPreço* com o menor valor possível dentro do contexto de veículos. Isso significa em outros termos que existe uma preferência pelos veículos mais baratos (menores preços).

Na Figura 24 temos esses dois exemplos de preferências extremas OWLPREF na sintaxe RDF/XML. No primeiro quadro temos a *Preferência Máxima* na propriedade *#temVelocidade* representando uma preferência veículos com maior velocidade, enquanto que no segundo quadro temos uma *Preferência Mínima* na propriedade *#temPreco* representando uma preferência pela veículos mais baratos.

```
<MaximumPreference rdf:ID= “PreferenciaVelocidadeMaxima”>
        <hasOWLProperty rdf:resource= “#temVelocidade />
        <hasContext rdf:resource= “#Veiculo”/>
</MaximumPreference>
```

```
<MinimumPreference rdf:ID= “PreferenciaMenorPreco”>
        <hasOWLProperty rdf:resource= “#temPreco />
        <hasContext rdf:resource= “#Veiculo”/>
</MinimumPreference>
```

Figura 23. Exemplo de preferências extremas sobre veículos: “Prefiro Velocidade Máxima” e “Prefiro menor Preço possível”

4.4 Preferências Compostas

É possível a combinação de preferências com o intuito de representar prioridades ou igualdade de importância entre as preferências de OWLPREF. Para este fim definimos o conceito de *Preferências Compostas* (*CompositePreference).* As *Preferências Compostas* permitem essa combinação de preferências com objetivo de ampliar a expressividade de OWLPREF.

4.4.1 Combinação de Preferências

Com a universalidade de domínios e atributos existentes, aumenta cada vez mais a quantidade de preferências mais complexas e mais específicas. Com isso, se faz necessário criar mecanismos que representem relacionamentos entre essas preferências que chamamos de relacionamentos hierárquicos ou não hierárquicos.

Um relacionamento hierárquico de preferências, como próprio nome diz, define hierarquias ou prioridades entre as preferências. Ou seja, definem-se diferentes níveis de importância para cada tipo de preferência definida resultando em algo parecido como: *preferência 1* tem mais prioridade que *preferência 2* que por sua vez tem mais prioridade que *preferência n*, e assim sucessivamente. (Kießling & Köstler, 2002) define esse tipo de preferência como uma “cascateamento” de preferências, onde a preferência considerada mais importante deve ser aplicada primeiro seguida pela segunda mais importante e assim por diante.

Por exemplo, uma preferência por menor *Preço* pode ser mais importante que a preferência por *Pagamento com Cartão de Crédito*. Para representação desse tipo de hierarquia entre preferências, definimos um tipo de *Preferência Composta* chamada de *Preferência Ordenada*. Vejamos um exemplo: supondo que uma pessoa em um processo de aquisição de um computador tem como preferências o menor preço possível e o tamanho máximo de memória RAM, sendo que para esta pessoa o menor preço é mais importante que o tamanho da memória RAM. Modelando esse exemplo em OWLPREF teríamos:

- Uma *Preferência Mínima* na propriedade Preço (Pref_1)
- Uma *Preferência Máxima* na propriedade tamanho da memória (Pref_2)

Contudo, somente com essas duas preferências simples não estaríamos representando fielmente as preferências da pessoa do exemplo visto que, para ela, preço (representado por Pref_1) é mais importante que o tamanho da memória (representado por Pref_2). Logo, para representarmos essa prioridade entre preferências, utilizamos uma *Preferência Ordenada* implicando que $Pref\_1\ tem\ mais\ importância\ do\ que\ Pref\_2$.

(Kießling, Hafenrichter, Fischer, & Holland, 2001) ressalta ainda uma característica desse tipo de relacionamento hierárquico na qual eles definem como "priorização de preferências" ou "preferências de ordem lexicográfica". Adaptando-as para nosso contexto, os autores dizem que:

- Seja *preferência 1* uma preferência que é mais importante que *preferência 2,* então se dois elementos são igualmente importantes de acordo com a *preferência 1,* então a *preferência 2* é que irá decidir qual dos dois é o melhor.

Com isso, segue nossa definição de ordem ou importância entre preferências.

Definição 11. Sejam P_1 *e* P_2, duas preferências de OWLPREF. Uma relação de importância entre elas tal como $P_1 \gg_o P_2$, define uma ordem de aplicação de preferências onde P_1 é aplicada primeiro e P_2 é aplicada nos elementos resultantes da aplicação de P_1.

Vejamos um exemplo dessa característica no mesmo cenário da aquisição de um computador onde há uma preferência por um computador no qual o menor preço é mais

importante que o tamanho da memória. Suponha que o conjunto de computadores em que está sendo analisada a preferência é formado pelos quatro indivíduos listados abaixo.

- PC1(Memória=512; Preço=1200)
- PC2(Memória=512; Preço=1000)
- PC3(Memória=1024; Preço=1000)
- PC4(Memória=2048; Preço=1400)

Desse modo, de acordo com a preferência mais importante (menor preço), os computadores PC2 e PC3 são igualmente importantes, porém como a segunda preferência mais importante é o tamanho da memória, então o PC3 é o que mais atende as preferências desse usuário.

Note que a segunda preferência acaba sendo utilizada como critério de desempate entre os computadores que atenderam a preferência de maior prioridade.

Na *Preferência Ordenada* temos uma lista de preferências ordenada pela importância com intuito de explicitar a importância entre cada uma delas: a primeira da lista tem maior importância enquanto que a segunda da lista que tem mais importância que a seguinte até chegar a última da lista a qual tem a menor importância em relação às todas as outras. Sua definição:

Definição 12. Uma *Preferência Ordenada* é uma *Preferência Composta* contendo uma lista de preferências OWLPREF ordenada, especificada na relação *hasList,* a qual representa a ordem de importância dessas preferências entre si, onde a preferência mais importante é avaliada primeiro e as seguintes são avaliadas sucessivamente. Todas as preferências na lista devem possuir o mesmo contexto expresso na relação *hasContext.*

Sintaxe

$$OrderPreference \sqsubseteq CompositePreference \sqcap (\leq 1\, \forall hasList.PreferenceList) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$$Seja\ P\ uma\ preferência\ OWLPref\ tal\ que\ P \in Preference,$$
$$OrderPreference \text{ é } definido\ como\ uma\ lista\ ordenada\ L\ tal\ que$$

$$L = \{P_i, P_{i+1}, \ldots, P_n\}\ onde\ (P_i \gg_o P_{i+1} \gg_o, \ldots, P_n)$$

Vejamos um exemplo: suponha que no contexto dos veículos existem três preferências de classes distintas (carros, motos e bicicletas). Agora suponha que uma pessoa tem como preferência não só por carros ou motos ou bicicletas isoladamente, mas uma preferência pelas três classes com uma ordem de importância entre elas: prefere carros à motos e motos à bicicletas. A representação dessa ordem de importância entre preferências em OWLPREF é realizada através de uma *Preferência Ordenada* contendo as três preferências de classes (*Carro*, *Moto*, *Bicicleta* - apresentadas na Figura 25), as quais estão ordenadas em uma lista (ver Figura 25). Por fim na Figura 27 mostramos a preferência ordenada deste exemplo.

```
<ClassPreference rdf:ID="ClassPrefer_Bicicleta">
    <hasOWLClass rdf:resource="#Bicicleta"/>
    <hasContext rdf:resource= “#Veiculo”/>
</ClassPreference>
```

```
<ClassPreference rdf:ID="ClassPrefer_Carro">
    <hasOWLClass rdf:resource="#Carro"/>
    <hasContext rdf:resource= “#Veiculo”/>
</ClassPreference>
```

```
<ClassPreference rdf:ID="ClassPrefer_Moto">
    <hasOWLClass rdf:resource="#Moto"/>
    <hasContext rdf:resource= “#Veiculo”/>
</ClassPreference>
```

Figura 24. Preferências de Classes sobre três conceitos: Carro, Moto e Bicicleta

```
<PreferenceList rdf:ID="PreferenceList_1">
    <tail rdf:resource="#PreferenceList_2"/>
    <head rdf:resource="#ClassPref_Carro"/>
</PreferenceList>
<PreferenceList rdf:ID="PreferenceList_2">
    <tail rdf:resource="#PreferenceList_3"/>
    <head rdf:resource="#ClassPref_Moto"/>
</PreferenceList>
<PreferenceList rdf:ID="PreferenceList_3">
    <tail rdf:resource="#nil"/>
    <head  rdf:resource="#ClassPref_Bicicleta"/>
</PreferenceList>
```

Figura 25. Lista de Preferências contendo as três Preferências de Classes da Figura 25

```
<OrderPreference rdf:ID="OrderPreference_Carro_Moto_Bicileta">
    <hasList rdf:resource="#PreferenceList_1"/>
    <hasContext rdf:resource= “#Veiculo”/>
</OrderPreference>
```

Figura 26. Preferência Ordenada contendo uma lista de preferências com a ordem de importância de entre instâncias de Carro, Moto e Bicicleta

Já em um relacionamento não hierárquico, as preferências possuem entre si o mesmo grau de importância. Para representar esse tipo de preferência nós definimos a *Preferência Composta de Pareto* (também conhecida como Cumulativa, Conjuntiva ou de *Pareto-optimal*) (Kießling, Hafenrichter, Fischer, & Holland, 2001), (Kießling & Köstler, 2002). Por exemplo, suponha essas duas preferências: *Prefira Carro Zero km* / *Prefira Carro Branco*. Se para o agente B, tanto faz o carro ser zero km ou branco, isto é o fato de o carro ser zero km é igualmente importante ao fato de ele ter a cor branca, então essa preferência pode ser representada como uma preferência composta de Pareto.

Essa preferência segue o princípio do ótimo de Pareto (que também é conhecido como Eficiência de Pareto) na qual corresponde a uma situação em que é impossível melhorar a situação de alguns consumidores sem prejudicar simultaneamente qualquer outro. Existem outros trabalhos que também utilizam o princípio do ótimo de Pareto para combinação de preferências como em (Kohncke & Balke, 2006), (Khatib, Morris, Morris, & Venable, 2003), (Brafman & Friedman, 2006). Formalmente definimos Preferência de Pareto como:

Definição 13. Uma *Preferência de Pareto* é uma *Preferência Composta* onde existe um conjunto de preferências especificadas na relação *hasPreferenceSet* nas quais são igualmente importantes entre si dentro de um mesmo contexto expresso na relação *hasContext.*

Sintaxe

$$ParetoPreference \sqsubseteq CompositePreference \sqcap (\geq 1\, \forall hasPreferenceSet.Preference) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$$Seja\ P\ uma\ preferência\ OWLPref\ tal\ que\ P \in Preference,$$
$$ParetoPreference \text{ é } definido\ como\ um\ conjunto\ S\ tal\ que$$
$$S = \{P_i, P_{i+1}, \dots, P_n\}\ onde\ (P_i \succ_o P_{i+1} \succ_o, \dots, P_n)$$

Se no exemplo, citado anteriormente, da aquisição de um computador a preferência por menor preço fosse tão importante quanto o tamanho da memória, então, ao invés de utilizarmos uma *Preferência Ordenada*, utilizaríamos uma *Preferência de Pareto,* como no exemplo da Figura 27.

```
<ParetoPreference rdf:ID="MenorPreco_Igualmente_Importante_MaiorMemoria">
      <hasPreferenceSet rdf:resource="#PreferenciaMinimaPreco"/>
      <hasPreferenceSet rdf:resource="#PreferenciaMaximaMemoria"/>
      <hasContext rdf:resource= “#Computador”/>
</ParetoPreference>
```

Figura 27. Preferência de Pareto representando que a preferência por menor preço é tão importante quanto à preferência por tamanho máximo de memória no contexto de computadores

É possível também combinarmos não só preferências simples, mas também combinarmos preferências compostas com simples e até mesmo compostas com compostas como forma de ampliar a capacidade de representação de OWLPREF.

Vejamos outro exemplo: João está interessado em adquirir um pacote turístico para uma viagem. João prefere viajar para a Europa. Menos importante que a Europa é se o meio de tranporte do pacote é de navio ou de avião. Porém, acima de tudo, João prefere ter o menor custo possível. Modelando essas preferências em OWLPREF, teríamos inicialmente quatro preferências sobre pacotes turísticos:

- Preferência positiva por pacotes cujo destino é Europa.
    - Preferência 1 - Destino(Europa).
- Preferência positiva por pacotes cujo meio de tranporte seja navio.
    - Preferência 2 - MeioDeTransporte(Navio).
- Preferência positiva por pacotes cujo meio de tranporte seja avião.
    - Preferência 3 - MeioDeTransporte(Avião).
- Preferência mínima por pacotes que tenham o menor custo.
    - Preferência 4 – CustoTotal(menor)

Com isso combinamos essas preferências através das preferências compostas com o objetivo de definir hierarquia entre elas e com isso representar corretamente a preferência real de João. Na Figura 28 mostramos os passos. São eles:

1. Se para João é indiferente viajar de navio ou de avião, então podemos concluir que essas duas preferências em relação ao meio de transporte são igualmente importantes. Portanto, como primeiro passo definiu-se uma Preferência de Pareto composta pelas duas preferências relacionadas ao meio de transporte (preferência 2 e preferência 3) - ver primeiro quadro do lado esquerdo da Figura 29.

2. Porém, João acha mais importante que o destino seja a Europa do que qual vai ser o meio de transporte (navio ou avião). Então como segundo passo, define-se uma preferência ordenada composta pela preferência pelo destino Europa (preferência 1) seguido da preferência de meios de transporte (preferência 5 definida no passo anterior) – ver quadro central da Figura 29.
3. Por fim, já que João considera que o custo total é mais importante do que todas as outras preferências, define-se outra preferência ordenada composta pela preferência pelo menor custo (preferência 4) seguido da preferência ordenada modelada no passo 2 (preferência 6) – ver quadro direito da Figura 29.

