Revisão de Crenças em Lógicas de Descrição

MAC0499 - Trabalho de Formatura Supervisionado

Luís Felipe de Melo Costa Silva

Orientadora: Profª Dra. Renata Wassermann

Instituto de Matemática e Estatística
Universidade de São Paulo
luis.melo.silva@usp.br

1. Introdução

FAZER

2. Ontologias

As ontologias, termo importado da Filosofia e que por lá significa "O estudo do ser enquanto ser", são, para a Ciência da Computação, reuniões de sentenças lógicas que exibem alguma informação sobre alguma área do mundo para resolver algum problema representado a ela.
São estudadas na área de Inteligência Artificial e, uma vez construídas, permitem comunicação, compartilhamento e reúso de conhecimentos, interessando várias áreas do saber, tais como a ciência da computação, filosofia, biologia, lógica e linguística.
As ontologias representam a explicitação de uma conceitualização. Uma conceitualização nada mais é do que uma modelagem de algum domínio do conhecimento, codificada por uma linguaem formal de representação.
Para ser construída, uma ontologia precisa de clareza, coerência, estendibilidade, viés mínimo de codificação e compromisso ontológico mínimo. Além disso, precisa de seus componentes. São eles: **Classes**; **Propriedades**: **Relações**: **Restrições**; e **Instâncias**.
A criação das ontologias costuma ser feita por equipes interdisciplinares. Um exemplo é a Gene Ontology. Elas sempre modelam áreas do conhecimento, nunca fatos específicos, e são produzidas para que todos possam acessar suas informações.

3. Lógicas de Descrição

Para caracterizar os axiomas lógicos de uma ontologia, é recomendado o uso de uma linguagem formal de representação. As Lógicas de Descrição (LD) são um bom meio para atingir o formalismo necessário para modelar uma ontologia. Elas são subconjunto da Lógica de Primeira Ordem (LPO), que por sua vez é uma extensão da Lógica Proposicional.
Para organizar o conhecimento, as Lógicas de Descrição utilizam algumas definições. Todas elas são utilizadas na construção de ontologias. São, a seguir:

- **Conceitos** Também chamados de classes, representam um conjunto de indivíduos. Nas ontologias, são equivalentes às Classes.
- **Papéis** Podem ser denotados como propriedades. Retratam as relações binárias que existem entre os indivíduos. Basicamente, são as Relações de uma ontologia. Alguns papéis podem ter uma função de caracterização, ou seja, definindo alguns atributos para o conceito. Tal função corresponde a uma Propriedade de uma Classe, em uma ontologia.
- **Indivíduos** Representam os particulares que existem dentro de um Conceito. Para as ontologias, são as Instâncias.

O conhecimento fica, portanto, dividido em duas partes, chamadas de *TBox* e *ABox*. A *TBox* contém o conhecimento terminológico, ou seja, os conceitos, as propriedades e as restrições. Já a *ABox* possui o conhecimento assertivo, ou seja, como os indivíduos se encaixam nas definições da *TBox*.

4. Revisão de Crenças

A área que cuida das alterações em estados de crenças, que é o que um agente acredita em um determinado momento, é chamada de Revisão de Crenças. Sua motivação é bem simples. Quando e deseja editar uma ontologia, é necessário saber se essa alteração influencia no que já existe na ontologia, ou seja, se ela se torna inconsistente ou não. A inconsistência acontece quando existe um conceito e a sua negação ao mesmo tempo na ontologia.
Existem três operações básicas nessa área, aplicadas a uma base de conhecimento K e uma sentença α:

- **Expansão**: A mais básica das operações é denotada por $K + \alpha$. É apenas a adição de α à base K, sem preocupação alguma com a consistência.
- **Contração**: Escrita como $K - \alpha$, é o contrário da operação acima, ou seja, representa a remoção de α da base K. Aqui a consistência é desejada, portanto, deve-se remover também todas as sentenças que implicam α.
- **Revisão**: É uma junção das operações anteriores. É conhecida por $K * \alpha$ e é a adição de um novo conhecimento com a preocupação com a consistência. Portanto, é a adição de α à base K e a remoção de tudo que implica o complemento de α.

