Revisão de Crenças em Lógica de Descrição e o Plug-In para o Protégé

Aluno: Fillipe Manoel Xavier Resina
Orientadora: Professora Renata Wassermann
Trabalho de Conclusão de Curso

UNIVERSIDADE DE SÃO PAULO
Instituto de Matemática e Estatística
Departamento de Ciência da Computação

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

RESUMO

As pesquisas na área de representação de conhecimento e modelagem de domínios têm crescido constantemente.

Nesse contexto, as **ontologias** têm um papel muito importante, pois fornecem uma representação formal do conhecimento. Entra em cena, então, a **Lógica de Descrição**, um dos principais formalismos para representação de conhecimento. Para tal, um dos *softwares* mais utilizados é o **Protégé**[5], programa editor de ontologias e baseado em Lógicas de Descrição.

Como conhecimento não é estático, o dinamismo faz com que o estudo de **revisão de crenças** ganhe importância, descrevendo como um agente, munido de um conjunto de crenças, deve mudá-las quando confrontado com novas informações e essa é a área de foco e estudo do projeto. A partir disso, visa-se desenvolver uma ferramenta que auxilie na revisão de ontologias.

1. Lógica de Descrição

A Lógica de Descrição (LD), além do formalismo citado, é a base da linguagem **OWL** (Web Ontology Language), recomendação da W3C desde 2004.

Em LD, o conhecimento é dividido em duas partes: **TBox** e **ABox**. Uma **TBox** refere-se à definição de conceitos e propriedades de um domínio (terminologias). Uma **ABox** refere-se à especificação de indivíduos de um ou mais conceitos (asserções).

Exemplo simples de um domínio:

1) Aves são Animais voadores

2) Pinguins são Aves

3) Tweety é um Pinguim

Em LD, temos:

- $\mathrm{Ave} \sqsubseteq \mathrm{AnimalVoador}$
- $\mathrm{Pinguim} \sqsubseteq \mathrm{Ave}$
- $\mathrm{Pinguim}(\mathrm{Tweety})$

2. Protégé

Como já citado, o Protégé é um editor de ontologias, sendo ele gratuito e de código aberto. Dispõe de muitos recursos e de uma rica interface gráfica. É possível criar uma ontologia, editá-la e fazer inferências sobre a mesma.

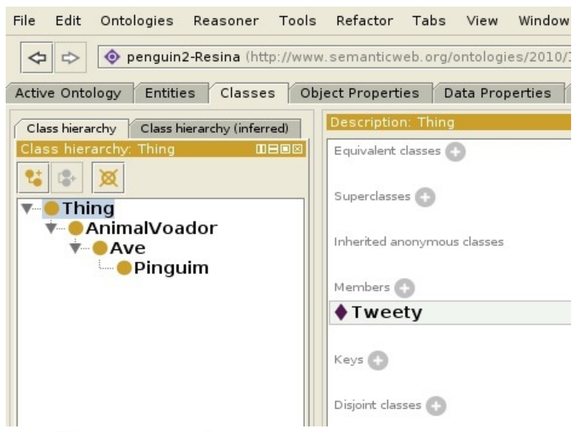

Figura 1: Ontologia da seção 1 anterior representada no programa

3. Revisão de Crenças

Em um conjunto de crenças, uma sentença A qualquer pode ser aceita, rejeitada ou indeterminada. Assim, uma **mudança de crença** em relação a uma sentença A pode ser de três tipos:

Expansão: A era indeterminada e agora é aceito ou A ou ¬A

Contração: A ou ¬A era aceita e agora A é indeterminada

Revisão: A era aceita e passa a ser rejeitada ou ¬A era aceita e passa a ser rejeitada.

Queremos garantir, além do **sucesso** da operação, o critério de **mudança mínima,** isto é, não será retirada nenhuma sentença desnecessariamente.

ALGORITMOS

Foram definidas duas funções de contração para aplicar a essa mudança: **Kernel** e **Partial Meet**. Como contração e revisão estão intimamente ligadas, por simplicidade falaremos aqui de contração.

Se queremos contrair uma sentença A de um conjunto, o algoritmo **Kernel** calculará os **conjuntos minimais que implicam A,** enquanto o **Partial Meet** computará os **conjuntos maximais que não implicam A** (conjunto resíduo). Por meio de qualquer um deles, eu obtenho um método confiável para contrair minha ontologia.