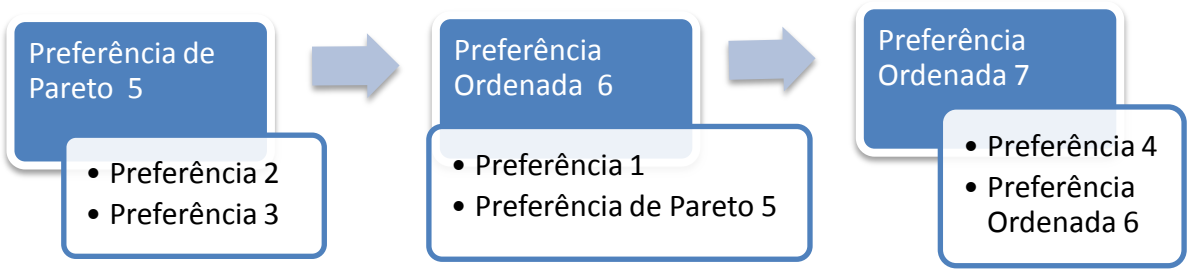


Figura 28. Exemplo do processo de combinação de preferências compostas com outras preferências tanto compostas quanto simples

Desse modo constituímos uma única preferência composta por diversas outras preferências hierarquicamente aninhadas de acordo com as prioridades definidas pelo usuário. Na Figura 30 temos uma representação visual dessas preferências aninhadas de acordo com tipo de preferência composta utilizada, uma englobando as outras.

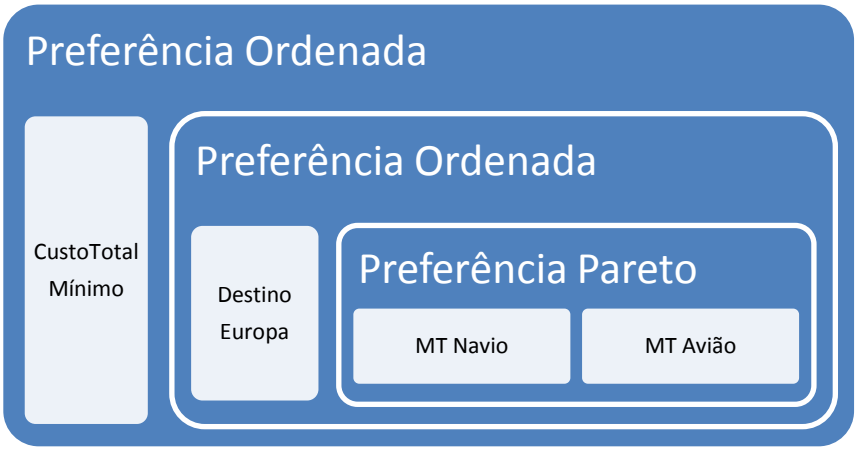


Figura 29. Preferências aninhadas representando hierarquia de prioridades entre as subpreferências

4.4.2 Novos Conceitos

Outra possibilidade que a combinação de preferências de OWLPREF permite é a criação de novos conceitos. Utilizando as *Preferências Simples* como construtores e as *Preferências Compostas* como operadores, podemos definir novos conceitos de preferências mais específicas permitindo que OWLPREF seja bastante extensível.

Vejamos um conceito.

A *Preferência de Intervalos* (*IntervalPreference)* representa a preferência instâncias que possuem um determinado valor dentro de um intervalo ou faixa de valores como no exemplo: *prefira passagens aéreas que custem entre R$10,00 e R$20,00.* Ou seja, os as passagens aéreas que tiverem o valor de custo dentro do intervalo de valores [10..20] terão preferência em relação a outras. Sua definição poderia ser feita da forma abaixo:

$$IntervalPreference \sqsubseteq DatatypePropertyPreference \sqcap (\leq 1 \forall hasInitialValue.v) \sqcap (\leq 1 \forall hasFinalValue.v) \sqcap (\leq 1 \forall hasOWLProperty.U) \sqcap (\leq 1 \forall hasContext.C)$$

Ou seja, uma *Preferência de Intervalos* poderia ser um tipo de *Preferência de Tipo de Dados* onde a propriedade e contexto estariam expressos nas relações já anteriormente citadas (*hasOWLProperty* e *hasContext*, respectivamente) e duas novas relações seriam criadas para expressas o intervalo de valores *hasInitialValue* e *hasFinalValue.*

Porém, se avaliarmos a definição do conceito em si da *Preferência de Intervalos,* podemos concluir que se podemos representá-la através da combinação das preferências de OWLPREF já apresentadas neste Capítulo. Ou seja, uma *Preferência de Intervalos* equivale a uma combinação não hierárquica (*Preferência de Pareto*) de duas preferências positivas em que a primeira representa o início do intervalo e a segunda o fim do intervalo. Portanto usando a combinação de preferências de OWLPREF, definimos *Preferência de Intervalos* como se segue.

Definição 14. Uma *Preferência de Intervalos* é uma *Preferência de Pareto* composta por duas *Preferências Positivas de Tipo de Dados* nas quais a primeira possui como restrição o operador menor igual e a segunda possui o operador maior igual na relação *hasOperator* de cada uma das duas preferências. Além disso, os valores das propriedades de cada uma dessas

preferências positivas correspondem aos valores limítrofes de uma preferência por faixa de valores.

Sintaxe

$$
\begin{aligned}
&IntervalPreference \\
&\quad \doteq ParetoPreference \\
&\quad \sqcap\ \exists hasPreferenceSet.((PositiveDatatypePreference \\
&\quad \sqcap (hasOperator.<) \sqcap (\exists hasDatatypePropertyValue.v_1) \\
&\quad \sqcap \exists hasOWLProperty.U)) \\
&\quad \sqcap \exists hasPreferenceSet.((PositiveDatatypePreference \\
&\quad \sqcap (hasOperator.>) \sqcap (\exists hasDatatypePropertyValue.v_2) \\
&\quad \sqcap (\exists hasOWLProperty.U)) \sqcap (\leq 1 \forall hasContext.C)
\end{aligned}
$$

Semântica

$Seja\ C_1\ uma\ classe\ que\ representa\ o\ contexto\ tal\ que\ C_1 \subseteq owl{:}Thing,$
$C_2\ uma\ classe\ tal\ que\ C_2 \subseteq C_1,$
$P\ uma\ classe\ que\ representa\ uma\ propriedade\ concreta\ onde\ P \subseteq U,$
$D\ uma\ classe\ que\ representa\ o\ domínio\ de\ valores\ tal\ que\ D \subseteq owl{:}Thing,$
$v_1 e\ v_2\ literais\ de\ P\ que\ representam\ os\ valores\ do\ intervalo\ tal\ que\ v_1, v_2 \in D,$
$IntervalPreference$ é $definida\ como\ sendo$

$$x \succ_o y\ iff\ x\ \in C_2\ \wedge\ y \in\ C_1 \backslash C_2, onde\ C_2 = \{\, o \mid (o,v) \in P \wedge v_1 < v < v_2 \}$$

Na Figura 30 temos o conceito de *Preferências de Intervalos* modelado na ferramenta Protege[23] (Protege, 2007) como uma *Preferência de Pareto* com três restrições:

- Uma *Preferência Positiva de Tipo de Dados* com operador maior que;
- Uma *Preferência Positiva de Tipo de Dados* com operador menor que;
- Possui apenas duas preferências.

[23] A maioria dos conceitos de OWLPREF foi modelada no Protege. No Apêndice I estão anexados todos os conceitos em RDF/XML gerados com auxílio da ferramenta.

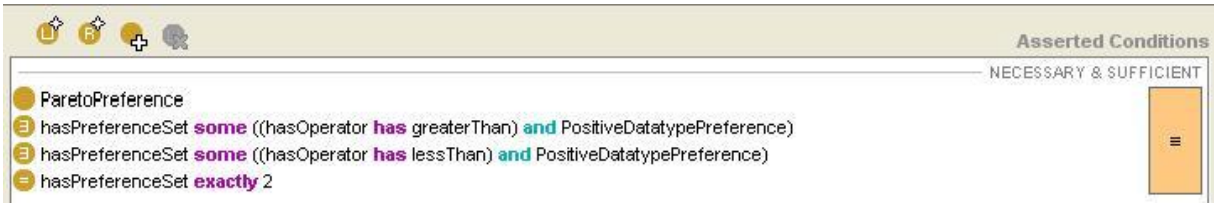


Figura 30. Preferência de Intervalos representada por uma *Preferência de Pareto* contendo duas preferências positivas como algumas restrições

Outros tipos de preferências também podem ser gerados utilizando os construtores de OWLPREF. Uma preferência, por exemplo, por instâncias que possuem certa propriedade com valores próximos a um determinado valor de referência é chamada, por alguns autores (Kießling, 2002), de preferências aproximadas (*around preference*). Um exemplo deste tipo de preferência poderia ser o seguinte: *prefira carros com ano de fabricação próximo a 2005.* Assim, supondo que exista uma determinada propriedade que represente a distância entre dois números, onde os anos 2004 e 2006 possuem como distância ao ano 2005 o valor 1; os anos 2003 e 2007 possuem como distância ao ano 2004, o valor 2 e assim sucessivamente. Com isso, uma preferência aproximada poderá ser representada utilizando uma *Preferência Mínima* de OWLPREF sobre essa propriedade relacionada ao ano de fabricação com valor 2005. Ou seja, os carros que possuam a menor distância ao valor 2005, no quesito ano de fabricação, serão considerados os mais próximos e, por isso, terão preferência sobre todos os outros carros.

4.4.3 Preferências e OWL

Outra vantagem do fato de OWLPREF ter sido modelado em OWL deve-se a possibilidade de representação de preferências em termos de classes especiais definidas em OWL como a representação de restrições.

Por exemplo, vamos supor uma ontologia que represente três classes (*Pessoa*, *Linguagem* e *Java*) e duas propriedades relativas às duas primeiras classes ("*programa*" e "*tipo*"). Suponha que exista uma preferência do tipo: "*Prefiro pessoas que programem em Linguagem Java".* Se a *Linguagem* do tipo *Java* estivesse modelada como uma única classe (classe *LinguagemJava* por exemplo) bastaríamos ter modelado uma preferência positiva especificado a propriedade "*programa*" com valor para essa classe (*programa=LinguagemJava).*

Nesse exemplo, no entanto, a modelagem foi feita com duas classes (classe *Linguagem* com *tipo=Java*). A representação da preferência seria então obtida com a criação de uma interseção (*owl:intersection*) entre a classe *Linguagem* com uma restrição (*owl:Restriction)* na propriedade *tipo* que obriga que todos os valores sejam da classe *Java.*

Na Figura 31, temos a representação desse exemplo onde o quadro branco da direita contém a interseção definida e a inclusão do mesmo como valor de uma propriedade da preferência positiva. É nesse momento que combinamos o poder de representação já existente em OWL com a complementação de nossa ontologia para representar preferências mais complexas e de forma mais adequada.

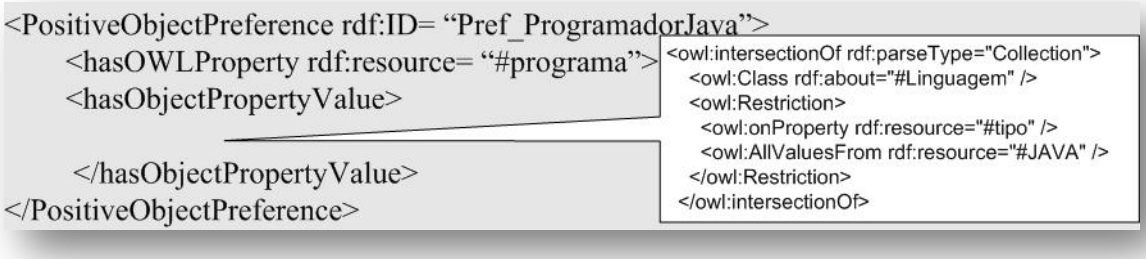


Figura 31. Combinação de representação em OWL e OWLPREF

4.5 Preferências Condicionais

A representação de preferências que variam de acordo com o contexto é possível através de preferências condicionais. Por exemplo, *se localização for Brasil, prefira processadores Pentium.* O conceito de *Preferência Condicional* é composto de duas propriedades (*onCondition* e *hasPreference),* além do contexto (*hasContext)* já explicado anteriormente. A primeira propriedade (*onCondition)* representa as condições em que a preferência se aplica (no caso do exemplo "*localização=Brasil*"). Já a segunda propriedade, *hasPreference,* refere-se à preferência propriamente dita que será aplicada, caso a condição, expressa na propriedade *onCondition,* seja verdadeira. As expressões condicionais, no caso, são representadas por um conceito de OWLPREF que chamamos de *Condição* (*Condition*). Esse conceito representa uma condição no formato (classe OWL, propriedade OWL, valor da propriedade OWL) semelhante às triplas RDF onde expressa-se qual classe OWL com determinada propriedade e valor deve ser verdadeira para que a preferência seja considerada.

Por exemplo, supondo que a condição seja "se o carro for azul", nesse caso, o conceito *Condição* de OWLPREF deve representar uma classe *Carro* com atributo *#temCor* com valor igual a "*azul*" ou representar um conceito que seja equivalente a "carro azul". Portanto, se no momento da aplicação da preferência existir conceito equivalente ao expressado na classe *Condição*, então a preferência será utilizada.

Figura 32. Preferência Condicional representa a interseção (se houver) dos três conjuntos. "Se localização for Brasil, prefira os processadores que são Pentium".

