Relatório de I.A.: OWL, Parte 1

Cauê Baasch de Souza
João Paulo Taylor Ienczak Zanette

12 de Outubro de 2018

TO-DO:

- Adicionar enunciado da 2ª parte (parte prática);
- Escolher domínio dos exemplos.

1 Geral

- Será necessário entregar com a parte 1 e uma breve descrição do domínio escolhido, bem como das principais classes e propriedades definidas, ontologias importadas (quando aplicável), e testes realizados com os indivíduos (o que foi inferido a partir da terminologia);
- Além disso, o arquivo RDF/OWL com o modelo conceitual.

2 Decisões do trabalho

2.1 Domínio escolhido

Linguagens, suas implementações e programas descritos nelas.

2.2 Principais Classes:

- **Language:** Representa linguagens de programação (e.g. Python, Ruby, C…);
- **Implementation:** Representa uma implementação de uma linguagem de programação. Por exemplo: CPython, Pypy e MicroPython são implementações de Python;
- **Compiler:** Representa um compilador, podendo incluir JITs. Identifica-se um compilador como AOT (*Ahead of Time*) pela propriedade `aot`. A negativa dessa propriedade identifica o compilador como JIT.
- **Interpreter:** Representa um interpretador de uma linguagem.

3 Parte 1: Pesquisa Teórica

Observações:

- O resultado da parte 1 deve ser um pequeno texto explicando o **entendimento** de vocês sobre os tópicos sugeridos. Portanto, o texto deve ter **no máximo** 2 páginas;
- Citar todas as fontes utilizadas para a pesquisa (Wikipédia também serve).

1. Em OWL 2, qual é a diferença entre os axiomas de class `subClassOf` e `equivalentTo`?
   - Apresente as definições de cada um e exemplos de uso dos dois, dentro do domínio escolhido pela dupla para a parte prática.
   - Descreva especialmente a diferença dos axiomas de classe quanto às inferências possíveis, ou seja, teste os exemplos no seu domínio e descreva as inferências.

Resposta:

`subClassOf:`

Dadas duas classes A e B e individuos que pertençam a apenas uma delas. Ao se fazer `subClassOf(:A :B)`, se está indicando que todo elemento pertencente a A também pertence a B, **mas o contrário não necessariamente**, ou seja: $A \subseteq B$. Sendo assim, para $\{1, 2, 3\} \in A$ e $\{4, 5, 6\} \in B$, `subClassOf(:A :B)` fará com que os indivíduos que pertencem a A e B sejam, respectivamente, $\{1, 2, 3\}$ e $\{1, 2, 3, 4, 5, 6\}$.

Exemplo:

No domínio escolhido para a parte prática, foi definido que toda **Máquina Virtual** (VM) é também um **Programa**. Porém, faz sentido que nem todo programa seja uma VM: nesse caso, não é possível inferir, por exemplo, que o GCC seja uma VM mesmo que seja um programa. Porém, é possível inferir que o GCC é um programa simplesmente por defini-lo como um compilador, afinal todo compilador é também um programa.

Na prática, tem-se as seguintes definições de classes em OWL2:

```
<owl:Class rdf:about="#Program">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>

<owl:Class rdf:about="#VirtualMachine">
  <rdfs:subClassOf rdf:resource="#Program"/>
</owl:Class>
```

E a instância "JVM" foi definida da forma:

```
<VirtualMachine rdf:about="#PyVM">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
</VirtualMachine>
```

Partindo dessa instância (e de outras VMs), é possível ver que, ao carregá-las para o leitor de OWL2, elas são atribuídas à classe `Program`:

```
Class own.tology.Program:
    Ancestors: {own.tology.Program, owl.Thing}
    Instances: [own.tology.PyVM, own.tology.JVM, own.tology.GraalVM]
```

É perceptível que, apesar de não declaradas explicitamente como programas, as instâncias foram inferidas como sendo da classe `Program`.

