ORION – Um Framework para Refinamento de Ontologias Através de Técnicas de Revisão de Teorias

Felipe Leão, Kate Revoredo, Fernanda Baião

NP2Tec – Núcleo de Pesquisa e Prática em Tecnologia
Departamento de Informática Aplicada
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Av. Pasteur, 458 – Rio de Janeiro – RJ – Brazil

{felipe.leao, katerevoredo, fernanda.baiao}@uniriotec.br

Abstract. *Ontology have been frequently used for knowledge representation. As it evolves, however, keeping its consistency becomes a costly and complex task, if no support is provided. This work proposes ORION – Ontology Refinement through theory revisION, which applies existing Theory Revision techniques to support Ontology evolution. Preliminary tests were successful in obtaining a more accurate refined Ontology.*

Resumo. *A utilização de Ontologias para representação do conhecimento sobre um domínio requer um trabalho contínuo de manutenção, uma vez que a observação de novas instâncias pode trazer inconsistências às regras previamente definidas. No entanto, a modificação manual da Ontologia pode ser muito custosa e complexa. Este trabalho propõe o ORION, um framework para refinamento automático de Ontologias através do uso de técnicas de revisão de teorias. Testes preliminares mostraram a viabilidade da proposta e a obtenção de uma Ontologia refinada mais precisa.*

1.Introdução

Ontologias propiciam descrições concisas e desambíguas sobre conceitos e relacionamentos em um domínio de interesse. A utilidade de uma ontologia depende da precisão com a qual a estrutura do domínio, descrita pela ontologia, representa as instâncias deste domínio. Quando novas instâncias válidas são recebidas, as regras previamente especificadas na ontologia podem deixar de ser verdade. Quando isto acontece, assumindo-se que as novas instâncias sejam observações inequívocas da realidade, estas regras devem ser revisadas. Em cenários reais, onde o número de instâncias e regras podem se tornar muito grande (como na Web Semântica), detectar e corrigir manualmente estas inconsistências se torna impraticável. É interessante então ter um mecanismo para automaticamente refinar uma ontologia.

Por outro lado, a área de Programação em Lógica Indutiva (ILP) [Muggleton, 1992] utiliza técnicas de revisão de teoria [Wrobel, 1996] para aprimorar uma base de conhecimento de forma automática, visando reduzir os erros, a dificuldade e o tempo despendido na aquisição de conhecimento. Um exemplo de sistema de revisão de teorias é o FORTE (*First Order Revision of Theories from Examples*) [Richards e Mooney, 1995], que revisa bases de conhecimento descritas em cláusulas de primeira ordem Horn e livre de funções (i.e., programas em PROLOG livres de funções).

Motivados então pela possibilidade de um refinamento automático de ontologias, em [Leão et al., 2010] nós propomos utilizar o sistema FORTE para refinar

os axiomas de uma ontologia exemplo, descritos em cláusulas de primeira ordem Horn. Entretanto, uma ontologia pode estar representada em outras linguagens, como OWL, RDF, OBO, lógicas de descrição, etc. Dessa forma, neste artigo, nós propomos um framework para refinamento de ontologias. Neste framework uma ontologia (descrita através de linguagens como OWL, RDF, OBO etc.) é mapeada para uma teoria a ser revisada pelo FORTE. A teoria revisada é então convertida novamente para uma ontologia refinada representada na sua linguagem original.

O trabalho está organizado da seguinte forma. A seção 2 apresenta as definições de ontologia, técnicas de revisão de teoria e cita a aplicabilidade de técnicas de revisão de teoria no refinamento de ontologias. Na seção 3 é proposto o framework ORION. A seção 4 aborda trabalhos relacionados. Finalmente, a seção 5 conclui o trabalho.

2. Conhecimento Preliminar

2.1. Ontologia

Uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada [Gruber, 1995]. Ela define um vocabulário comum entre aqueles que compartilham informações de um mesmo domínio. Formalmente, uma ontologia é definida através dos conjuntos finitos $\mathcal{C}$, $\mathfrak{R}$ e $\mathfrak{I}$, onde $\mathcal{C}$ é um conjunto de conceitos, $\mathfrak{R}$ é um conjunto de relacionamentos binários, que relacionam dois conceitos de $\mathcal{C}$, sendo $\mathcal{C} \cap \mathfrak{R} = \emptyset$. $\mathfrak{I}$ é um conjunto de expressões em lógica de primeira ordem, descrevendo dependências entre elementos de $\mathfrak{R}$.