Na Figura 32, representamos o exemplo da preferência condicional por processadores *Pentium*. O conjunto maior representa o conjunto dos processadores. Entende-se que, caso a condição seja verdadeira (localização=Brasil), isto é o conjunto Brasil exista, então existe uma preferência pelos elementos da interseção entre o conjunto Brasil e o conjunto *Pentium*. Por exemplo, a condição (localização=Brasil) da Figura 40 pode ser expressa pelo conceito *Condição* atribuindo na expressão (classe, propriedade, valor) os valores (*Processador*, *Localização*, *Brasil*).

Dito isso, nós definimos *Preferência Condicional* de OWLPREF como segue abaixo.

Definição 16. Uma *Preferência Condicional* é uma *Preferência* de OWLPREF que possui uma pré-condição que deve ser válida especificada na relação *onCondition* para que a preferência especificada na relação *hasPreference s*eja aplicada dentro do contexto expresso na relação *hasContext.*

Sintaxe

$$ConditionalPreference \sqsubseteq Preference \sqcap (\leq 1 \forall hasPreference.Preference) \sqcap (\leq 1 \forall onCondition.Condition) \sqcap (\leq 1 \forall hasContext.C)$$

Semântica

$$Seja\ C\ classe\ que\ representa\ uma\ condição\ tal\ que\ C = \{0,1\},$$
$$P\ uma\ preferência\ tal\ que\ P \subseteq Preference,$$
$$ConditionalPreference \text{ é } definida\ como\ sendo$$

$$\exists P\ iff\ C = 0 \ \vee \nexists P\ iff\ C = 1$$

4.6 OWLPREF API

Somente a modelagem da ontologia de OWLPREF não seria suficiente para permitir que agentes ou outras aplicações pudessem trabalhar com conceitos de OWLPREF na Web Semântica. Isso exigiria o desenvolvimento de um raciocinador que interpretasse os conceitos da ontologia e retornasse dados de acordo com as preferências especificadas. Por esta razão, desenvolvemos uma API[24] (*Application Programming Interface)* ou interface para programação de aplicativos com o objetivo de permitir que esses agentes e aplicações possam interpretar e aplicar as preferências modeladas em OWLPREF.

4.6.1 Estrutura de OWLPREF API

Em termos gerais, OWLPREF API interpreta as preferências modeladas em OWLPREF e realiza as consultas levando em consideração as preferências especificadas. Para realizar essas consultas, a API faz o mapeamento dos conceitos de OWLPREF para sentenças de linguagens de consultas de dados semânticos (mais especificamente SPARQL *Preference*).

Conforme já discutido na Seção 2.4 do Capítulo 2, SPARQL é atualmente a linguagem de consulta de dados semânticos mais importante e utilizada. Essa estratégia de

[24] De modo geral uma API é um conjunto de rotinas, funções ou métodos estabelecidos por um biblioteca ou programa para a utilização de suas funcionalidades. A intenção é abstrair-se de detalhes de implementação e apenas utilizar suas funcionalidades.

mapeamento de conceitos de OWLPREF em SPARQL *Preference* permitiu focar-nos na modelagem da ontologia, abstraindo-se de detalhes de implementações e desempenho com relação à forma de como essa consulta aos dados será realizada.

De qualquer modo, implementações de outras linguagens de consulta podem ser desenvolvidas e/ou incorporadas à OWLPREF API de maneira similar como foi realizada com SPARQL. Para isso, seria necessária a geração de interfaces para a linguagem desejada e com isso adicioná-las às classes da OWLPREF API.

OWLPREF API foi desenvolvido na linguagem de programação Java[25], utilizando bibliotecas do *framework* Jena (Jena, 2007).

Basicamente, foram desenvolvidas classes Java representando cada um dos conceitos de OWLPREF e, através de métodos de manipulação de dados do Jena, relacionamos cada um dos conceitos de preferência de OWLPREF à classe correspondente em Java. Isso permitiu que trabalhássemos com mais eficiência no mapeamento para linguagens de consultas uma vez que trabalharíamos em uma linguagem comum (apenas Java). O seu fluxo de funcionamento é simples e consiste basicamente praticamente em três etapas: identificação de preferências, geração de consultas e execução das consultas.

Na etapa de identificação de preferências, OWLPREF API efetua a leitura das instâncias de preferências em OWLPREF com o objetivo de identificar os tipos de preferências que estão representandos. Após isso, as classes Java correspondentes a cada um dos tipos de preferências identificados são instanciadas para então se fazer a associação com o conceito de OWLPREF em modelado em OWL. Fazer essa associação significa informar à classe, que representa o conceito OWLPREF, todas as informações necessárias para a compreensão correta da preferência modelada (nome da propriedade, nome da classe, valores, etc.).

Após a identificação do conceito e, conseqüentemente, seu mapeamento na classe representante da preferência, a API possui condições de fazer a geração da consulta relativa à preferência identificada. Isso porque, dependendo do tipo de preferência, uma consulta pode ter formas diferentes de geração e necessitar (ou não) de determinados valores. Por exemplo, uma consulta com uma *Preferência Positiva* necessita da identificação de qual é a propriedade

[25] Mais informações sobre a linguagem Java podem ser encontradas no endereço http://java.sun.com

que se aplica e o valor da mesma. Entretanto, uma consulta com uma *Preferência Extrema* (máxima ou mínima) necessita apenas da identificação da propriedade visto, que o objetivo é a maximização/minimização do valor da propriedade, não fazendo sentido a especificação do valor da propriedade.

Com as preferências identificadas e as informações das mesmas inseridas nas classes correspondentes da API, passamos para a fase de geração de consultas. Neste estágio, ocorre o mapeamento propriamente dito do modelo de preferências de OWLPREF para sentenças SPARQL *Preference.* Esse mapeamento varia de acordo com o tipo das preferências onde, para cada conceito OWLPREF, definimos qual é a sentença SPARQL *Preference* correspondente. Com esse objetivo, uma tabela de correspondência foi construída contendo todas essas relações, permitindo assim que a API, baseando-se nas informações desta tabela, esteja apta a gerar as sentenças SPARQL *Preference* corretamente. A validação das sentenças geradas foi feita através do site http://prefs.l3s.uni-hannover.de/ disponibilizado pelos autores do SPARQL *Preference.* Para cada preferência da ontologia, uma instância foi definida e mapeada em uma sentença SPARQL.

Na Figura 33 apresentamos um exemplo[26] do mapeamento de uma *Preferência Positiva de Tipo de Dados* para uma sentença SPARQL *Preference* equivalente. A preferência positiva representa uma preferência por instâncias da classe *Hotel* que possuem a propriedade número de estrelas (*#numStars)* com valor igual 5 (ou seja, uma preferência por hotéis cinco estrelas). Note que a propriedade destacada na linha 2 (*#numStars)* da preferência positiva corresponde à propriedade destacada na linha 4 da sentença SPARQL *Preference.* O mesmo acontece com o operador (destacado linha 3 da preferência) que a API converte para o operador de igual (=) da linha 6 da sentença e com o valor da propriedade (linha 4 da preferência para linha 6 da sentença). Por fim, o contexto (linha 6 do primeiro quadro) é convertido como valor do predicado *rdf:type* (linha 5 do segundo quadro) uma vez que representa instâncias da classe *Hotel.* Ou seja, a API identifica que trata-se de uma *Preferência Positiva de Tipo de Dados* e, com isso, verifica na tabela de correspondência que a propriedade especificada em *hasOWLProperty* (linha 2 da preferência) equivale à tripla especificada dentro do modificador *WHERE* da sentença SPARQL *Preference.* O mesmo processo é repetido para os outros itens destacados até que a sentença completa fique pronta.

[26] No Apêndice II demonstramos outros possíveis exemplos.

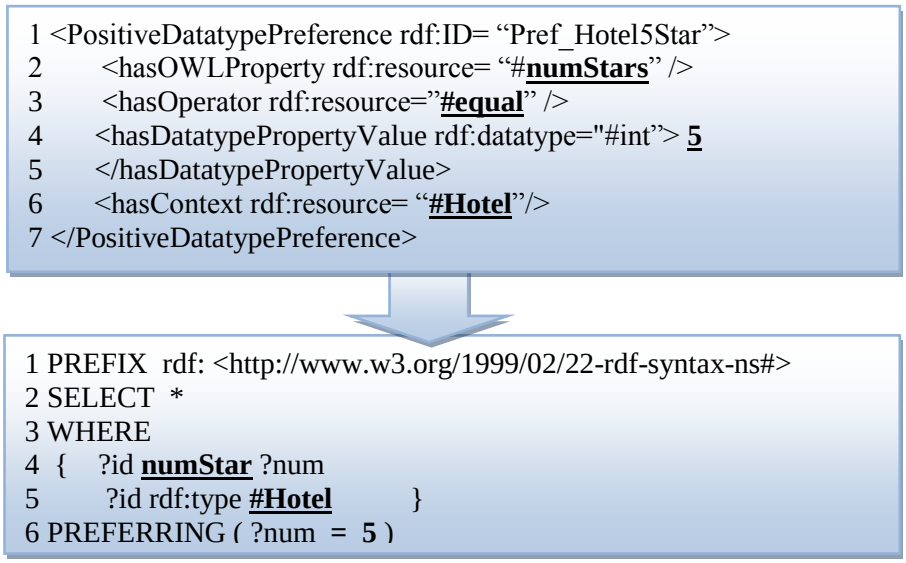


Figura 33. Exemplo de mapeamento de uma preferência em OWLPREF para uma sentença SPARQL *Preference*

Por fim, após a geração da sentença correspondente à preferência OWLPREF especificada, chegamos à etapa de execução de consultas. Nessa etapa, OWLPREF *API* utiliza o próprio motor de inferência de SPARQL para realizar a consulta e retorna os dados de acordo com a sentença gerada.

Para a execução propriamente dita, OWLPREF API disponibiliza uma classe que pode ser utilizada por um agente/aplicação que necessita de dois parâmetros: os dados (base de instâncias RDF/OWL na qual será realizada a consulta) e uma instância OWLPREF (preferência específica). Baseado nesses parâmetros, OWLPREF API funciona como uma espécie de “filtro”, onde são retornadas, dentro da base de instâncias especificada, todas as instâncias que atendem as preferências identificadas pela API. Caso não haja nenhuma instância compatível com a preferência repassada, então esta preferência é simplesmente ignorada.

5 Prova de Conceito

Como forma de demonstrar uma aplicação prática de OWLPREF, apresentamos neste Capítulo a utilização de OWLPREF em uma prova de conceito onde agentes compartilham conhecimento para atingir um objetivo comum.

Escolhemos o cenário de um processo de configuração de peças de computador, descrito com mais detalhes em (Furtado, Ayres, & Fernandes, 2007). Nessa proposta, uma abordagem multiagente foi utilizada para realizar um processo de montagem e configuração de um computador através de um pedido de um cliente, em que cada agente representa diversos fornecedores distribuídos e, com isso, possuem acessos a bases distintas.

O fluxo do processo é baseado no processo de manufatura de computadores da Dell descrito em (Greenwood & Calisti, 2004). Resumidamente, o fluxo do processo é o seguinte:

- O cliente faz um pedido através do telefone ou da Web;
- O pedido do cliente contém informações como: características especiais desejadas, preferências sobre as peças, preferências sobre pagamento, dados pessoais etc.;
- O pedido é analisado para que a configuração do computador seja produzida, levando em consideração as informações de fábricas espalhadas pelo mundo, além das informações repassadas pelo cliente;
- A configuração final é apresentada ao usuário para que o mesmo confirme o pedido e o computador seja produzido.

Nessa aplicação, agentes interagem para cooperativamente encontrar propostas de peças que atendam às restrições e preferências do cliente até a montagem final do produto. Quando um agente solicita proposta de peça para outro agente fornecedor, é necessário que este envie a preferência do usuário modelada em OWLPREF. Com isso, o agente fornecedor propõe as peças que mais se adequam às preferências repassadas.

5.1 Descrição do Fluxo

Na Figura 34 temos a descrição do fluxo desse processo de configuração de peças. As preferências do usuário modeladas em OWLPREF são repassadas ao agente pessoal 1 *(Personal-Agent 1)* que solicita a cooperação de outros agentes que representam fornecedores de peças. Com isso, repassa as preferências para cada um desses agentes que buscam peças de acordo com essas preferências nas bases de dados que possui acesso. Com isso oferecem propostas de peças ao agente que fez as requisições. Este avalia todas as propostas que recebe e escolhe a que tiver mais de acordo com as preferências do usuário. A partir de então, repete-se o fluxo até que todas as partes sejam configuradas e, com isso, o processo de configuração de um computador seja finalizado.