A Contração e a Revisão não são definidas diretamente, mas sim, por 6 postulados básicos e 2 postulados suplementares cada. Computacionalmente, as operações são implementadas por construtores, com exceção da Expansão, que consiste apenas na adição de uma sentença ao conjunto.
Os construtores estudados nesse trabalho foram:

Contração *Kernel*: Definida da seguinte forma:

$$K -_\sigma \alpha = K \setminus \sigma(K \perp\!\!\!\perp \alpha),$$

onde $-_\sigma$ é a operação chamada de Contração *Kernel*, σ, uma função de incisão, que seleciona no mínimo uma fórmula de cada elemento de um conjunto-*Kernel*, e $\perp\!\!\!\perp$, a representação de um conjunto-*Kernel*, que é o conjunto de todos os subconjuntos minimais de K que implicam α.

Contração *Partial Meet*: Escrita assim:

$$K -_\gamma \alpha = \bigcap \gamma(K \perp \alpha),$$

onde $-_\gamma$ é a operação chamada de Contração *Partial Meet*, γ é uma função de seleção que seleciona algumas crenças do conjunto, e $\perp$ é a representação de um conjunto-resíduo, que é o conjunto de todos os subconjuntos maximais de K que não implicam α.

Pseudocontração SRW: Com a seguinte fórmula:

$$K -_* \alpha = \bigcap \gamma(\mathrm{Cn}^*(K) \perp \alpha),$$

que é muito semelhante ao construtor anterior, exceto pelo uso de um operador de consequência lógica mais fraco do que o usual. Um operador de consequência é usado para inferir todo o conhecimento possível de uma base de conhecimento.

Revisão *Kernel*: Apresentada como:

$$K *_\sigma \alpha = K \setminus \sigma(K \downarrow\!\!\downarrow \alpha),$$

onde $*_\sigma$ é a operação chamada de Revisão *Kernel*, σ é uma função de incisão, e $\downarrow\!\!\downarrow$ é a representação de um conjunto-*kernel* para a revisão, que é o conjunto de todos os subconjuntos minimais inconsistentes de K.

5. Ferramentas Computacionais

5.1 OWL

O surgimento da Web Semântica, no fim dos anos 1990, possibilitou o surgimento de uma linguagem de representação para axiomas em lógicas de descrição.
O grupo responsável pelos estudos dessa linguagem, chamado de *Web Ontology Working Group*, lançou a primeira versão da OWL (*Ontology Web Language* em 2002, e ela se tornou o padrão reconhecido pela W3C em 2004. Em 2007, a segunda versão da linguagem, a OWL 2 foi lançada, tornando-se o padrão para representação em 2012.
Essa última versão possui cinco sublinguagens, cada uma baseada em uma Lógica de Descrição diferente.

5.2 Protégé

Para facilitar o desenvolvimento e a edição de ontologias, foi criado, pelo Centro de Pesquisa para Informática Biomédica da Universidade de Stanford, o Protégé. Ele é um editor semântico compatível com a OWL 2.
Ele possui vários recursos. Uma ontologia sobre música poderia ficar assim no Protégé:

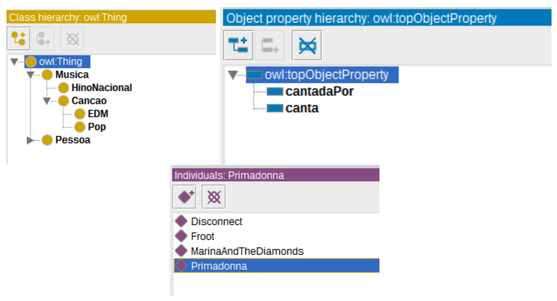

Figura 1: *A hierarquia das classes, as propriedades da ontologia e as instâncias definidas, respetivamente*

Ele também possui o recurso de raciocinadores, que permitem fazer inferências sobre a ontologia. Com o uso deles, é possível inferir que `Primadonna` **`cantadaPor`** `MarinaAndTheDiamonds`.
Além disso, também é possível fazer buscas na ontologia. Com o uso do SPARQL (sigla para SPARQL *Protocol and RDF QUery Language*, que possui uma sintaxe que lembra a do SQL, é possível tratar a ontologia como uma base de dados.

6. Implementação e Resultados

O **plug-in** construído nesse trabalho é uma reunião de implementações anteriores. As operações cobertas pelo programa são: **Contração *Kernel***, **Contração *Partial Meet***, **Revisão *Kernel*** e **Pseudocontração SRW**.
Ele é um **plug-in** linha de comando, e o seu desenvolvimento foi feito usando o IntelliJ. As dependências foram gerenciadas pelo *Apache Maven* 3.5.2. Para as lógicas internas, são necessários a *OWL API* 5.1.6, usada para todo o trabalho com os axiomas e o *HermiT Reasoner* 1.3.8, um motor de inferências. Para facilitar a entrada dos parâmetros, foi utilizado o *JCommander* 1.58.
São feitos aqui alguns exemplos simples para ilustrar a funcionalidade do *plug-in*. Todos os exemplos estão relacionados à ontologia discutida neste trabalho.