`equivalentTo:`

Dadas duas classes A e B e indivíduos que pertençam a apenas uma delas. Ao se fazer `equivalentTo`, se está indicando que todo elemento pertencente a A também pertence a B, **e vice-versa**. Sendo assim, para $\{1, 2, 3\} \in A$ e $\{4, 5, 6\} \in B$, `equivalentTo(:A :B)` fará com que tanto A quanto B sejam compostas pelos indivíduos $\{1, 2, 3, 4, 5, 6\}$.

Exemplo:

Foi definido que existem **Linguagens** e **Dialetos**. É fato que todo dialeto também é uma linguagem de programação, visto que tem um compilador próprio, não necessariamente um programa escrito em um dialeto compila na linguagem na qual foi baseado, além de tem suas próprias especificidades. Da mesma forma, a descrição de uma linguagem de programação descreve também um dialeto (como uma espécie de "dialeto-pai"). Porém, ainda assim, nem um nem outro são o mesmo conceito: dialetos são variações de uma linguagem, enquanto que linguagens são linguagens como ampla definição. Sendo assim, qualquer elemento descrito como linguagem será inferido como dialeto e vice-versa.

2. Compare a Lógica Descritiva que fundamenta a OWL 2 (na sua variação mais expressiva) com lógica de 1ª ordem. Apresente um exemplo do que é possível expressar com lógica de 1ª ordem que não conseguimos com lógica descritiva.

   Resposta:

   Na prática, DLs (*Descriptive Logics*) geralmente podem ser vistas como subconjuntos **decidíveis** de FOL (*First-Order Logic*) [1]. Isso significa que, para algumas DLs, é verdadeiro que um problema x pertencente ao conjunto de problemas descritíveis com tal DL é também descritível com FOL, porém o contrário não se mantém verdade, dado que FOL descreve também problemas indecidíveis.

   Da parte de OWL 2, esta é uma DL cuja expressividade é semelhante à de $\mathcal{SROIQ}^{(\mathcal{D})}$ (nomenclatura que se dá devido às descrições de DL fazem parte de OWL 2 — e.g. $\mathcal{I}$ indica que OWL 2 inclui propriedades de inversão; e $\mathcal{O}$ indica que inclui nominais tais como `owl:hasValue`, que impõem restrições aos valores dos objetos [2]). Porém, toda documentação encontrada sobre DLs é imprecisa sobre qual o ponto que divide ela de FOL, então sinceramente não faço a mínima ideia do que dá para fazer com FOL que não dê para fazer com DLs.

$$\forall x \forall y \forall z (\phi_T(x, y) \wedge \phi_T(y, z) \rightarrow \phi_T(x, z)) \tag{1}$$

   Equação 1: Relação de transitividade para uma determinada relação T. ϕT mapeia um conceito T descrito em DL a um predicado unário.

   Não conseguimos, porém, encontrar um exemplo claro do que não é possível adsf

4 Parte 2: Prática

1. Desenvolver um modelo conceitual (ontologia de domínio) utilizando OWL.

   Construam uma representação de domínio (utilizando *classes*, *relações* e *indivíduos*) de alguma área de interesse (pesquisa em computação, temas dos TCCs, filmes, esportes, músicas, hobbies, etc.).

   - A avaliação considera principalmente a utilização dos axiomas de propriedades e de classes. Quanto mais restrições forem modeladas, melhor;
   - Dentre as restrições, utilize no mínimo alguma de cardinalidade qualificada (`min`, `max`, ...) em alguma propriedade de objeto (ver na documentação da OWL, disponível no Moodle);
   - Outro aspecto importante é a definição dos membros das classes. Utilizem suas definições para testar a definição dos axiomas e também para testar o funcionamento do mecanismo de inferência da linguagem.

Referências

[1] Dmitry Tsarkov and Ian Horrocks. DL Reasoner vs. First-Order Prover. In *Description Logics*, 2003.

[2] Wikipedia. Description Logic.

[3] "Martin Kuba". "OWL 2 and SWRL tutorial". "`https://dior.ics.muni.cz/~makub/owl/`".

[4] Sebastian Rudolph. Foundations of Semantic Web Technologies, OWL 2 — Syntax and Semantics. "https://pdfs.semanticscholar.org/presentation/6bb5/b0107f88a98fd8c2d561c3cc71e3cc814303.pdf".