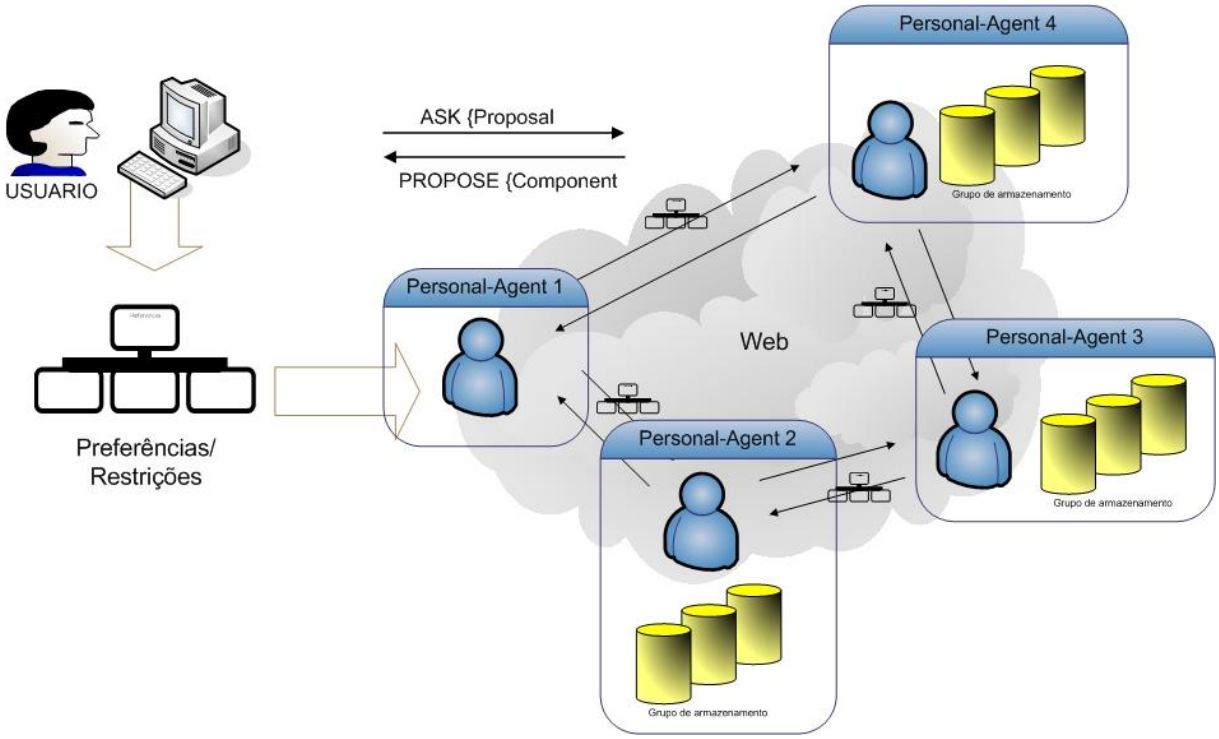


Figura 34. Fluxo do Processo de Configuração de peças através da cooperação de agentes

Vejamos um exemplo: Suponha que um cliente A efetuou um pedido de um disco rígido e informou que tem preferência por aqueles que tenham preço menor que R$ 200,00. Em OWLPREF essa preferência é modelada como uma *Preferência Positiva de Tipo de Dados* (*PositivaDatatypePreference)* com as seguintes propriedades:

- *hasOWLProperty* - refere-se à propriedade da instâncias em que a preferência se aplica. No caso, refere-se à propriedade Preço (*#price*) de uma ontologia comum entre os agentes que representa o domínio de computadores.
- *hasOperator* - define o operador que é aplicado na representação da preferência. Como no exemplo são os discos rígidos que custam menos que R$ 200,00 então se utiliza o operador *Menor Que* (*#lessThan*).
- *hasDatatypePropertyValue* - define o valor de preferência da propriedade referenciada no *hasOWLProperty*. No caso do exemplo, o valor R$ 200,00.
- *hasContext* – define o contexto em que a preferência se aplica. No caso do exemplo, o contexto refere-se ao domínio de discos rígidos (*#HardDisk).*

Na Figura 35, mostramos a preferência do cliente com as propriedades acima definidas que é repassada ao agente pessoal 1 (*Personal Agent 1*) para que solicite propostas de peças levando em consideração essa preferência.

```
<PositiveDatatypePreference rdf:ID= “Preco_Menor_200”>
        <hasOWLProperty  rdf:resource= “http://www.unifor.br/mia/pc.owl#price” />
        <hasOperator rdf:resource= “#lessThan” />
        <hasDatatypePropertyValue rdf:datatype="&xsd;float”> 200
        </hasDatatypePropertyValue>
        <hasContext rdf:resource= “#HardDisk”/>
</PositiveDatatypePreference>
```

Figura 35. Preferência do cliente A: discos rígidos com preços menores que R$ 200,00

De posse destas informações, o agente 1 tem condições de iniciar as solicitações de propostas de outras agentes, repassando a estes as preferências do cliente A.

5.2 Comunicação entre Agentes

Todas as solicitações de propostas feitas pelo agente 1, bem como toda a comunicação entre os outros agentes, são realizadas através de troca de mensagens. As mensagens seguem o padrão FIPA/ACL[27].

Na Figura 36 temos um exemplo do formato da mensagem que é enviada pelo agente 1 (*Personal-Agent 1*) ao agente 2 (*Personal-Agent 2).* Na parte destacada da Figura está especificada qual o tipo de peça que foi solicitado pelo cliente A, no caso um disco rígido (*#HardDisk*) e a preferência OWLPREF que representa os interesses do cliente A – discos rígidos com preços menores que R$ 200,00 modelado na Figura 35. Na Figura 37 temos a mensagem de retorno do agente 2 com as peças que encontrou, levando em consideração as preferências repassadas ao mesmo. Os resultados são gravados num arquivo chamado *result.owl* contendo as instâncias das peças para que possam acessadas pelo agente 1.

```
(REQUEST
  :sender(agent-identifier :name personalAgent1@mia.unifor.br)
  :receiver(set (agent-identifier :name personalAgent2@mia.unifor.br :addresses
(sequence http://grid10:7778/acc)) )
  :content "(propose http://mia.unifor.br/pc.owl#HardDisk),
            (preferences http://mia.unifor.br/preferences.owl#Preco_Menor_200)"
  :language fipa-sl
  :ontology configuration-ontology
  :protocol fipa-request
  :conversation-id PSMOWLS-MIA-1315817452067
)
```

Figura 36. Mensagem solicitando proposta de disco rígido com a preferência do cliente A: preços preferencialmente menores que 200.

[27] Mais informações sobre FIPA/ACL bem como especificações podem ser encontradas no endereço http://www.fipa.org/repository/aclspecs.html

```
(INFORM
 :sender(agent-identifier :name personalAgent2@mia.unifor.br)
 :receiver(set  (agent-identifier
          :name personalAgent1@mia.unifor.br
          :addresses (sequence http://grid10:7778/acc)))
 :content "((response
                (proposeProcess :input-string
                (propose http://mia.unifor.br/computer.owl#HardDisk),
                (preferences http://mia.unifor.br/preferences.owl#Preco_Menor_200)
                 proposeProcess :result (http://mia.unifor.br/Agent2/result.owl))
 ))"
 :language fipa-sl
 :ontology configuration-ontology
 :protocol fipa-request
 :conversation-id PSMOWLS-MIA-1210317290357
)
```

Figura 37. Mensagem de retorno com resultados da busca por discos rígidos de acordo com preferências do cliente A (preço menor que 200)

5.3 Estrutura Interna dos Agentes

Os agentes do estudo de caso foram desenvolvidos na plataforma JADE (JADE, 2000) que é ao mesmo tempo um ambiente de execução e biblioteca para desenvolvimento de agentes (Ayres, 2003). Sua estrutura interna é simples. No caso dos agentes que recebem propostas, o fluxo de funcionamento é o seguinte:

1. O agente recebe uma proposta.
2. O agente decide se aceita fazer a proposta.
3. Caso aceite, o agente utiliza OWLPREF API para fazer a leitura das preferências e gerar a consulta SPARQL *Preference* correspondente.
4. Realiza a consulta e envia os resultados ao agente solicitante.

O código utilizado para acessar OWLPREF API pode ser visto na Figura 38. A API disponibiliza um método onde dois parâmetros são requeridos: a URI indicando a preferência a ser utilizada e a URI indicando a localização dos dados a serem consultados. No caso do exemplo da Figura 38, o primeiro parâmetro é a preferência indicada na mensagem enviada pelo agente 1 e o segundo parâmetro são as bases de dados que o agente 2 possui acesso.

```
OWLPrefAPI.select (http://www.unifor.br/mia/preferences.owl#Preco_Menor_200,
                   http://www.unifor.br/mia/Agent2/database.owl);
```

Figura 38. Trecho do código de chamada da OWLPREF API realizada pelo agente 2

Na Figura 39 temos uma base de instâncias de disco rígido com características diferentes.

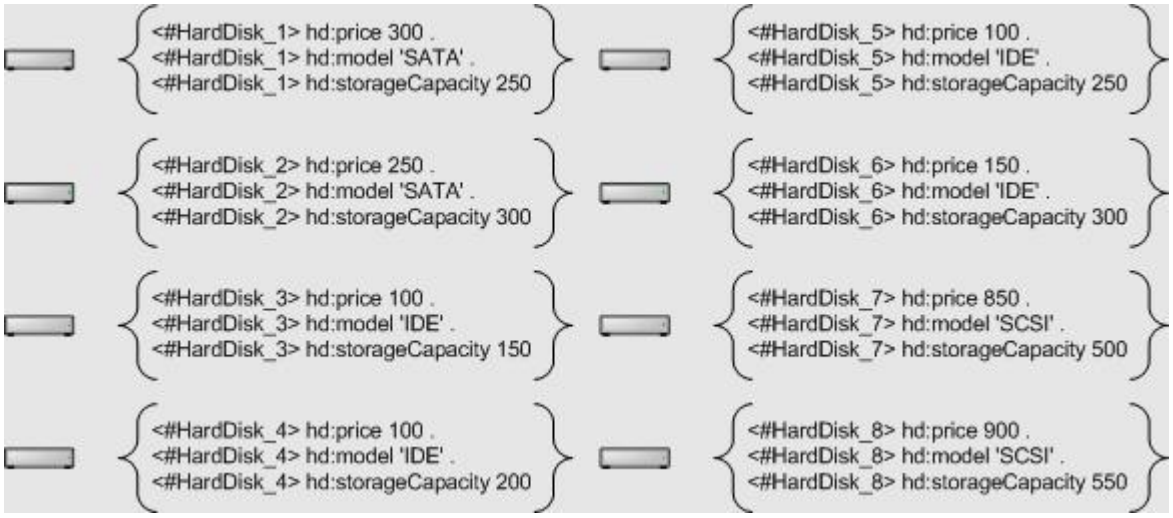


Figura 39. Exemplo de uma base contendo instâncias de discos rígidos com características diferentes

Supondo que essa base seja a que o agente 2 tem acesso e que se utiliza a preferência especificada na Figura 36, então, o módulo gerador de consultas SPARQL irá gerar a consulta da Figura 41 e com os respectivos dados como retorno (discos rígidos com preços menores que R$ 200,00 preferencialmente).

```
 1 PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
 2 PREFIX  rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 3 SELECT  *
 4 WHERE
 5   { ?id  rdf:type            pc:HardDisk ;
 6          pc:model            ?model ;
 7          pc:price            ?price ;
 8          pc:storageCapacity  ?storageCapacity .
 9   }
10 PREFERRING ( ?price < 200 )

-------------------------------------------------------------------------------------------------------
| id                                                                    | model | price | storageCapacity |
=======================================================================================================
| <file:///srv/www/vhosts/test/cgi-bin/d596274.n3#HardDisk_6> | "IDE" | 150   | 300             |
| <file:///srv/www/vhosts/test/cgi-bin/d596274.n3#HardDisk_5> | "IDE" | 100   | 250             |
| <file:///srv/www/vhosts/test/cgi-bin/d596274.n3#HardDisk_4> | "IDE" | 100   | 200             |
| <file:///srv/www/vhosts/test/cgi-bin/d596274.n3#HardDisk_3> | "IDE" | 100   | 150             |
-------------------------------------------------------------------------------------------------------
```

Figura 40. Consulta executada pelo agente 2 e os dados retornados baseados nas preferências

Esses dados são persistidos em um arquivo e enviados ao agente solicitante, conforme formato de mensagem apresentado na Figura 37.

Se, por acaso, o cliente A além da preferência citada, tivesse outra preferência (maior capacidade de disco rígido) e, ambas as preferências são igualmente importantes para o cliente A, então teríamos uma *Preferência de Pareto* composta pela preferência anterior e uma *Preferência Máxima* na propriedade capacidade de disco (*#storageCapacity)*. A consulta e os dados retornados seriam os apresentados na Figura 41

```
 1 PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
 2 PREFIX  rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 3 SELECT  *
 4 WHERE
 5   { ?id  rdf:type            pc:HardDisk ;
 6          pc:model            ?model ;
 7          pc:price            ?price ;
 8          pc:storageCapacity  ?storageCapacity .
 9   }
10 PREFERRING ( ?price < 200 ) AND HIGHEST(?storageCapacity)

-------------------------------------------------------------------------------------------------------
| id                                                                          | model  | price | storageCapacity |
=======================================================================================================
| <file:///srv/www/vhosts/test/cgi-bin/d596274.n3#HardDisk_8> | "SCSI" | 900   | 550             |
| <file:///srv/www/vhosts/test/cgi-bin/d596274.n3#HardDisk_6> | "IDE"  | 150   | 300             |
-------------------------------------------------------------------------------------------------------
```

Figura 41. Preferência pelo Preço Menor que R$ 200,00 é tão importante quanto Preferência pela Maior Capacidade de disco.

5.4 Considerações Finais sobre a Prova de Conceito

Com isso, mostramos nessa prova de conceito o compartilhamento de OWLPREF em um contexto em que agentes distribuídos se comunicam e trabalham cooperativamente com o mesmo objetivo: compartilhar o mesmo conhecimento, que no caso mais específico do nosso cenário, equivale ao mesmo tipo de preferência de um certo usuário.

Isto permitiu um maior número de propostas de peças mais de acordo com as preferências dos usuários, possibilitando resultados mais customizados. O não uso do compartilhamento dessas preferências iria ocasionar a proposta de peças que talvez não atendesse aos anseios ou preferências dos usuários.