Em um domínio que represente uma família podemos dizer, por exemplo, que Professor e Pessoa são dois conceitos, relacionados entre si com a semântica de especialização ("É UM") evidenciando, dentre outras coisas, que uma instância do conceito Professor é também uma instância do conceito Pessoa. A expressão (regra) **pessoa(x) ← professor(x)** descreve a semântica do relacionamento "É UM" entre as instâncias dos conceitos Pessoa e Professor, pertencendo ao conjunto $\mathfrak{I}$.

Uma ontologia pode ser representada como um grafo direcionado G(V,E), onde V é um conjunto finito de vértices, e E um conjunto finito de arestas . Cada vértice em V é rotulado por um conceito de $\mathcal{C}$ e cada aresta em E é rotulada por um relacionamento em $\mathfrak{R}$. Alternativamente, uma ontologia pode ser representada através de modelos baseados em RDF [Klyne e Carroll, 2004] como na Figura 1, onde duas instâncias de Pessoa são descritas.

```
1  <?xml version="1.0"?>
2  <rdf:RDF
3  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4  xmlns:fam="http://meu.site.com.br/RDF/universidade-ns#">
5
6      <rdf:Description rdf:about="http://meu.site.com.br/FelipeLeao">
7          <fam:primeironome>Felipe</fam:primeironome>
8          <fam:ultimonome>Leão</fam:ultimonome>
9      </rdf:Description>
10
11     <rdf:Description rdf:about="http://meu.site.com.br/KateRevoredo">
12         <fam:primeironome>Kate</fam:primeironome>
13         <fam:ultimonome>Revoredo</fam:ultimonome>
14     </rdf:Description>
15
16 </rdf:RDF>
```

Figura 1 - Instâncias de Pessoa representadas em sintaxe RDF

Entretanto, parte da ontologia, neste caso o conjunto $\Im$, não pode ser representada em RDF, dado que esta linguagem não provê mecanismos para especificar ou descrever a semântica dos relacionamentos entre os conceitos. Para superar estas deficiências, o RDF foi estendido para outros modelos como o RDF Schema[1] e OWL[2]. A Figura 2 ilustra a definição dos conceitos Pessoa, Professor e Aluno e os relaciona através da combinação dos modelos RDF Schema e OWL.

```
1  <?xml version="1.0"?>
2
3  <rdf:RDF
4  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6  xmlns:owl="http://www.w3.org/2002/07/owl#"
7  xml:base="http://meu.site.com.br/universidade#">
8
9      <rdfs:Class rdf:ID="Pessoa" />
10
11     <rdfs:Class rdf:ID="Professor">
12         <rdfs:subClassOf rdf:resource="#Pessoa"/>
13         <owl:disjointWith rdf:resource="#Aluno"/>
14     </rdfs:Class>
15
16     <rdfs:Class rdf:ID="Aluno">
17         <rdfs:subClassOf rdf:resource="#Pessoa"/>
18     </rdfs:Class>
19
20 </rdf:RDF>
```

Figura 2 - Definição e relacionamento entre conceitos expressos pelas sintaxes RDF Schema e OWL

2.2. Revisão de Teoria

A aquisição de conhecimento em ILP é uma tarefa que consome tempo e suscetível a erros. Sistemas de revisão de teoria visam aperfeiçoar, de forma automática, bases de conhecimento, através de métodos de aprendizado de máquina [Wrobel, 1996].

A Figura 3 apresenta um esquema genérico para revisão de teoria. O sistema de revisão de teoria recebe uma teoria inicial e um conjunto consistente de exemplos. A teoria inicial é composta por um componente invariante, denominada *conhecimento preliminar* (CP) e uma componente que pode ser modificada (A). O conjunto de exemplos é dividido em exemplos positivos (E^+) e exemplos negativos (E^-). Os exemplos positivos são afirmações corretas, enquanto exemplos negativos são afirmações incorretas sobre o domínio.

Figura 3 - Esquema para Revisão de Teoria

O Sistema de Revisão de Teorias deverá gerar uma teoria final "minimamente revista" capaz de provar todos os exemplos positivos em E^+ e refutar todos os exemplos negativos em E^-. Desta forma, a teoria final torna-se consistente com a base de dados. Um exemplo de sistema de revisão de teorias é o FORTE [Richards e Mooney, 1995].

[1] http://www.w3.org/TR/rdf-schema/
[2] http://www.w3.org/TR/owl2-overview/

A qualidade das revisões produzidas depende crucialmente de quais cláusulas da teoria são escolhidas para serem modificadas e qual a modificação aplicada, definindo então um conjunto de **pontos de revisão**. Após especificar onde a teoria deve ser modificada, é preciso determinar como modificá-la, definindo então **operadores de revisão**. Exemplos de operadores de revisão são: exclusão de cláusula, adição de antecedente, exclusão de antecedente e adição de cláusula. Uma função de avaliação é utilizada para escolher a melhor proposta de revisão a ser implementada. Um possível algoritmo de revisão que utiliza uma abordagem gulosa de subida de encosta (*greedy hill-climbing*) é exibido na Figura 4.