No Plug-In, referenciado na próxima seção, as operações foram implementadas por Kernel, sendo possível calcular os conjuntos resíduos pelo **Algoritmo de Reiter**[4] (cortes mínimos em conjuntos).

4. Plug-In

O Plug-In, ainda não completo, foi desenvolvido em **Java** a partir de um protótipo[4], por meio da **OWL API**[6] (ferramenta para trabalhar com ontologias OWL) e utilizando o motor de inferências (*reasoner*) **Pellet**[7]. Vamos vê-lo com o exemplo da seção 2 para contração:

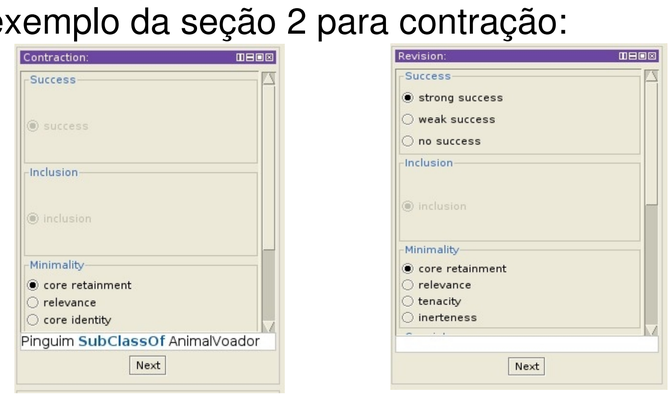

Figura 2: As telas do Plug-In para contração e revisão no programa; o de contração (esquerda) contém, no editor, a sentença que eu desejo contrair da ontologia da seção 2

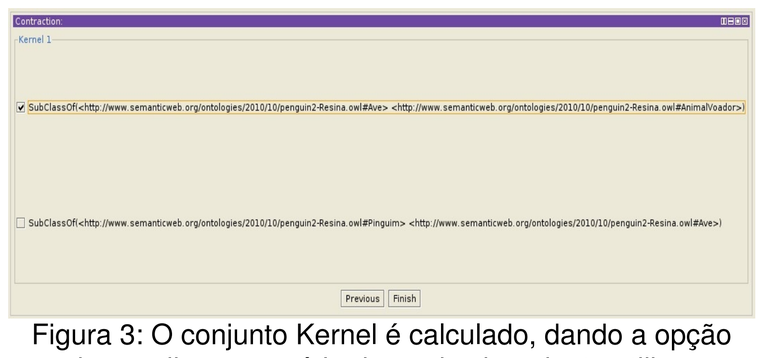

Figura 3: O conjunto Kernel é calculado, dando a opção de escolha ao usuário de qual subconjunto utilizar

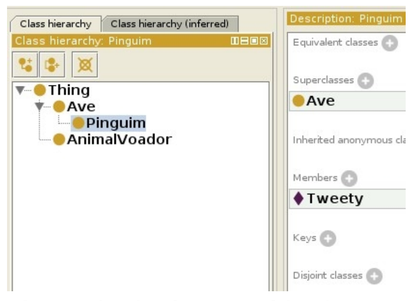

Figura 4: A ontologia é contraída de acordo, ou seja, Pinguim não é mais subclasse de AnimalVoador

Conclusão

Apesar dos muitos estudos e pesquisas, a área de representação de conhecimento ainda tem bastante a conquistar. No caso deste projeto, a existência de uma ferramenta que indique as possíveis melhores formas de revisar uma ontologia, e não apenas se ela é inconsistente ou não, é um grande avanço. Como **trabalho futuro**, continuarei a desenvolver o plug-in, contribuindo, também, para testar e avaliar a **eficiência** dos algoritmos Kernel e Partial Meet e avaliando em que situações um se mostra mais vantajoso que o outro.

REFERÊNCIAS

1. Peter Gardenfors. Knowledge in Flux - Modeling the Dynamics of Epstemic States. *MIT Press*, 1988.

2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. The Description Logic Handbook. *Cambridge University Press*, 2003.

3. M. Ribeiro, R. Wassermann. Base Revision for Ontology Debugging. *Journal of Logic and Computation*, Vol. 19, No. 5, 721-743, 2008.

4. M. Ribeiro and R. Wassermann. The Ontology Revisor Plug-In for Protégé. In *Proceedings of the Third Workshop on Ontologies and their Applications (WONTO 2008)*, 2008.

5. http://protege.stanford.edu/

6. http://owlapi.sourceforge.net/

7. http://clarkparsia.com/pellet/