6 Conclusão

6.1 Considerações Finais

Neste trabalho definimos OWLPREF como uma representação declarativa de preferências em OWL. Em termos gerais podemos dizer que OWLPREF trata-se de uma extensão de OWL para incluir a representatividade do conceito de preferências qualitativas. A principal contribuição de OWLPREF deve-se ao fato de permitir a representação de preferências em uma linguagem de ontologia compatível com os propósitos da Web Semântica. Essa característica permite que agentes e aplicações possam utilizar noções de preferências na personalização de resultados de buscas e na troca de mensagens para o compartilhamento de conhecimento.

Descrevemos a semântica de cada um dos conceitos de OWLPREF e mostramos como combinar os construtores de OWLPREF para a elaboração de novos conceitos de preferências. Mostramos ainda como a integração com os próprios construtores de OWL pode ampliar o poder de representação de preferências mais complexas. A expressividade de OWLPREF foi demonstrada através de exemplos dos diversos tipos de preferências que podem ser representados (positivas, negativas, máximas, mínimas, ordenadas, condicionais, aproximadas, intervalos etc.).

Outra característica importante é a facilidade de reuso e compartilhamento, bem como o suporte a inferências que OWLPREF possui, uma vez que foi implementada sob a linguagem OWL e, indiretamente, acaba herdando muito das características que OWL já possui.

Outra contribuição que identificamos está na independência de domínio. Isso acontece porque OWLPREF refere-se sempre a atributos e classes OWL, não importando qual tipo de domínio esses atributos e classes fazem parte nem sendo necessária qualquer alteração na ontologia de domínio, para que possam ser expressas preferências.

Por fim, entendemos como última contribuição deste trabalho a disponibilização de uma API que permite que preferências modeladas em OWLPREF possam ser aplicadas sobre dados RDF/OWL através do mapeamento para consultas SPARQL *Preference.*

6.2 Limitações de Escopo e Trabalhos Futuros

A expressividade de OWLPREF é limitada para a representação de preferências em domínios onde a questão temporal é relevante (Ngoc, Lee, & Lee, 2005). Extensões nessa direção podem ser perseguidas quer seja através do uso de *OWL-Time* (Hobbs & Pan, 2006) ou através da definição de uma lógica temporal como a seguida por (Bacchus, Baier, & McIlraith, 2007).

Este trabalho pode ser estendido ainda para que o processo de aquisição das preferências seja feito de forma interativa e amigável com os usuários. A automatização da captura das preferências dos usuários ainda é um problema, pois pode ser uma tarefa complexa e custosa para os mesmos.

Outra possível extensão refere-se à disponibilização de OWLPREF API como um Serviço Web. Desta forma, evitaríamos a necessidade de incluir o código da API na estrutura interna dos agentes que desejassem utilizar as funcionalidades da API. Bastaria, no caso, acionar as funcionalidades disponíveis em um Serviço Web. Este serviço, assim como a API, vai requerer dois argumentos: o primeiro seria a base de dados a ser consultada e o segundo as preferências OWLPREF a serem utilizadas.

Outros trabalhos futuros seriam integrações com outras linguagens baseadas em OWL como, por exemplo, OWL-S (Martin, et al., 2004). OWL-S é uma extensão de OWL para descrição de serviços Web que auxilia a automatização da localização e composição de serviços. No caso, OWLPREF pode ser utilizada para a especificação de preferências sobre serviços, utilizando os conceitos de OWL-S.

Acreditamos que o trabalho aqui descrito pode contemplar extensões no sentido de representar preferências condicionais através da modelagem de conceitos definidos em SWRL (Horrocks I. , Patel-Schneider, Boley, Tabet, Grosof, & Dean, 2004), com o intuito de

aproveitar o poder de expressividade dessa linguagem de representação de regras e manter-se dentro do quadro de linguagens de representação semântica para Web.

Por fim, ressaltamos que OWLPREF está disponibilizada na Web (http://unifor.s41.eatj.com/ontologies/owlpref.owl) a fim de que aplicações na Web Semântica passem a utilizá-la. As seguintes possibilidades podem ser vislumbradas:

- Personalização de conteúdos – Sites podem disponibilizar conteúdo mais apropriado ao usuário fazendo a "leitura" das preferências dos usuários no momento em que acessam seus endereços (preferências OWLPREF armazenadas como um tipo de *cookie*). Por exemplo: um site de compras pode apresentar primeiramente os produtos de acordo com a preferência do usuário; um site de notícias pode apresentar textos relacionados aos interesses dos usuários.
- Otimização de buscas – *search-engines* semânticas podem captar as preferências do usuário antes de retornar os resultados. Desta forma, a apresentação dos resultados sempre vai levar em consideração os gostos e interesses dos demandantes das consultas.
- Grids Semânticos – Trata-se de uma composição de máquinas ou recursos computacionais para processamento em larga escala. As preferências por processadores específicos podem ser modeladas em OWLPREF e com isso o escalonador de recursos pode considerar essas preferências no momento de disponibilização de certos recursos para determinados usuários ou cenários específicos.

Referências Bibliográficas

Abel, F., Herder, E., Kärger, P., Olmedilla, D., & Siberski, W. (setembro de 2007). Exploiting preference queries for searching learning resources. *2nd European Conference on Technology Enhanced Learning (EC-TEL)* .

Agrawal, R., & Wimmers, E. L. (Maio de 2000). A Framework for Expressing and Combining Preferences. *Proceedings of the ACM SIGMOD International Conference on Management of Data* , pp. 297-306.

Agrawal, R., Kiernan, J., Srikant, R., & Xu, Y. (2005). XPref: a preference language for P3P. *Computer Networks: The International Journal of Computer and Telecommunications Networking , 48,* pp. 809 - 827.

Antoniou, G., & Harmelen, F. v. (2004). *A Semantic Web Primer.* Cambridge, Massachusetts: The MIT Press.

Armstrong, W. E. (Março de 1948). Uncertainty and the Utility Function. *The Economic Journal , 58,* pp. 1-10.

ARQ. (Julho de 2007). Acesso em 2007, disponível em ARQ - A SPARQL Processor for Jena: http://jena.sourceforge.net/ARQ/

Ayres, L. (Junho de 2003). Estudo e Desenvolvimento de Sistemas Multiagentes usando JADE: Java Agent Development framework. *Monografia de conclusao de curso de Informática.*

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2002). *The Description Logic Handbook.* Cambridge: Cambridge University Press.

Bacchus, F., Baier, J. A., & McIlraith, S. A. (6-12 de Janeiro de 2007). A Heuristic Search Approach to Planning with Temporally Extended Preferences. *(IJCAI 2007) Proceedings of the 20th International Joint Conference on Artificial Intelligence* , pp. 1808-1815.

Balke, W. -T., Güntzer, U., & Lofi, C. (Junho de 2007). Incremental Trade-Off Management for Preference Based Queries. *International Journal of Computer Science & Applications (IJCSA) , 4 (2)*.

Balke, W.-T., & Wagner, M. (Junho de 2003). Cooperative Discovery for User-centered Web Service Provisioning. *Proceedings of the First IEEE International Conference on Web Services (ICWS 2003)* .

Barthélemy, J. -P., Flament, C., & Monjardet, B. (1989). Ordered sets and social sciences. *I. Rival (Ed.), Ordered Sets, D. Reidel, Dordrecht* , pp. 721-758.

Bechhofer, S., Harmelen, F. v., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., et al. (10 de Fevereiro de 2004). *OWL Web Ontology Language Reference.* Acesso em 19 de Julho de 2007, disponível em W3C Recommendation: http://www.w3.org/TR/2004/REC-owl-ref-20040210/

Beckett, D. (9 de Fevereiro de 2006). *SPARQL RDF Query Language Reference v1.8.* Acesso em 20 de Julho de 2007, disponível em SPARQL RDF Query Language Reference v1.8: http://www.dajobe.org/2005/04-sparql/SPARQLreference-1.8.pdf

Beckett, D. (4 de Dezembro de 2006). *Turtle - Terse RDF Triple Language.* Acesso em 20 de Julho de 2007, disponível em Turtle - Terse RDF Triple Language: http://www.dajobe.org/2004/01/turtle/

Beckett, D., & Broekstra, J. (14 de Junho de 2007). *SPARQL Query Results XML Format.* Acesso em 20 de Julho de 2007, disponível em W3C Working Draft: http://www.w3.org/TR/2007/WD-rdf-sparql-XMLres-20070614/

Beckett, D., & McBride, B. (10 de Fevereiro de 2004). *RDF/XML Syntax Specification (Revised).* Acesso em 18 de Julho de 2007, disponível em W3C Recommendation : W3C Recommendation 10 February 2004

Beckett, D., Biddulph, M., Dumbill, E., & McCarthy, P. (Julho de 2007). *Planet RDF*. Acesso em 20 de Julho de 2007, disponível em The PANTS Collective: http://planetrdf.com/

Berners-Lee, T. (18 de Setembro de 2006). *Keynote at AAAI Conference 2006.* Acesso em 19 de Julho de 2007, disponível em Keynote at AAAI Conference 2006: http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html

Berners-Lee, T. (2000). *Semantic Web - XML2000.* Acesso em 2007, disponível em W3C: http://www.w3.org/2000/Talks/1206-xml2k-tbl/

Berners-Lee, T., Hendler, J., & Lassila, O. (Maio de 2001). *The Semantic Web.* Fonte: Scientific American: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

Bistarelli, S., Montanari, U., & Rossi, F. (1995). Constraint Solving over Semirings. *Proceedings International Joint Conference on Artificial Intelligence (IJCAI'95)* .

Boley, H., & Kifer, M. (Março de 2007). *RIF Core Design.* Acesso em Julho de 2007, disponível em World Wide Web Consortium - Working Draft WD-rif-core-20070330: http://www.w3.org/TR/2007/WD-rif-core-20070330/

Borst, W. N. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse.* Acesso em 18 de Julho de 2007, disponível em Thesis (Phd): http://www.ub.utwente.nl/webdocs/inf/1/t0000004.pdf

Börzsönyi, S., Kossmann, D., & Stocker, K. (Abril de 2001). The Skyline Operator. *Proceedings of the 17th International Conference on Data Engineering* , pp. 421 - 430.

Boutilier, C. (1994). Toward a Logic for Qualitative Decision Theory. (J. D. Torasso, Ed.) *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)* , pp. 75--86.

Boutilier, C., Bacchus, F., & Brafman, R. I. (2001). UCP-Networks: A Directed Graphical Representation of Conditional Utilities. *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)* , pp. 56-64.

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H., & Poole, D. (Maio de 2004). Preference-Based Constrained Optimization with CP-nets. *Computational Intelligence , 20 Issue 2*, pp. 137-157.

Boutilier, C., Brafman, R. I., Hoos, H. H., & Poole, D. (1999). Reasoning With Conditional Ceteris Paribus Preference Statements. *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI'99)* .

Brafman, R. I., & Friedman, D. (2006). Adaptive Rich Media Presentations via Preference-Based Constrained Optimization. *Preferences: Specification, Inference, Applications* .

Brafman, R. I., & Tennenholtz, M. (1997). Modeling Agents as Qualitative Decision Makers. *Artificial Intelligence , 94*, pp. 217-268.

Bray, T., Hollander, D., Layman, A., & Tobin, R. (16 de Agosto de 2006). *Namespaces in XML 1.0.* Acesso em 19 de Julho de 2007, disponível em W3C Recommendation: http://www.w3.org/TR/2006/REC-xml-names-20060816

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (16 de Agosto de 2006). *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. Acesso em 20 de Julho de 2006, disponível em W3C Recommendation: http://www.w3.org/TR/2006/REC-xml-20060816

Brickley, D., & Guha, R. V. (10 de Fevereiro de 2004). *RDF Vocabulary Description Language 1.0: RDF Schema.* Acesso em 18 de Julho de 2007, disponível em W3C Working Draft: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

Brickley, D., & Miller, L. (24 de Maio de 2007). *FOAF Vocabulary Specification 0.9.* Acesso em 20 de Julho de 2007, disponível em Namespace Document 'Rehydrated' Edition: http://xmlns.com/foaf/spec/20070524.html

Brin, S., & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems* , pp. 107--117.

Broome, J. (1991). Weighting Goods. (B. Blackwell, Ed.)

Carrano, A. (1988). Establishing the order of human chromosome-specific DNA fragments. *Biotechnology and the Human Genome* , pp. 37-50.

Chamberlin, D. D., & Boyce, R. F. (1974). SEQUEL: A structured English query language. *International Conference on Management of Data, Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control* , pp. 249–264.

Chomicki, J. (Dezembro de 2003). Preference Formula in Relational Queries. *ACM Transactions on Database Systems , 28 (4)*, pp. 427-466.

Clark, J., & DeRose, S. (16 de Novembro de 1999). *XML Path Language (XPath) Version 1.0.* Acesso em 21 de Julho de 2007, disponível em W3C Recommendation: http://www.w3.org/TR/xpath

Clark, K. G. (25 de Março de 2005). *RDF Data Access Use Cases and Requirements.* Acesso em 20 de Julho de 2007, disponível em W3C Working Draft: http://www.w3.org/TR/2005/WD-rdf-dawg-uc-20050325/

Clark, K. G. (6 de Abril de 2006). *SPARQL Protocol for RDF.* Acesso em 20 de Julho de 2007, disponível em W3C Candidate Recommendation: http://www.w3.org/TR/2006/CR-rdf-sparql-protocol-20060406/

Dodds, D., Watt, A., Birbeck, M., Cousins, J., Rivers-Moore, D., Worden, R., et al. (2001). *Professional XML Meta Data.* Wrox Press.

Doyle, J., & Wellman, M. P. (1992). Modular utility representation for decision-theoretic planning. *Proceedings of the first international conference on Artificial intelligence planning systems* , pp. 236 - 242.