Algorithm 1 Algoritmo de revisão de teorias de primeira-ordem

```
repita
   gerar pontos de revisão;
   para cada ponto de revisão
      gerar possíveis revisões;
      avaliar possíveis revisões de acordo com alguma métrica;
      atualizar melhor revisão possível;
   implementar melhor revisão;
até que nenhuma revisão melhore a teoria
```

Figura 4 - Algoritmo do Sistema de Revisão de Teorias

2.3. Refinamento de Ontologias Através de Revisão de Teoria

O Refinamento de ontologias através de técnicas de revisão de teoria foi proposto inicialmente em [Leão et. al., 2010], utilizando a ontologia da Família. O conjunto de axiomas da ontologia foi mapedo para cláusulas Horn de primeira ordem, definindo o conjunto CP e o conjunto A das entradas do sistema FORTE. Além disso, as instâncias da ontologia foram utilizadas como exemplos positivos e na geração de exemplos negativos. Após a revisão das cláusulas constantes em A pelo FORTE, estas são convertidas para a linguagem utilizada para descrever os axiomas. É importante observar que, devido ao seu poder de expressão, a lógica de primeira ordem (neste caso clausula de Horn) é capaz de expressar todo o conhecimento originalmente expresso em OWL e que as alterações propostas pelo FORTE, também são possíveis de serem representadas em OWL

A proposta se provou de grande utilidade ao gerar axiomas revisados, capazes de representar o domínio de estudo com maior acurácia. Entretanto, a variedade de linguagens de representação de ontologias ocasiona o aumento de complexidade no uso de sistemas de revisão de teoria para o refinamento de ontologias, dado que para cada linguagem um determinado procedimento de mapeamento deverá ser executado. Além disso, é necessário um conhecimento prévio de como executar o sistema de revisão.

3. ORION – Um Framework para Refinamento de Ontologias

Objetivando facilitar a utilização de sistemas de revisão de teoria no refinamento de ontologias, propõe-se criar o framework ORION (Ontology Refinement through RevisION techniques), capaz de receber uma ontologia e devolvê-la refinada, abstraindo as conversões necessárias e o conhecimento sobre o funcionamento do sistema de revisão de teoria. O framework converte automaticamente os dados de uma ontologia para o formato aceito como entrada pelo FORTE e o executa. Uma vez revisados os

axiomas, o conversor atua novamente, realizando o caminho inverso, convertendo a saída do sistema de revisão para uma nova ontologia. A ontologia resultante será um modelo aprimorado da ontologia inicial, possuindo um maior poder de expressividade. O esquema de funcionamento do framework ORION pode ser visto na Figura 5.

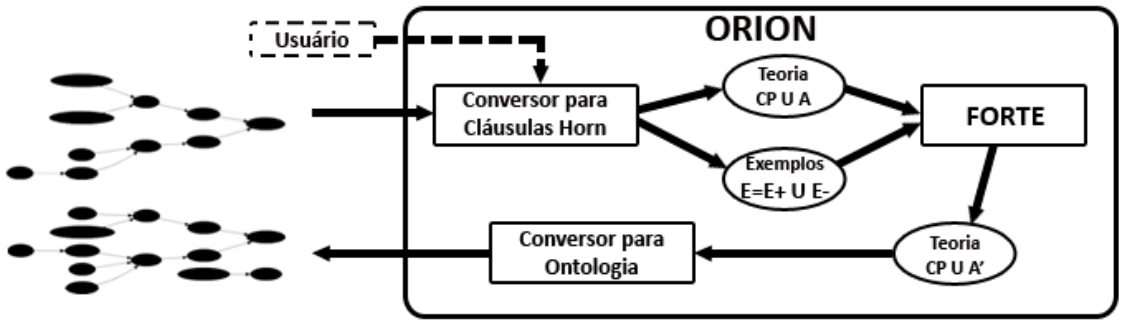

Figura 5 - Esquema de funcionamento do framework ORION

Dependendo do nível de complexidade do domínio retratado pela ontologia, o mapeamento terá de ocorrer de forma semi-automática, fazendo uso da interação com o usuário para estabelecer, por exemplo, axiomas base, que não devem ser revisados decidindo assim o que fará parte dos conjuntos CP e A.