6.1 Contração *Kernel*

Para a contração *Kernel*, será usado um exemplo análogo. As classes estudadas aqui são Cancao, Pop e SynthPop. Os axiomas da ontologia são SynthPop $\sqsubseteq$ Pop e Pop $\sqsubseteq$ Cancao.
Suponha que o subgênero SynthPop evoluiu e está distante do Pop, mas ainda mantém o nome. Portanto, não se deseja mais que ele seja uma subclasse de Pop.

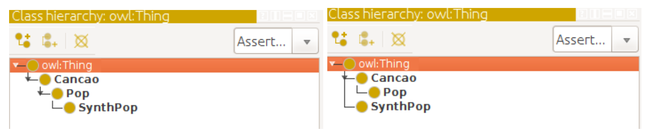

Figura 2: *As classes da ontologia e o resultado da operação.*

6.2 Contração *Partial Meet*

Para a contração *Partial Meet*, o exemplo utilizado tem as classes `Musica`, `Cancao` e `HinoNacional`, e os axiomas `HinoNacional` $\sqsubseteq$ `Cancao` e `Cancao` $\sqsubseteq$ `Musica`. Como definido anteriormente, na verdade, `Cancao` e `HinoNacional` são classes na mesma hierarquia. Após a sua execução, acontece o desejado, mas aparentemente, esse não é o melhor resultado.

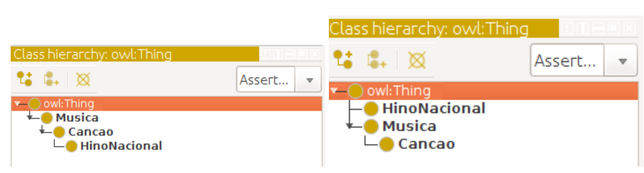

Figura 3: *A estrutura da ontologia e resultado da operação.*

6.3 Revisão *Kernel*

O exemplo usado para a Revisão *Kernel* será simples também. Sejam um fragmento da ontologia de música os seguintes axiomas: `FunkAmericano` $\sqsubseteq$ `Cancao` e `FunkBrasileiro` $\sqsubseteq$ `FunkAmericano`.
No entanto, hoje já se classifica o Funk Brasileiro como distinto do Funk Americano. Logo, vamos adicionar à nossa base que os dois subgêneros são disjuntos. Após a execução da operação, temos o seguinte resultado. Note que a propriedade de disjunção está no produto final.

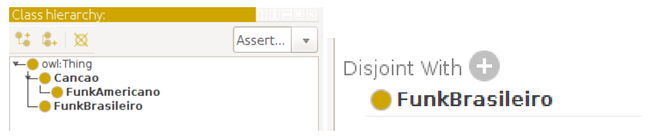

Figura 4: *O resultado da operação e a presença da disjunção, na classe* `FunkAmericano`*, respectivamente.*

6.4 Pseudocontração SRW

Para a Pseudocontração SRW, serão consideradas as classes `Cantor` e `Compositor` da ontologia musical, e o seguinte axioma: `Compositor` $\sqsubseteq$ `Cantor`. Ela possui um indivíduo, `markRonson`, pertencente à classe `Compositor`. Tudo isso implica que `markRonson` também é um `Cantor`, o que está errado. Depois da operação, temos que `Compositor` deixa de ser subclasse de `Cantor`, e então, `markRonson` não pertence mais à classe `Cantor`.

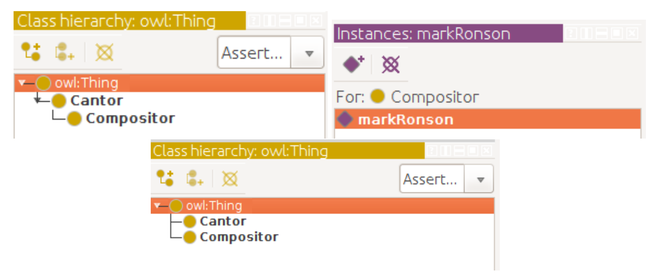

Figura 5: *A estrutura da ontologia, a instância e o resultado da operação.*