Doyle, J., & Wellman, M. P. (1994). Representing Preferences as Ceteris Paribus Comparatives. *Proceedings of the AAAI Spring Symposium on Decision-Theoretic Planning* , pp. 69--75.

Dubois, D., Prade, H., & Sabbadin, R. (2001). Decision-theoretic foundations of qualitative possibility theory. *European Journal of Operational Research 128* , pp. 459-478.

ESW Wiki. (2007). Acesso em 20 de Julho de 2007, disponível em Sparql Calendar Demo: http://esw.w3.org/topic/SparqlCalendarDemo

Faltings, B., Torrens, M., & Pu, P. (2004). Solution Generation with Qualitative Models of Preferences. (ACM, Ed.) *International Journal of Computational Intelligence* , pp. 246–263.

Fishburn, P. C. (6 de Abril de 1999). Preference structures and their numerical representations. *Theoretical Computer Science , 217 Number 2*, pp. 359-383(25).

Freitas, F. L. (2003). Ontologias e a Web Semântica. In: R. Vieira, & F. Osorio, *Anais do XXIII Congresso da Sociedade Brasileira de Computação* (Vols. 8 - Jornada de Mini-Cursos em Inteligência Artificial, pp. 1-52). Campinas: SBC.

Furtado, V., Ayres, L., & Fernandes, G. (Outubro de 2007). A Multiagent Approach for Configuring and Explaining Workflow of Semantic Web Services. *International Journal of Information Technology and Web Engineering* , pp. 1-20.

Greenwood, D., & Calisti, M. (10-13 de Outubro de 2004). Engineering Web service - agent integration. *IEEE International Conference on Systems, Man and Cybernetics 2004* , pp. 1918 - 1925.

Gruber, T. R. (Novembro de 1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies , 43* (4-5), pp. 907-928.

Ha, V., & Haddawy, P. (1999). A Hybrid Approach to Reasoning with Partially Elicited Preference Models. *Proceedings Fifteenth Conference on Uncertainty in Artificial Intelligence* , pp. 263--270.

Hansson, S. O. (2001). Preference Logic. (D. G. Guenthner, Ed.) *Handbook of Philosophical Logic (Second Edition) , 4*, pp. 319-393.

Heflin, J. (10 de Fevereiro de 2004). *OWL Web Ontology Language Use Cases and Requirements.* Acesso em 19 de Julho de 2007, disponível em W3C Recommendation: http://www.w3.org/TR/2004/REC-webont-req-20040210/

Hillmann, D. (7 de Novembro de 2005). *Dublin Core Metadata Initiative.* Acesso em 20 de Julho de 2007, disponível em Using Dublin Core: http://dublincore.org/documents/2005/11/07/usageguide/

Hobbs, J. R., & Pan, F. (27 de Setembro de 2006). *Time Ontology in OWL*. Acesso em 30 de Julho de 2007, disponível em W3C Working Draft: http://www.w3.org/TR/2006/WD-owl-time-20060927/

Horridge, M., Knublauch, H., Rector, A., Stevens, R., & Wroe, C. (27 de Agosto de 2004). A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools. *The University Of Manchester* .

Horrocks, I., & Patel-Schneider, P. F. (2003). Reducing OWL entailment to description logic satisfiability. (D. Fensel, K. Sycara, & J. Mylopoulos, Eds.) *Proceendis of the 2003 International Semantic Web Conference (ISWC 2003)* , pp. 17-29.

Horrocks, I., Patel-Schneider, P. F., & Harmelen, F. v. (2003). From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics Vol.1 Numero.1* , pp. 7-26.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (21 de Maio de 2004). *W3C Member Submission.* Acesso em 05 de Junho de 2007, disponível em SWRL: A Semantic Web Rule Language Combining OWL and RuleML: http://www.w3.org/Submission/SWRL/

Hristidis, V., Koudas, N., & Papakonstantinou, Y. (2001). PREFER: A System for the Efficient Execution of Multi-parametric Ranked Queries. *Proceedings of the ACM SIGMOD International Conference on Management Data.* , pp. 259-270.

IMG. (2005). Acesso em Junho de 2005, disponível em Information Management Group Web Page: http://img.cs.man.ac.uk/

JADE. (Fevereiro de 2000). Acesso em 23 de Julho de 2007, disponível em JADE: Java Agent DEvelopment framework: http://jade.tilab.com/

Jena. (2007). Acesso em 5 de Junho de 2007, disponível em A Semantic Web Framework for Java: http://jena.sourceforge.net/

Joseki. (2007). Acesso em 20 de Julho de 2007, disponível em Joseki - A SPARQL Server for Jena: http://www.joseki.org/

Kalyanpur, A. (26 de Julho de 2006). Debugging and Repair of OWL Ontologies. *Phd dissertation* . University of Maryland, College Park.

Keeney, R. L., & Raiffa, H. (setembro de 1977). Decisions with Multiple Objectives: Preferences and Value Tradeoffs. *Journal of the American Statistical Association , 72, No. 359*, pp. 683-684.

Khatib, L., Morris, P. H., Morris, R. A., & Venable, K. B. (9-15 de Agosto de 2003). Tractable Pareto Optimization of Temporal Preferences. (G. Gottlob, & T. Walsh, Eds.) *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* , pp. 1289-1294.

Kießling, W. (20-23 de Agosto de 2002). Foundations of Preferences in Database Systems. *Proceedings of 28th International Conference on Very Large Data Bases (VLDB)* , pp. 311-322.

Kießling, W., & Köstler, G. (2002). Preference SQL - Design, Implementation, Experiences. *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)* , pp. 990-1001.

Kießling, W., Hafenrichter, B., Fischer, S., & Holland, S. (Setembro de 2001). Preference XPATH: A Query Language for E-Commerce. *Proceedings of the 5th Internationale Konferenz fr Wirtschaftsinformatik* , pp. 427-440.

Kim, J., Dou, D., Liu, H., & Kwak, D. (2007). Constructing A User Preference Ontology for Anti-Spam Mail Systems. *Proceedings of the 20th Canadian Conference on Artificial Intelligence (Canadian AI'07)* , pp. 272-283.

Kohncke, B., & Balke, W.-T. (dezembro de 2006). Personalized Digital Item Adaptation in Service-Oriented Environments. *First International Workshop on Semantic Media Adaptation and Personalization, SMAP '06* , pp. 91-96.

Köhncke, B., & Balke, W.-T. (2007). Preference-Driven Personalization for Flexible Digital Item Adaptation. *Multimedia Systems Journal (MMSJ) , 12(7)*.

Koutrika, G., & Ioannidis, Y. (2006). Personalization of Queries Based on User Preferences. In: G. Bosi, R. I. Brafman, J. Chomicki, & W. Kießling, *Preferences:*

Specification, Inference, Applications. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.

Lacroix, M., & Lavency, P. (1987). Preferences: Putting More Knowledge into Queries. *Proceedings of the 13th International Conference on Very Large Data Bases* , pp. 217-225.

Lamparter, S., Ankolekar, A., Studer, R., Oberle, D., & Weinhardt, C. (2006). A Policy Framework for Trading Configurable Goods and Services in Open Electronic Markets. *Proceedings of the 8th International Conference on Electronic Commerce: The new e-commerce - Innovations for Conquering - ICEC 06* , pp. 162-173.

Lassila, O., & Swick, R. R. (22 de Fevereiro de 1999). *Resource Description Framework (RDF) Model and Syntax Specification.* Acesso em Julho de 2007, disponível em W3C Recommendation 22 February 1999: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Lukasiewicz, T., & Schellhase, J. (2007). Variable-strength conditional preferences for ranking objects in ontologies. *Journal of Web Semantics, Volume 5* , pp. 180-194.

Manola, F., & Miller, E. (10 de Fevereiro de 2004). *RDF Primer.* Acesso em 18 de Julho de 2007, disponível em W3C Recommendation: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

Mantha, S. M. (Novembro de 1992). First-order preference theories and their applications. *Ph.D Thesis* .

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., et al. (22 de Novembro de 2004). *OWL-S: Semantic Markup for Web Services.* Acesso em 24 de Julho de 2007, disponível em W3C Member Submission: http://www.w3.org/Submission/OWL-S

McCarthy, P. (10 de Maio de 2005). *IBM developerWorks.* Acesso em 20 de Julho de 2007, disponível em Search RDF data with SPARQL: http://www-128.ibm.com/developerworks/java/library/j-sparql/

McGuinness, D. L., & Harmelen, F. v. (10 de Fevereiro de 2004). *OWL Web Ontology Language Overview*. Acesso em 19 de Julho de 2007, disponível em W3C Recommendation: http://www.w3.org/TR/2004/REC-owl-features-20040210/

McGuinness, D. L., Fikes, R., Rice, J., & Wilder, S. (12-15 de Abril de 2000). An Environment for Merging and Testing Large Ontologies. *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)* .

Motik, B. (Novembro de 2005). On the Properties of Metamodeling in OWL. (Springer, Ed.) *Proceedings of the 4th International Semantic Web Conference , 3729*, pp. 548-562.

Motro, A. (Julho de 1988). VAGUE: a user interface to relational databases that permits vague queries. *ACM Transactions on Information Systems (TOIS) , 6 , Issue 3*, pp. 187 - 214 .

Mylonas, P., & Wallace, M. (2006). Using ontologies and fuzzy relations in multimedia personalization. *First International Workshop on Semantic Media Adaptation and Personalization (SMAP'06)* , pp. 146-150.

Ngoc, K. A., Lee, Y.-K., & Lee, S.-Y. (4 de novembro de 2005). OWL-Based User Preference and Behavior Routine Ontology for Ubiquitous System. *The 4th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE'05)* , pp. 1615-1622.

Noy, N. F., & McGuinness, D. L. (Março de 2001). Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880* .

OpenLink Software. (2007). Acesso em 20 de Julho de 2007, disponível em Virtuoso SPARQL Query Demo: http://demo.openlinksw.com/sparql_demo/

Oztürk, M., Tsoukià, A., & Vincke, P. (2005). Preference Modelling. In: J. Figueira, S. Greco, & M. Ehrgott, *Multiple Criteria Decision Analysis: State of the Art Surveys* (pp. 27--72). Boston, Dordrecht, London: Springer Verlag.

Patel-Schneider, P. F., Hayes, P., & Horrocks, I. (10 de Fevereiro de 2004). *OWL Web Ontology Language Semantics and Abstract Syntax.* Acesso em 19 de Julho de 2007, disponível em W3C Recommendation: http://www.w3.org/TR/2004/REC-owl-semantics-20040210/

Perny, P., & Spanjaard, O. (2005). A preference-based approach to spanning trees and shortest paths problems. *European Journal of Operational Research , 162*, pp. 584-601.

Pothipruk, P., & Governatori, G. (20-22 de Novembro de 2005). A Formal Ontology Reasoning with Individual Optimization: A Realization of the Semantic Web. *Web Information Systems Engineering (WISE 2005)* , pp. 119-132.

Protege. (2007). Acesso em 20 de Julho de 2007, disponível em Protege: http://protege.stanford.edu/

Prud'hommeaux, E., & Seaborne, A. (14 de Junho de 2007). *SPARQL Query Language for RDF*. Acesso em 20 de Julho de 2007, disponível em W3C Candidate Recommendation: http://www.w3.org/TR/2007/CR-rdf-sparql-query-20070614/

Rossi, F., Venable, K. B., & Walsh, T. (Julho de 2004). mCP nets: representing and reasoning with preferences of multiple agents. *Proceedings of American Association for Artificial Intelligence (AAAI)* .

Ruttkay, Z. (26-29 de Junho de 1994). Fuzzy Constraint Satisfaction. *Proceedings of the Third IEEE Conference on Fuzzy Systems. IEEE World Congress on Computational Intelligence , 2*, pp. 1263-1268 .

Schiex, T. (1992). Possibilistic Constraint Satisfaction Problems or "How to handle soft constraints?". *Proceedings of the Eight International Conference on Uncertainty in Artificial Intelligence (UAI'92)* , pp. 268--275.

Seaborne, A., & Manjunath, G. (24 de Abril de 2007). *SPARQL/Update - A language for updating RDF graphs.* Acesso em 20 de Julho de 2007, disponível em SPARQL/Update - A language for updating RDF graphs: http://jena.hpl.hp.com/~afs/SPARQL-Update.html

Siberski, W., Pan, J. Z., & Thaden, U. (2006). Querying the Semantic Web with Preferences. *Proceedings of the 5th International SemanticWeb Conference (ISWC)* , pp. 612–624.

Smith, M. K., Welty, C., & McGuinness, D. L. (10 de Fevereiro de 2004). *OWL Web Ontology Language Guide.* Acesso em 19 de Julho de 2007, disponível em W3C Recommendation: http://www.w3.org/TR/2004/REC-owl-guide-20040210/

Tsinaraki, C., & Christodoulakis, S. (4-6 de Janeiro de 2006). A Multimedia User Preference Model that Supports Semantics and its Application to MPEG 7/21. *12th International Conference on Multi Media Modeling (MMM 2006)* , pp. 121-128.

Tsinaraki, C., & Christodoulakis, S. (2006). A User Preference Model and a Query Language that allow Semantic Retrieval and Filtering of Multimedia Content. *First International Workshop on Semantic Media Adaptation and Personalization (SMAP'06)* , pp. 121-128.