Neste trabalho será considerada apenas a OWL como sintaxe para construção de ontologias, tanto por ser um padrão recomendado pelo W3C [Klyne e Carroll, 2004] e uma das linguagens mais utilizadas pela comunidade acadêmica quanto pela sua importância para a viabilidade da Web Semântica.

3.1. Mapeamento de uma Ontologia para o Modelo de Teoria do FORTE

Esta seção descreve como alguns elementos previstos na sintaxe da linguagem OWL são mapeados para elementos utilizados no modelo de teoria do sistema FORTE.

3.1.1. Propriedades de Objetos (Relacionamentos entre classes)

Object Properties são a representação dos relacionamentos entre conceitos em uma Ontologia. Para cada *Object Property*, é gerada uma cláusula para representar o relacionamento. No metamodelo da OWL, uma *Object Property* é descrita pelos atributos *About*, *Domain* e *Range*, que são mapeados, respectivamente, como o nome da cláusula, o seu primeiro termo e o seu segundo termo. Adicionalmente, os tipos do domínio (*domain*) e do escopo (*range*) do relacionamento são adicionados como antecedentes da cláusula, para restringir as interpretações possíveis das instâncias que podem estar relacionadas no conjunto de fatos. Um exemplo de conversão do *Object Property* **cursa** pode ser visto na Figura 6. Uma vez gerados, os relacionamentos são incluídos no conjunto CP ou no conjunto A, dependendo da indicação do especialista do domínio se tal cláusula deve estar sujeita (A) ou não (CP) à revisão.

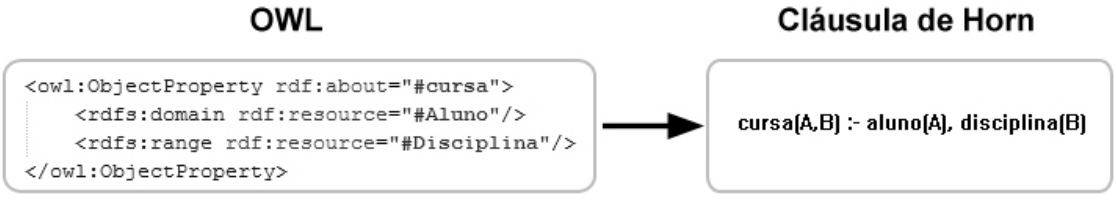

Figura 6 - Mapeamento de *Object Property* para uma Cláusula de Horn

3.1.2. Propriedades de Tipos de Dados (Atributos da classe)

Uma propriedade de tipo de dado (*Datatype Property*) representa, na interpretação mais comum da sintaxe OWL, os atributos de uma classe da Ontologia. O mapeamento de um *Datatype Property* ocorre de forma semelhante à conversão dos *Object Properties*, sendo utilizados principalmente os atributos *About* e *Domain*. A Figura 7 ilustra a conversão do *Datatype* **genero** da ontologia para o atributo que deverá ser inserido entre os *object_attributes* do FORTE.

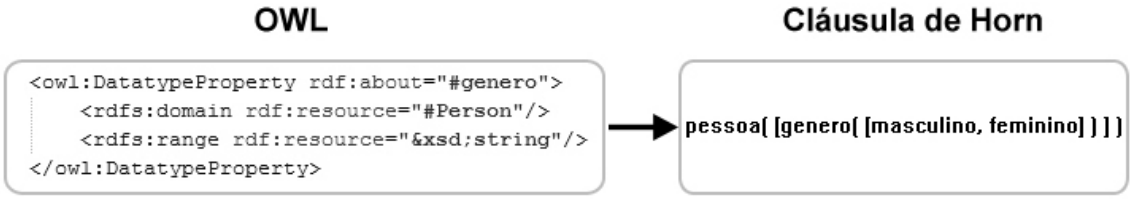

Figura 7 - Mapeamento de um *Datatype Property* para uma Cláusula de Horn

Vale ressaltar que em ontologias de domínio reais pode haver um número muito grande de atributos (*Datatype Properties*) para cada classe da Ontologia, o que acarretaria em um aumento significativo do tamanho da teoria a ser revista e consequentemente um aumento do custo de revisão, podendo tornar o processo inviável na prática e diminuir a acurácia da teoria revisada. No entanto, a relevância de cada atributo para o domínio em questão e para o processo de revisão é bastante variável, existindo atributos essenciais para a revisão (por exemplo, o atributo **gênero** de uma Pessoa quando se fala de um domínio médico, onde seria incorreto dizer que uma pessoa do gênero masculino pode estar grávida) e atributos que poderiam ser descartados do processo de revisão (por exemplo, o atributo email de uma Pessoa). Este problema é análogo ao problema de seleção de atributos na área de mineração de dados [Han e Kamber, 2006], para o qual existem várias técnicas e algoritmos propostos na literatura. Existe um consenso, no entanto, de que a decisão pelos atributos relevantes a serem considerados deve ser do especialista do domínio. Portanto, ao mapear as propriedades de tipos de dados, o ORION prevê interação com o usuário para definir quais delas, que de outra forma seriam ignoradas, devem ser incluídas.

Uma restrição do sistema FORTE é não permitir atributos contínuos, portanto é necessário informar os valores que o *object_attribute* poderá assumir. No caso exibido na Figura 7 foi necessário informar ao conversor que os valores possíveis para **genero** eram **masculino** e **feminino**.

Alguns casos extras podem ser previstos, como a possibilidade de um *Datatype Property* se referir a mais de um conceito em seu atributo *Domain*. Neste caso seria necessário gerar a referencia do atributo para cada um dos conceitos no FORTE.

3.1.3. Class

A tag *class* da OWL é utilizada na definição de um conceito da ontologia, podendo agregar em si múltiplos axiomas, que juntos formam a sua descrição formal. Para o sistema de revisão o conceito é uma regra, definida através de relacionamentos.

Toda *class* deve possuir pelo menos um axioma *subClassOf*, relacionando-a a um outro conceito da ontologia, indicando quais propriedades deverá herdar. Caso o axioma seja omitido, considera-se que o objeto é subclasse do objeto raiz *Thing*. O axioma *subClassOf* pode ainda ser utilizado em conjunto com a tag *Restriction* para definir algumas restrições para a semântica da classe. Em conjunto, todas as

características e propriedades de uma classe formarão a cláusula de Horn que definirá o conceito para o sistema de revisão. Na Figura 8 é mostrado o conceito **Professor,** composto por três axiomas *subClassOf*, sendo dois deles restrições sobre relacionamentos obrigatórios ao conceito, um com valor específico e um fazendo referência a qualquer instância de um determinado conceito.

```
<owl:Class rdf:about="#Professor">
    <rdfs:subClassOf rdf:resource="#Pessoa"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#temQualificacao"/>
            <owl:hasValue rdf:resource="#DSc"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#temTopicoPesquisa"/>
            <owl:someValuesFrom rdf:resource="#TopicoPesquisa"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Figura 8 - Definição OWL para o conceito Professor

O conversor deverá mapear a classe gerando a clausula Horn **professor(A)**, cujo corpo é mostrado pela Figura 9.

```
professor(A):- pessoa(A),temQualificacao(A, DSc),temTopicoPesquisa(A, T),TopicoPesquisa(T)
```

Figura 9 - Cláusula de Horn gerada pelo conversor definindo o conceito Professor

A especificação OWL comporta ainda outros tipos de axiomas, como o *disjointWith,* que pode ser visto na Figura 10 e indica a disjunção entre os conceitos **Aluno** e **Professor** (para o domínio exemplo nenhum **Aluno** pode ser **Professor**).

```
<owl:Class rdf:about="#Aluno">
    <rdfs:subClassOf rdf:resource="#Pessoa"/>
    <owl:disjointWith rdf:resource="#Professor"/>
</owl:Class>
```

Figura 10 - Definição OWL para o conceito Aluno

A conversão do axioma *disjointWih* não é direta como a do axioma *subClassOf*, já que isto requer a representação de antecedentes negativos a uma cláusula, e o sistema de revisão FORTE não é capaz de revisar cláusulas com antecedentes negativos. Neste caso, optou-se por definir um novo conceito para o conjunto disjunto de instâncias, e restringir que este novo conceito seja incluído como antecedente negativo em cláusulas pertencentes ao conjunto CP, ou seja, não sujeitas à revisão. O novo conceito então poderá ser utilizado como antecedente tanto em A quanto em CP. Foi convencionado utilizar a prefixo **não** para indicar este tipo de cláusula. No exemplo, definiu-se o conceito **naoProfessor**, e então utilizou-se este como antecedente no corpo da cláusula que define o conceito **Aluno**. A Figura 11 mostra, respectivamente, a definição das cláusulas de Horn **naoProfessor(A)** e **aluno(A)**.

```
naoProfessor(A) :- not profesor(A).
-------------------------------------
aluno(A) :- pessoa(A), naoProfessor(A).
```

Figura 11 - Cláusula de Horn definindo os conceitos naoProfessor e Aluno

3.1.4. Instâncias

As instâncias da ontologia servirão para formar o conjunto E a ser aplicado ao sistema de revisão. Baseado neste conjunto, o sistema de revisão é capaz de verificar inconsistências na teoria e propor modificações, a fim de provar todos os exemplos positivos e negar todos os negativos O mapeamento das instâncias OWL em exemplos do FORTE é explicado a seguir.

3.1.4.1. Exemplos Positivos (E^+)

As instâncias da ontologia, por natureza, constituem o subconjunto de exemplos positivos, dado que se referem aos indivíduos existentes na ontologia. Uma única instância da ontologia é capaz de gerar mais de um exemplo para o conjunto E^+.

O primeiro exemplo a ser obtido é o conceito instanciado. Na Figura 12 pode-se ver que uma instância pode se referir ao conceito instanciado pela própria *tag*. Neste caso seria gerado o exemplo positivo **topicoPesquisa(inteligenciaArtificial)**.

```
<TopicoPesquisa rdf:about="#InteligenciaArtificial">
    <rdf:type rdf:resource="&owl;Thing"/>
</TopicoPesquisa>
```

Figura 12 - Representação OWL para uma instância de TopicoPesquisa

A Figura 13 ilustra uma instância do conceito Professor, de onde é possível extrair mais exemplos positivos com base nos relacionamentos descritos pela instância.

```
<Professor rdf:about="#FernandaBaiao">
    <rdf:type rdf:resource="#Professor"/>
    <firstName rdf:datatype="&xsd;string">Fernanda</firstName>
    <lastName rdf:datatype="&xsd;string">Baião</lastName>
    <genero rdf:datatype="&xsd;string">feminino</genero>
    <matricula rdf:datatype="&xsd;int">210503082</matricula>
    <homepage xml:lang="en">http://uniriotec.br/~fernanda.baiao</homepage>
    <temTopicoPesquisa rdf:resource="#InteligenciaArtificial"/>
    <temTopicoPesquisa rdf:resource="#Ontologia"/>
    <pertenceCentro rdf:resource="#CCET"/>
    <temQualificacao rdf:resource="#DSc"/>
    <orientadorDe rdf:resource="#FelipeLeao"/>
</owl:Professor>
```

Figura 13 - Instância utilizada para a extração de exemplos positivos

Desta instância podemos retirar exemplos como:

```
professor(fernandaBaiao);
temTopicoPesquisa(fernandaBaiao, inteligenciaArtificial);
pertenceCentro(fernandaBaiao, ccet);
temQualificacao(fernandaBaiao, dsc);
orientadorDe(fernandaBaiao, felipeLeao).
```

É possível ainda extrair outros exemplos com base nos *Datatype Properties* incorporados pela instância, como por exemplo, **genero(fernandaBaiao, feminino)**.

3.1.4.2.Exemplos Negativos (E^-)

Para ser capaz de revisar uma teoria, o sistema de revisão precisa receber um subconjunto de exemplos negativos, que seriam as afirmações incorretas do estado atual do domínio. No entanto, como a ontologia representa somente as afirmações corretas sobre o domínio representado, não é possível extrair os exemplos negativos diretamente da ontologia.

Os Exemplos negativos deverão ser gerados através da negação de afirmações presentes na ontologia. Uma combinação de métodos pode ser utilizada. Por exemplo, os axiomas de classe *disjointWith*, apresentados na subseção 3.1.3. Segundo este axioma, os conceitos **Aluno** e **Professor** são disjuntos (Figura 10), portanto , o exemplo positivo **professor(fernandaBaiao)**, pode ser utilizado para gerar o exemplo negativo **aluno(fernandaBaiao)**, da mesma forma que um exemplo positivo de aluno pode ser utilizado na geração de um exemplo negativo de professor. Outro método para construção de exemplos negativos seria considerar o axioma de classe *subclassOf*. Sendo a classe **Monitor** (Figura 14) uma subclasse de **Aluno** (Figura 10 da subseção 3.1.3), um exemplo positivo **aluno(felipeLeao)** pode ser utilizado na criação do exemplo negativo **monitor(felipeLeao)**, já que uma instância definida como Aluno não deve ser entendida como uma instância da classe Monitor.

```
<owl:Class rdf:about="#Monitor">
    <rdfs:subClassOf rdf:resource="#Aluno"/>
</owl:Class>
```

Figura 14 - Definição OWL para o conceito Monitor

3.2. Mapeamento do Modelo Revisado para uma Ontologia

Com base na teoria revisada pelo sistema de revisão o framework ORION montará uma ontologia refinada em OWL capaz de representar com maior precisão o domínio estudado. A ontologia refinada comportará o mesmo conjunto de instâncias da ontologia original, dado que o sistema de revisão não permitirá que exemplos positivos deixem de ser provados pela teoria revisada.

As cláusulas de Horn que representam relacionamentos possuem em sua cabeça a indicação do relacionamento entre duas variáveis, enquanto as cláusulas que definem conceitos possuem em sua cabeça somente uma variável. O framework mapeará relacionamentos para *Object Properties* e conceitos para *Classes*. A Figura 15 exibe duas cláusulas de Horn que serão convertidas, respectivamente, para um objeto *Class* e um *Object Property*.

```
centro(A) :- temCurso(A,C),curso(C).
-------------------------------------
temCurso(X,Y) :- centro(X), curso(Y).
```

Figura 15 – Cláusulas de Horn representando um conceito e um relacionamento

Cláusulas de Horn que representam conceitos terão em seu corpo relacionamentos e referências a outros conceitos. Relacionamentos deverão ser traduzidos para restrições (através da combinação entre o axioma *subClassOf* e a tag

Restriction), e referências a outros conceitos poderão representar a relação de subclasse de um outro conceito ou auxiliar na definição de uma restrição (como a variável C auxilia a definir qualquer instância do conceito Curso). Caso um relacionamento se refira a uma instância específica, a restrição deverá se referir a este valor, caso faça referência a uma variável, será utilizado o atributo RDF *someValuesFrom.* O conceito expresso pela cláusula **centro(a)** na Figura 15 foi mapeado para um objeto *Class* da OWL na Figura 16.

```
<owl:Class rdf:about="#Centro">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#temCurso"/>
            <owl:someValuesFrom rdf:resource="#Curso"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Figura 16 - Definição do conceito Centro na linguagem OWL

Ao mapear um relacionamento para um *ObjectProperty*, o conceito expresso pela primeira variável será o *Domain*, enquanto o segundo será o *Range*. O resultado do mapeamento do relacionamento **temCurso(X,Y)**, mostrado previamente pela Figura 15, pode ser visto na Figura 17.

```
<owl:ObjectProperty rdf:about="#temCurso">
    <rdfs:domain rdf:resource="#Centro"/>
    <rdfs:range rdf:resource="#Curso"/>
</owl:ObjectProperty>
```

Figura 17 - Definição do relacionamento temCurso na linguagem OWL

Na seção 3.1.3 falou-se sobre a geração de cláusulas com o prefixo **nao**, necessárias para indicar negação no corpo de cláusulas que deverão ser revisadas pelo sistema de revisão de teorias. Estas cláusulas não serão mapeadas diretamente, mas servirão para indicar disjunções (através do axioma *disjointWith*) em objetos *Class*.

3.3. Discussões e Resultados Obtidos

O framework ORION se encontra em estágio de desenvolvimento, com a implementação do mapeamento conforme especificado neste trabalho. Testes preliminares foram executados com pequenos trechos em OWL, resultando nas cláusulas de Horn ilustradas anteriormente. A viabilidade de aplicação do sistema FORTE para revisão de uma ontologia no domínio da família também já foi verificada em Leão et al. (2010), onde foram obtidos resultados positivos, com a teoria sendo revisada com sucesso aumentando sua acurácia de representação.

Atualmente estão sendo realizados testes com o framework Jena[3], que permite manipular e realizar inferências a partir de um modelo ontológico. O Jena está sendo utilizado para recuperar os dados da ontologia, que serão convertidos pelo ORION para o formato aceito pelo FORTE.

[3] http://jena.sourceforge.net/

4. Trabalhos Relacionados

Alguns trabalhos na literatura abordam o tema de revisão de ontologias. Kang e Lau (2007) propuseram a aplicação de técnicas de revisão de crenças (*belief revision*) para revisão de ontologias. No entanto, a técnica de revisão de crenças é monotônica, ou seja, as modificações na base de conhecimento são restritas à remoção de assertivas da base de conhecimento. Por conta disto, Sun et al. (2006) propuseram um novo operador de revisão para lidar com a não monotonicidade do problema de revisão de crenças, e ilustrou a utilização deste operador em um exemplo.

As vantagens do framework ORION advêm do fato de que a restrição da monotonicidade não se aplica à técnica de revisão de teorias, a qual contempla operadores de generalização e de especialização do conjunto de regras e já apresenta resultados satisfatórios para a revisão de teorias descritas em cláusulas de primeira ordem Horn e livre de funções.

5. Conclusão

Ontologias são artefatos de grande utilidade na descrição de domínios para diversos fins. Tipicamente ontologias são construídas por especialistas, considerando seu conhecimento preliminar sobre o domínio. Novas instâncias do domínio podem ser constantemente incorporadas a ontologias, o que pode resultar em um modelo inconsistente com algumas instâncias, especialmente quando estas instâncias representam situações previamente desconhecidas pelo especialista. Quando a ontologia se encontra em estado inconsistente é preferível refiná-las automaticamente ao invés de reaprendê-la do zero, ou tentar modificá-la manualmente. Motivados por estes fatos, neste trabalho propusemos um framework capaz de receber uma ontologia e refiná-la, através do uso de técnicas de revisão de teorias. Foi descrito como os elementos da ontologia devem ser convertidos para uma teoria a ser processada pelo sistema de revisão e como a teoria revista pode ser mapeada para um modelo ontológico que represente com maior acurácia o domínio alvo.

Trabalhos futuros incluem a extensão do framework para trabalhar com múltiplas linguagens de definição de ontologias além da OWL, como lógica de descrição (*Description Logic*), lógica de primeira ordem (*First-Order Logic*) e OBO (*Open Biomedical Ontologies*). Por outra perspectiva, é possível que o domínio esteja associado à incerteza, possuindo axiomas probabilísticos, formalmente representados utilizando ICL [Poole, 1993], BLP [Kersting e De Raedt, 2001]ou Probabilistic Prolog [De Raedt et al., 2007], desta forma propomos também trabalhar com o sistema de revisão de teorias probabilísticas PFORTE [Paes et al., 2006] para refinar ontologias probabilísticas. Finalmente, pretende-se propor técnicas adicionais para geração de exemplos negativos.

Agradecimentos

O primeiro autor gostaria de agradecer à FAPERJ pelo apoio recebido

Referências Bibliográficas

Buntine, W. (1991). “Theory refinement on Bayesian networks”. In Proceedings 17th Conference Uncertainty in Artificial Intelligence, pages 52–60, San Mateo, CA.

De Raedt, L., Kimmig, A., and Toivonen, H. (2007). "Problog: A probabilistic prolog and its application in link discovery". In Proc. IJCAI-2007, pp 2462–2467.

Garcez, A. e Zaverucha, G. (1999). "The connectionist inductive learning and logic programming system". Applied Intelligence, 11:59–77.

Gruber, T. (1995). "Toward principles for the design of ontologies used for knowledge sharing". International Journal of Human-Computer Studies, 43:907-928.

Han, J.; Kamber, M. (2006). Data Mining: Concepts and Techniques, Morgan Kaufmann Pubs

Kang, S. and Lau, S. (2007) Ontology Revision: An Application of Belief Revision Approach, in Sharman, R. et al (eds.) Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems, Springer, pp. 297-318.

Kersting, K. and De Raedt, L. (2001). "Towards combining inductive logic programming with Bayesian networks". In Proceedings of the 12th Int. Conference on Inductive Logic Programming, LNAI 2157 Springer Verlag, Strasbourg, 118–131.

Klyne, G. e Carroll, J. (2004). "Resource description framework (rdf): W3C recommendation". available at: www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

Leão, F., Revoredo, K., Baião, F. (2010) "Ontology Refinement through Theory Revision Techniques", In III Workshop on Computational Intelligence, Brazil.

Mitchell, T. (1997). "Machine Learning'. McGraw-Hill, New York.

Muggleton, S. (1992). "Inductive logic programming". Academic Press, New York.

Paes, A., Revoredo, K., Zaverucha, G., Costa, V. S. (2006). "Pforte: Revising probabilistic fol theories". Proc 18th Brazilian AI Symposium (SBIA-06), LNAI 4140, pp 441–450. Springer.

Poole, D. (1993). "Probabilistic Horn abduction and Bayesian networks". Artificial Intelligence, 64(1):81–129.

Ramachandran, S. e Mooney, R. (1998). "Theory refinement of bayesian networks with hidden variables". Proc 15th Intl Conference on Machine Learning (ICML), pp 454–462.

Richards, B. L. and Mooney, R. J. (1995). "Automated refinement of first-order Horn-clause domain theories. Machine Learning", 19:95–131.

Towell, G. e Shavlik, J. (1994). Knowledgebased artificial neural networks. Artificial Intelligence, 70(1–2):119–165.

Wogulis, J. e Pazzani, M. (1993). "A methodology for evaluationg theory revision systems: results with Audrey II". In Proceedings of the 13th International Join Conference on Artificial Intelligence, pages 1128–1134, Chambery, France.

Wrobel, S. (1996). "First-order theory refinement". In Raedt, L. D., editor, Advances in Inductive Logic Programming, pages 14–33. IOS Press.

Sun, Y., Sui, Y., Zhiping, L. (2006). "One Axiomatic System for the Ontology Revision". Intelligent Information Processing III: 91-100