Web Clipboard. (2007). Acesso em 20 de Julho de 2007, disponível em SPARQL Demo: http://www.sparqlets.org/clipboard/demo

Wikipedia, c. (15 de Julho de 2007). *Ontology*. Acesso em 18 de Julho de 2007, disponível em Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/w/index.php?title=Ontology&oldid=144886694

Wikipedia, c. (2007). *Preference.* (T. F. Wikipedia, Ed.) Acesso em 13 de julho de 2007, disponível em Preference: http://en.wikipedia.org/w/index.php?title=Preference&oldid=138165745

APÊNDICE I: Ontologia de Preferências OWLPREF em RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns="http://www.unifor.br/mia/owlpref.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.unifor.br/mia/owlpref.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Operator">
   <rdfs:label xml:lang="pt">Operador</rdfs:label>
   <rdfs:comment xml:lang="pt">representação de operadores para propriedades de tipo de dados
OWL</rdfs:comment>
   <rdfs:comment xml:lang="en">representation of operators for owl datatype
properties</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="DatatypeCondition">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   ></rdfs:comment>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:allValuesFrom rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
     <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="hasOWLProperty"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Condition"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="OrderPreference">
   <rdfs:label xml:lang="pt">Preferência Ordenada</rdfs:label>
   <rdfs:subClassOf>
    <owl:Class rdf:ID="CompositePreference"/>
   </rdfs:subClassOf>
   <rdfs:comment xml:lang="pt">Representação de Preferência Ordenada de OWLPref. - Representa
ordem entre preferências OWLPref
Ex: Preferência 1 &gt; Preferência 2 &gt; Preferência 3</rdfs:comment>
   <rdfs:comment xml:lang="en">Order Preference representation of OWLPref. - Represents partial
order
   between the OWLPref preferences.
Preference 1 &gt; Preference 2 &gt; Preference 3</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="ClassPreference">
```

```
    <rdfs:subClassOf>
      <owl:Class rdf:ID="SimplePreference"/>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="pt">Preferência de Classes</rdfs:label>
    <rdfs:comment xml:lang="pt">Representação da Preferência de Classes de
OWLPref</rdfs:comment>
    <rdfs:comment xml:lang="en">Class Preference representation of OWLPref</rdfs:comment>
    <owl:disjointWith>
      <owl:Class rdf:ID="AttributePreference"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="MaximumPreference">
    <rdfs:comment xml:lang="pt">Representação de Preferência Máxima de OWLPref.
- Valores extremos máximos da propriedade específicada são preferríveis</rdfs:comment>
    <rdfs:comment xml:lang="en">Maximum Preference representation of OWLPref.
- Maximum extremal value of the specific property is preferred.</rdfs:comment>
    <rdfs:label xml:lang="pt">Preferência Máxima</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="ExtremalPreference"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="PositiveDatatypePreference">
    <rdfs:comment xml:lang="pt">Representação de Preferência Positiva de Tipo de Dados de
OWLPref.
- Preferência Positiva de uma propriedade de tipo de dados.
Ex: prefiro Carros com Cor =  'Azul'
    prefiro carros com Preco com Valor &gt; 50</rdfs:comment>
    <rdfs:comment xml:lang="en">Positive Datatype Preference representation of OWLPref.
- Positive preference about a datatype property.
Ex: prefer Cars with Color =  'Blue'
    prefer cars with Price with Value &gt; 50</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="DatatypePreference"/>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="pt">Preferência Positiva de Tipo de Dados</rdfs:label>
    <owl:disjointWith>
      <owl:Class rdf:ID="NegativeDatatypePreference"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#ExtremalPreference"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#NegativeDatatypePreference">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#DatatypePreference"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#PositiveDatatypePreference"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#ExtremalPreference"/>
    </owl:disjointWith>
    <rdfs:comment xml:lang="pt">Representação de Preferência Negativa de Tipo de Dados de
OWLPref.
 Qualquer valor nao contido no conjunto especificado são preferríveis.
Apenas sobre propriedades de tipo de dados.
```

```
Ex: Eu nao prefiro Carros com Cor =  'Azul'
   Eu nao prefiro Preco com Valor &gt; 50</rdfs:comment>
    <rdfs:comment xml:lang="en">Negative Datatype Preference representation of OWLPref.
 Any value not contained in the negative set is preferred.
Its about a datatype property.
Ex: I not  prefer Cars with Color =  'Blue'
    I not prefer Price with Value &gt; 50</rdfs:comment>
    <rdfs:label xml:lang="pt">Preferência Negativa de Tipo de Dados</rdfs:label>
  </owl:Class>
  <owl:Class rdf:about="#DatatypePreference">
    <owl:disjointWith>
      <owl:Class rdf:ID="ObjectPreference"/>
    </owl:disjointWith>
    <rdfs:comment xml:lang="pt">Representação de Preferência de Tipo de Dados de
OWLPref</rdfs:comment>
    <rdfs:comment xml:lang="en">Datatype Preference representation of OWLPref</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#hasOWLProperty"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#AttributePreference"/>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="pt">Preferência de Tipo de Dados</rdfs:label>
  </owl:Class>
  <owl:Class rdf:about="#AttributePreference">
    <owl:disjointWith rdf:resource="#ClassPreference"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#SimplePreference"/>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="pt">Preferência de Atributos</rdfs:label>
    <rdfs:comment xml:lang="pt">Representação de Preferência de Atributos de OWLPref
</rdfs:comment>
    <rdfs:comment xml:lang="en">Representation of Attribute Preference of OWLPref
</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#ObjectPreference">
    <owl:disjointWith rdf:resource="#DatatypePreference"/>
    <rdfs:comment xml:lang="pt">Representação de Preferência de Objetos de
OWLPref</rdfs:comment>
    <rdfs:comment xml:lang="en">Object Preference representation of OWLPref</rdfs:comment>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Preferência de Objetos</rdfs:label>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#hasOWLProperty"/>
        </owl:onProperty>
      </owl:Restriction>
```

```
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#AttributePreference"/>
  </owl:Class>
  <owl:Class rdf:ID="NegativeObjectPreference">
    <rdfs:comment xml:lang="pt">Preferência Negativa de Objetos de OWLPref
- Preferência Negativa sobre propriedade de objetos.
Qualquer valor que nao faca parte do conjunto negativo é preferido.
Ex: Eu nao prefiro Carros com Cor.Azul</rdfs:comment>
    <rdfs:comment xml:lang="en">Negative Object Preference representation of OWLPref.
- Negative preference about a object property.
Any value not contained in the negative set is preferred.
Ex: I not  prefer Cars with Color.Blue'</rdfs:comment>
    <rdfs:label xml:lang="pt">Preferência Negativa de Objetos</rdfs:label>
    <rdfs:subClassOf rdf:resource="#ObjectPreference"/>
    <owl:disjointWith>
      <owl:Class rdf:ID="PositiveObjectPreference"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="ParetoPreference">
    <rdfs:comment xml:lang="pt">Representação de Preferência de Pareto de OWLPref.
- Preferência Acumulada onde todas as preferências tem a mesma prioridade ou são igualmente importantes.</rdfs:comment>
    <rdfs:comment xml:lang="en">Pareto Preference representation of OWLPref.
-Cumulative Preference. All preferences have the same priority / are equally important</rdfs:comment>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Preferência de Pareto</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompositePreference"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Condition">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Represents a OWLPref Condition</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="Preference">
    <rdfs:label xml:lang="pt">Preferência</rdfs:label>
    <rdfs:comment xml:lang="en">a generic representation of a OWLPref preference</rdfs:comment>
    <rdfs:comment xml:lang="pt">Representacao generica de uma preferência OWLPref</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="List">
    <rdfs:comment xml:lang="pt">Representacao do conceito de Lista para uso de listas de OWLPref.
A intenção é fornecer uma versão compatival com OWL-DL de  rdf:List</rdfs:comment>
    <rdfs:comment xml:lang="en">A representation of List to OWLPref lists.
This is intended to provide an OWL-DL compatible version of rdf:List</rdfs:comment>
    <rdfs:label xml:lang="pt">Lista</rdfs:label>
  </owl:Class>

  <owl:Class rdf:ID="ClassList">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#ClassList"/>
        <owl:onProperty>
```

```
      <owl:FunctionalProperty rdf:about="#tail"/>
     </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#head"/>
     </owl:onProperty>
     <owl:allValuesFrom rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#List"/>
  <rdfs:label xml:lang="pt">Lista de Classes</rdfs:label>
  <rdfs:comment xml:lang="pt">Lista OWLPref de classes owl</rdfs:comment>
  <rdfs:comment xml:lang="en">OWLPref list of owl classes</rdfs:comment>
  <owl:disjointWith>
    <owl:Class rdf:ID="AttributeList"/>
  </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:ID="ObjectCondition">
  <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#hasOWLProperty"/>
     </owl:onProperty>
     <owl:allValuesFrom rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Condition"/>
 </owl:Class>
 <owl:Class rdf:ID="IntervalPreference">
  <rdfs:subClassOf rdf:resource="#ParetoPreference"/>
  <owl:equivalentClass>
    <owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasPreferenceSet"/>
       </owl:onProperty>
       <owl:someValuesFrom>
        <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
           <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#hasOperator"/>
           </owl:onProperty>
           <owl:hasValue rdf:resource="#greaterThan"/>
          </owl:Restriction>
          <owl:Class rdf:about="#PositiveDatatypePreference"/>
         </owl:intersectionOf>
        </owl:Class>
       </owl:someValuesFrom>
      </owl:Restriction>
```

```
      <owl:Restriction>
       <owl:someValuesFrom>
        <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
           <owl:onProperty>
            <owl:FunctionalProperty rdf:about="#hasOperator"/>
           </owl:onProperty>
           <owl:hasValue rdf:resource="#lessThan"/>
          </owl:Restriction>
          <owl:Class rdf:about="#PositiveDatatypePreference"/>
         </owl:intersectionOf>
        </owl:Class>
       </owl:someValuesFrom>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasPreferenceSet"/>
       </owl:onProperty>
      </owl:Restriction>
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasPreferenceSet"/>
       </owl:onProperty>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >2</owl:cardinality>
      </owl:Restriction>
     </owl:intersectionOf>
    </owl:Class>
   </owl:equivalentClass>
   <rdfs:comment xml:lang="pt">Preferência por Intervalos</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#AttributeList">
   <rdfs:comment xml:lang="pt">lista OWLPref de propriedades OWL</rdfs:comment>
   <rdfs:comment xml:lang="en">OWLPref list of owl properties</rdfs:comment>
   <rdfs:label xml:lang="pt">Lista de Atributos</rdfs:label>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#tail"/>
     </owl:onProperty>
     <owl:allValuesFrom rdf:resource="#AttributeList"/>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:allValuesFrom>
      <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#ObjectProperty"/>
       </owl:unionOf>
      </owl:Class>
     </owl:allValuesFrom>
     <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#head"/>
```

```
      </owl:onProperty>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#List"/>
    <owl:disjointWith rdf:resource="#ClassList"/>
  </owl:Class>
  <owl:Class rdf:about="#CompositePreference">
    <rdfs:subClassOf rdf:resource="#Preference"/>
    <owl:disjointWith>
     <owl:Class rdf:ID="ConditionalPreference"/>
    </owl:disjointWith>
    <owl:disjointWith>
     <owl:Class rdf:about="#SimplePreference"/>
    </owl:disjointWith>
    <rdfs:comment xml:lang="pt">Preferência Composta é uma combinação de preferências OWLPref
</rdfs:comment>
    <rdfs:comment xml:lang="en">Composite Preference is a combination of OWLPref preferences
</rdfs:comment>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Preferência Composta</rdfs:label>
  </owl:Class>
  <owl:Class rdf:about="#ExtremalPreference">
    <rdfs:comment xml:lang="pt">Representação de Preferência Extrema de OWLPref
</rdfs:comment>
    <rdfs:comment xml:lang="en">Extremal Preference representation of OWLPref</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#DatatypePreference"/>
    <rdfs:label xml:lang="pt">Preferência Extrema</rdfs:label>
    <owl:disjointWith rdf:resource="#NegativeDatatypePreference"/>
    <owl:disjointWith rdf:resource="#PositiveDatatypePreference"/>
  </owl:Class>
  <owl:Class rdf:about="#MinimumPreference">
    <rdfs:comment xml:lang="pt">Representação de Preferência Mínima de OWLPref.
- Valores extremos mínimos da propriedade específicada são preferríveis</rdfs:comment>
    <rdfs:comment xml:lang="en">Minimum Preference representation of OWLPref.
- Minimum extremal value of the specific property is preferred.</rdfs:comment>
    <rdfs:label xml:lang="pt">Preferência Mínima</rdfs:label>
    <rdfs:subClassOf rdf:resource="#ExtremalPreference"/>
  </owl:Class>
  <owl:Class rdf:about="#ConditionalPreference">
    <rdfs:comment xml:lang="pt">Representação de uma Preferência Condicional OWLPref
</rdfs:comment>
    <rdfs:comment xml:lang="en">Representation of a conditional OWLPref Preference
</rdfs:comment>
    <rdfs:label xml:lang="pt">Preferência Condicional</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Preference"/>
    <owl:disjointWith>
     <owl:Class rdf:about="#SimplePreference"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#CompositePreference"/>
  </owl:Class>
  <owl:Class rdf:about="#SimplePreference">
    <owl:disjointWith rdf:resource="#ConditionalPreference"/>
    <owl:disjointWith rdf:resource="#CompositePreference"/>
    <rdfs:subClassOf rdf:resource="#Preference"/>
```

```
   <rdfs:comment xml:lang="pt">representação genérica de uma Preferência Simples de
OWLPref</rdfs:comment>
   <rdfs:comment xml:lang="en">a generic representation of a SimpleOWLPref preference
</rdfs:comment>
   <rdfs:label xml:lang="pt">Preferência Simples</rdfs:label>
  </owl:Class>
  <owl:Class rdf:about="#PositiveObjectPreference">
   <owl:disjointWith rdf:resource="#NegativeObjectPreference"/>
   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Preferência Positiva de Objetos</rdfs:label>
   <rdfs:subClassOf rdf:resource="#ObjectPreference"/>
   <rdfs:comment xml:lang="pt">Representacao de Preferência Positiva de Objetos de OWLPref.
É uma preferência positiva sobre propriedades de objetos.
Qualquer valor no conjunto positive é preferrível ao resto.
Ex: Eu prefiro Carros com Cor.Azul</rdfs:comment>
   <rdfs:comment xml:lang="en">Positive Object Preference representation of OWLPref.
- Positive preference about a object property.
 Any value in the positive set is rated better than the rest.
Ex: I  prefer Cars with Color.Blue'</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#PreferenceList">
   <rdfs:comment xml:lang="pt">Lista de preferências OWLPref</rdfs:comment>
   <rdfs:comment xml:lang="en">OWLPref list of OWLPref preferences</rdfs:comment>
   <rdfs:label xml:lang="pt">Lista de Preferências</rdfs:label>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:allValuesFrom rdf:resource="#PreferenceList"/>
     <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#tail"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:allValuesFrom rdf:resource="#Preference"/>
     <owl:onProperty>
      <owl:FunctionalProperty rdf:about="#head"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="#List"/>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#hasPreferenceSet">
   <rdfs:domain rdf:resource="#ParetoPreference"/>
   <rdfs:range rdf:resource="#Preference"/>
   <rdfs:label xml:lang="pt">temConjuntoDePreferências</rdfs:label>
   <rdfs:comment xml:lang="pt">conjunto de preferências OWLPref</rdfs:comment>
   <rdfs:comment xml:lang="en">set of OWLPref preferences</rdfs:comment>
  </owl:ObjectProperty>
  <owl:FunctionalProperty rdf:about="#tail">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
   <rdfs:range rdf:resource="#List"/>
   <rdfs:domain rdf:resource="#List"/>
  </owl:FunctionalProperty>
```

```
  <owl:FunctionalProperty rdf:ID="hasDatatypePropertyValue">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#PositiveDatatypePreference"/>
          <owl:Class rdf:about="#NegativeDatatypePreference"/>
          <owl:Class rdf:about="#DatatypeCondition"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:comment xml:lang="pt">valor preferido de uma propriedade OWL de tipo de dados
</rdfs:comment>
    <rdfs:comment xml:lang="en">value that is preferred of a datatype owl property</rdfs:comment>
    <rdfs:label xml:lang="pt">temValorDePropriedadeDeTipoDeDados</rdfs:label>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="hasObjectPropertyValue">
    <rdfs:label xml:lang="pt">temValorDePropriedadeDeObjetos</rdfs:label>
    <rdfs:comment xml:lang="pt">valor preferido de uma propriedade owl de objetos</rdfs:comment>
    <rdfs:comment xml:lang="en">value that is preferred of a object owl property</rdfs:comment>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#PositiveObjectPreference"/>
          <owl:Class rdf:about="#NegativeObjectPreference"/>
          <owl:Class rdf:about="#ObjectCondition"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#head">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#List"/>
    <rdfs:label xml:lang="pt">primeiro</rdfs:label>
    <rdfs:comment xml:lang="pt">primeiro elemento</rdfs:comment>
    <rdfs:comment xml:lang="en">first element</rdfs:comment>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="hasPreference">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#ConditionalPreference"/>
    <rdfs:range rdf:resource="#Preference"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#hasPreferenceList">
    <rdfs:label xml:lang="pt">temListaDePreferências</rdfs:label>
    <rdfs:comment xml:lang="pt">lista de preferências OWLPref</rdfs:comment>
    <rdfs:comment xml:lang="en">list of OWLPref preferences</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#OrderPreference"/>
    <rdfs:range rdf:resource="#List"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="hasOWLClass">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:comment xml:lang="pt">classes de OWL</rdfs:comment>
```

```
    <rdfs:comment xml:lang="en">classes of OWL</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:label xml:lang="pt">temClasseOWL</rdfs:label>
    <rdfs:domain>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#ClassPreference"/>
       <owl:Class rdf:about="#Condition"/>
      </owl:unionOf>
     </owl:Class>
    </rdfs:domain>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#hasOperator">
    <rdfs:comment xml:lang="pt">operadores para propriedades de tipo de dados OWL
</rdfs:comment>
    <rdfs:comment xml:lang="en">operators for datatype properties OWL</rdfs:comment>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >temOperador</rdfs:label>
    <rdfs:range rdf:resource="#Operator"/>
    <rdfs:domain>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#PositiveDatatypePreference"/>
       <owl:Class rdf:about="#NegativeDatatypePreference"/>
       <owl:Class rdf:about="#DatatypeCondition"/>
      </owl:unionOf>
     </owl:Class>
    </rdfs:domain>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#hasOWLProperty">
    <rdfs:domain>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#AttributePreference"/>
       <owl:Class rdf:about="#Condition"/>
      </owl:unionOf>
     </owl:Class>
    </rdfs:domain>
    <rdfs:range>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
       <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
       <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#ObjectProperty"/>
      </owl:unionOf>
     </owl:Class>
    </rdfs:range>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:label xml:lang="pt">temPropriedadeOWL</rdfs:label>
    <rdfs:comment xml:lang="pt">propriedades de OWL: ObjectProperty ou
DatatypeProperty</rdfs:comment>
    <rdfs:comment xml:lang="en">properties of OWL:ObjectProperty or DatatypeProperty
</rdfs:comment>
  </owl:FunctionalProperty>
```

```
  <owl:FunctionalProperty rdf:ID="onCondition">
    <rdfs:domain rdf:resource="#ConditionalPreference"/>
    <rdfs:range rdf:resource="#Condition"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:FunctionalProperty>
  <Operator rdf:ID="greatherThanOrEqual">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Maior Ou Igual Que</rdfs:comment>
  </Operator>
  <Operator rdf:ID="equal">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Igual Que</rdfs:comment>
  </Operator>
  <Operator rdf:ID="lessThanOrEqual">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >menor Ou Igual Que</rdfs:comment>
  </Operator>
<rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Class">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.unifor.br/mia/owlpref.owl#greaterThan">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
 </rdf:Description>
 <rdf:Description rdf:about="http://www.unifor.br/mia/owlpref.owl#lessThan">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</rdf:Description>
 <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#ObjectProperty">
            <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
 </rdf:Description>
<rdf:Description rdf:about="http://www.w3.org/2002/07/owl#DatatypeProperty">
           <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3, Build 418)  http://protege.stanford.edu -->
```

APÊNDICE II: Exemplos do Mapeamento de OWLPREF para SPARQL *Preference*

- Preferência Positiva de Tipo de Dados: *Prefiro discos rígidos com modelo SATA.*

```
<PositiveDatatypePreference rdf:ID= “Pref_HD_SATA”>
      <hasOWLProperty rdf:resource= “&pc;model” />
      <hasOperator rdf:resource=” #equal” />
       <hasDatatypePropertyValue rdf:datatype="&xsd;string”> SATA
      </hasDatatypePropertyValue>
      <hasContext rdf:resource= “&pc;HardDisk”/>
</PositiveDatatypePreference>
```

```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
{    ?id pc:model ?model .
      ?id rdf:type pc:HardDisk    }
PREFERRING ( ?model = ‘SATA’)
```

- Preferência Negativa de Tipo de Dados: *Prefiro disco rígidos que não sejam do modelo IDE.*

```
<NegativeDatatypePreference rdf:ID= “Pref_HD_N_IDE”>
      <hasOWLProperty rdf:resource= “&pc;model” />
      <hasOperator rdf:resource=” #equal” />
      <hasDatatypePropertyValue rdf:datatype="&xsd;string”> IDE
      </hasDatatypePropertyValue>
      <hasContext rdf:resource= “&pc;HardDisk”/>
</NegativeDatatypePreference>
```

```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
{    ?id pc:model ?model .
      ?id rdf:type pc:HardDisk    }
PREFERRING ( ?model != ‘IDE’)
```

- Preferência Positiva de Objetos: *Prefiro discos rígidos que sejam compatíveis com notebooks.*

```
<PositiveObjectPreference rdf:ID= “Pref_HD_SATA”>
      <hasOWLProperty rdf:resource= “&pc;hasCompatibility” />
      <hasObjectPropertyValue rdf:resource="&pc;Notebook”/>
      <hasContext rdf:resource= “&pc;HardDisk”/>
</PositiveObjectPreference>
```

```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
 {    ?id pc:hasCompability ?compability .
       ?id rdf:type pc:HardDisk    }
 PREFERRING ( ?compability = pc:Notebook)
```

- Preferência Máxima: *Prefiro discos rígidos com maior capacidade de armazenamento.*

```
<MaximumPreference rdf:ID= “Pref_HD_MAX_STO”>
      <hasOWLProperty rdf:resource= “&pc;storageCapacity” />
      <hasContext rdf:resource= “&pc;HardDisk”/>
</ MaximumPreference >
```

```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
 {    ?id pc:storageCapacity ?sto .
       ?id rdf:type pc:HardDisk    }
 PREFERRING HIGHEST(?sto)
```

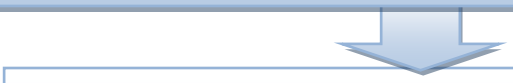
- Preferência Mínima: *Prefiro discos rígidos com menor preço.*

```
<MinimumPreference rdf:ID= “Pref_HD_MIN_STO”>
      <hasOWLProperty rdf:resource= “&pc;price” />
      <hasContext rdf:resource= “&pc;HardDisk”/>
</ MinimumPreference >
```

```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
 {    ?id pc:price ?price .
       ?id rdf:type pc:HardDisk    }
 PREFERRING LOWEST(?price)
```

- Preferência Ordenada: *Prefiro discos rígidos com maior capacidade prioritariamente a discos rígidos com menor preço.*

```
<OrderPreference rdf:ID="Preferencia_Ordenada">
    <hasPreferenceList>
        <PreferenceList rdf:ID="List_1">
            <head rdf:resource="#Pref_HD_MAX_STO "/>
            <tail> <PreferenceList rdf:ID="List_2”>
                <head rdf:resource="#Pref_HD_MIN_STO"/>
                 <tail rdf:resource="#nil"/>
                </PreferenceList> </tail>
        </PreferenceList>
   </hasPreferenceList>
   <hasContext rdf:resource= “#HardDisk”/>
</OrderPreference>
```



```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
 {    ?id pc:price ?price .
      ?id pc:storageCapacity ?sto .
       ?id rdf:type pc:HardDisk    }
 PREFERRING HIGHEST(?sto) CASCADE LOWEST(?price)
```

- Preferência de Pareto: *Prefiro discos rígidos com maior capacidade tão quanto os discos rígidos com menor preço.*

```
<ParetoPreference rdf:ID="Preferencia_Pareto">
    <hasPreferenceSet rdf:resource="#Pref_HD_MAX_STO"/>
    <hasPreferenceSet rdf:resource="##Pref_HD_MIN_STO"/>
   <hasContext rdf:resource= “#HardDisk”/>
</ParetoPreference>
```

```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
 {    ?id pc:price ?price .
      ?id pc:storageCapacity ?sto .
       ?id rdf:type pc:HardDisk    }
 PREFERRING HIGHEST(?sto) AND LOWEST(?price)
```

- Preferência por Intervalos: *Prefiro discos rígidos que custem entre 100 e 300.*

```
<ParetoPreference rdf:ID="Preferencia_Intervalos100e300">
    <hasPreferenceSet>
        <PositiveDatatypePreference rdf:ID=”PrecoMaior100”>
            <hasOWLProperty rdf:resource= “&pc;price” />
            <hasOperator rdf:resource=” #greaterThan” />
            <hasDatatypePropertyValue rdf:datatype="&xsd;int”>100
            </hasDatatypePropertyValue>
        </PositiveDatatypePreference>
    </hasPreferenceSet>
    <hasPreferenceSet>
        <PositiveDatatypePreference rdf:ID=”PrecoMenor300”>
            <hasOWLProperty rdf:resource= “&pc;price” />
            <hasOperator rdf:resource=” #lessThan” />
            <hasDatatypePropertyValue rdf:datatype="&xsd;int”>300
            </hasDatatypePropertyValue>
        </PositiveDatatypePreference>
    </hasPreferenceSet>
    <hasContext rdf:resource= “&pc;HardDisk”/>
</ParetoPreference>
```

```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
 {    ?id pc:price ?price .
      ?id rdf:type pc:HardDisk     }
 PREFERRING (?price>100) AND (?price<300)
```

- Preferência Condicional: *Se localização do fornecedor for Brazil, então prefiro discos rígidos que custem entre 100 e 300.*

```
<ConditionalPreference rdf:ID="ConditionalPreference_1">
      <hasPreference rdf:resource="#Preferencia_Intervalos100e300"/>
      <onCondition rdf:resource="#ObjectCondition_1"/>
      <hasContext rdf:resource="&pc;HardDisk"/>
   </ConditionalPreference>
   <ObjectCondition rdf:ID="ObjectCondition_1">
      <hasObjectPropertyValue rdf:resource="&p2;BRAZIL"/>
      <hasOWLClass rdf:resource="&pc;HardDisk"/>
      <hasOWLProperty rdf:resource="&pc;hasSupplierLocation"/>
   </ObjectCondition>
```

```
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  pc:    <http://mia.unifor.br/computer.owl#>
SELECT  *
WHERE
 {    ?id pc:price ?price .
      ?id rdf:type pc:HardDisk     .
      ?id pc:hasSupplierLocation ?location .
FILTER (?location= ‘BRAZIL’) }
 PREFERRING (?price>100) AND (?price<300)
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo