Guião Laboratorial #7

Web Semântica – Modelação de ontologias em OWL (parte 2)

Graça Gaspar
(com base em versões anteriores de João Marques da Silva e João Balsa)

Abril de 2020

Esta aula continua o estudo da criação de ontologias OWL, tirando partido da ferramenta Protégé. Vamos usar mais alguns tipos de expressões de definição de classes. Vamos ainda ver como representar afirmações de modo a que se possam obter certas inferências.

1 Classes descritas por expressões

Para os exemplos deste capítulo, vamos partir da ontologia do Ensino Superior que desenvolvemos na aula anterior (contida no ficheiro anexo `ensinoSuperior1.owl`).

Convém relembrar o que essa ontologia já contém. A figura 1 apresenta a hierarquia de classes e a hierarquia de propriedades.

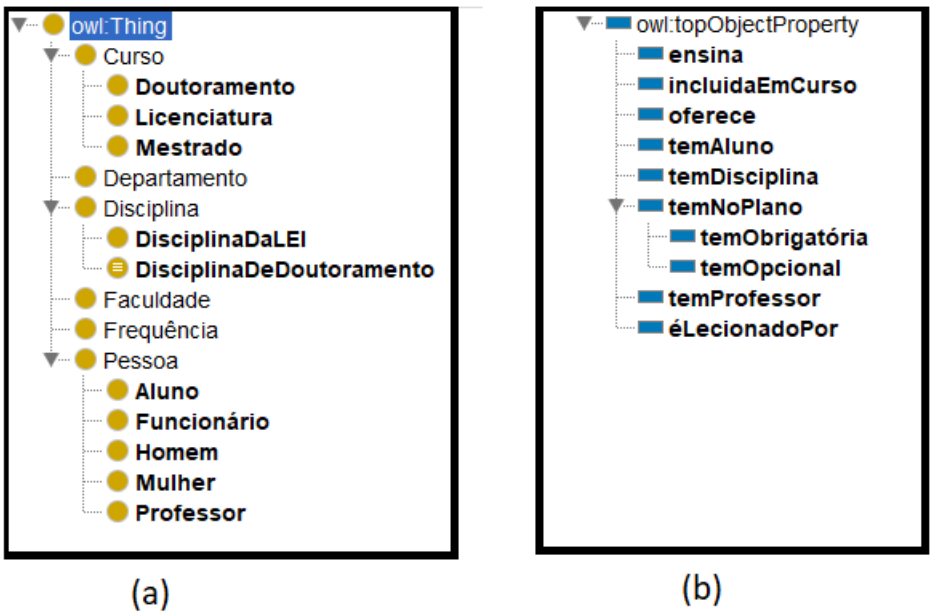

Figura 1: (a) Hierarquia de classes (b) Hierarquia de Propriedades

A figura 2 mostra como as propriedades estabelecem relações entre as principais classes :

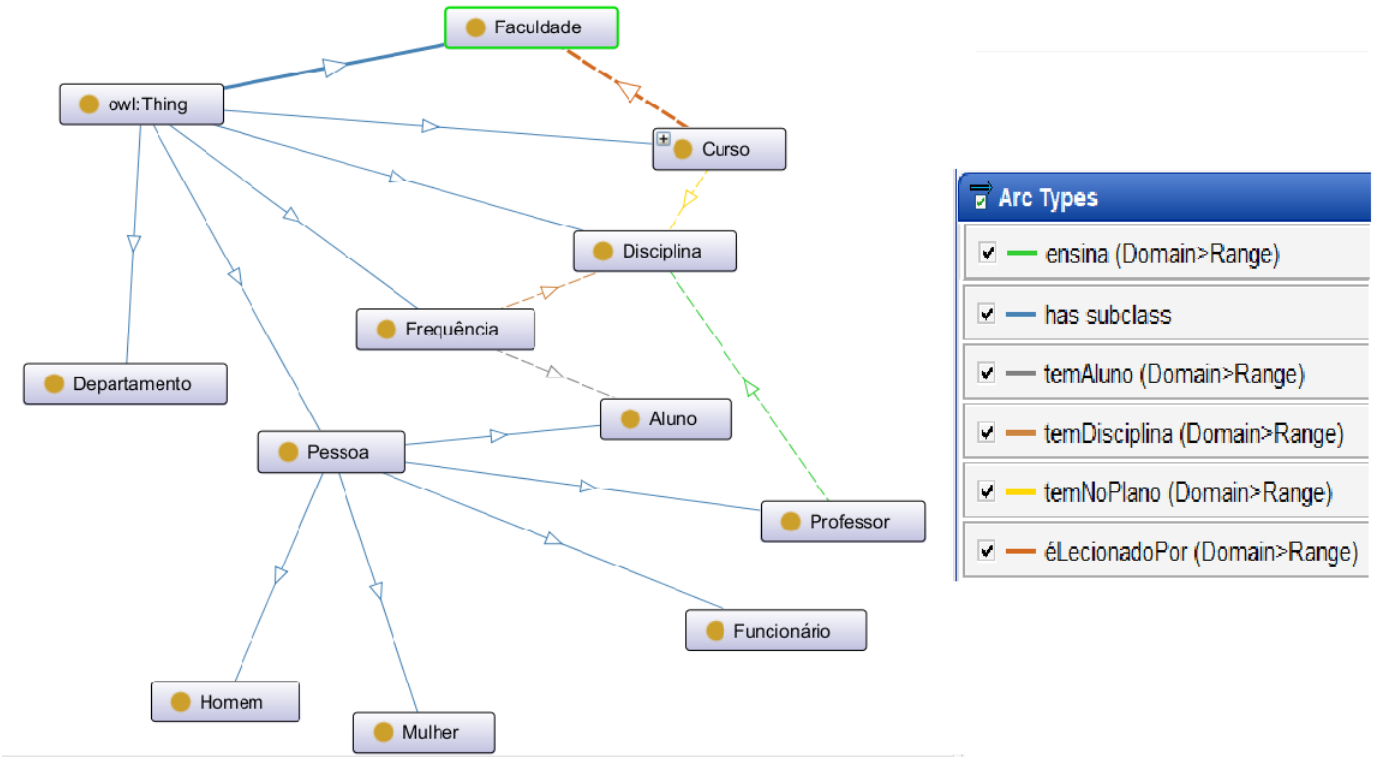

Figura 2: Representação gráfica da ontologia

Vamos ver exemplos de expressões para definir classes, que ainda não usámos na aula passada.

1.1 União e intersecção

É importante ter a noção de que, em OWL, **uma classe representa implicitamente um conjunto** de Individuals (instâncias).

Daí que existam formas de definir classes que correspondem a operações sobre conjuntos, embora também possam ser vistas (na sintaxe de Manchester é assim que se exprimem) como correspondendo à aplicação de operadores lógicos.

Exemplo 1:

Vamos definir uma nova classe CursoPosGraduacao como sendo a união da classe Mestrado com a classe Doutoramento.

Na sintaxe de Manchester essa definição corresponde a:

```
Class: CursoPosGraduacao
    EquivalentTo:
        Doutoramento or Mestrado
```

Nota: Para inserir uma destas expressões de classes em Protégé, após clicar no botão para adicionar uma subclasse ou uma equivalência, na janela que aparece escolha o Tab *Class Expression Editor*.

Vamos em seguida definir uma nova classe Contratado como sendo a união da classe Professor com a classe Funcionário. Podemos defini-la de forma análoga à que acabámos de usar. Mas, se quisermos indicar simultaneamente que é a união de duas classes que são disjuntas, também podemos defini-la como uma DisjointUnionOf.

E se quisermos definir a classe Monitor como correspondendo aos alunos que ensinam disciplinas?

Neste caso, trata-se de uma intersecção de duas classes, a classe Aluno e uma classe que ainda não descrevemos, embora tenha obviamente a ver com a propriedade ensina. Poderiamos exprimir a classe Monitor da seguinte forma:

```
Class: Monitor
    EquivalentTo:
        Aluno
         and (ensina some Disciplina)
```

Mas é preciso ter cuidado com as interseções de classes (na sintaxe de Manchester, classes ligadas por and):

> Classes disjuntas têm uma interseção vazia e o Reasoner irá indicar que a classe resultante é insatisfazível (no Protégé as classes classificadas pelo Reasoner como insatisfazíveis ficam com a cor vermelha).

Portanto, se declarámos que o domínio de ensina é Professor e a classe Professor é disjunta de Aluno temos que fazer alguma alteração: podemos estender o domínio de ensina, que poderá passar a ser (Professor **or** Aluno).

Repare que ao fazer essa alteração o *Reasoner* altera automaticamente o contradomínio da propriedade temProfessor, que é a inversa de ensina (para observar essa alteração pode ter que sincronizar de novo o *Reasoner*).

1.2 Complementaridade

Na teoria de conjuntos, o **complementar** de um conjunto X é o conjunto de todos os elementos que **não** pertencem a X.

Exemplo 2:

Vamos definir a classe Mulher como sendo complementar da classe Homem.

A seguinte definição está incorrecta:

```
Class: Mulher
    EquivalentTo:
                not Homem
```

Porquê? A classe complementar de Homem nesta ontologia inclui não só as Mulheres, mas também as Disciplinas, Faculdades, Cursos, etc. Assim, quando muito poderiamos dizer que Mulher é subclasse de not Homem.

A definição correcta seria:

```
Class: Mulher
    EquivalentTo:
        Pessoa
         and (not (Homem))
```

1.3 Restrições de cardinalidade

Existem ainda expressões que permitem definir uma nova classe por restrição do número de valores que uma dada propriedade pode ter, para cada instância dessa classe.

Existem 3 formas de restrições deste tipo, genéricamente chamadas restrições de cardinalidade: quando queremos impor um número exacto de valores, ou quando queremos impor um limite máximo ou um limite mínimo. Na sintaxe de Manchester, essas restrições usam as palavras reservadas **exactly**, **max** e **min**, repectivamente.

Exemplo 3:

Vamos definir a classe EnsinaVariasDisciplinas com o seguinte significado:
"Pessoas que ensinam no mínimo duas disciplinas"
Essa classe pode ser definida em Owl por:

```
Class: EnsinaVariasDisciplinas
    EquivalentTo:
        ensina min 2 Disciplina
```

Repare que classificar uma propriedade como **functional** equivale a dizer que o seu domínio tem no máximo 1 valor nessa propriedade.

O ficheiro `ensinoSuperior3.owl` é o resultado, na sintaxe de Manchester, de incluir na ontologia ensinoSuperior1 todos os exemplos deste capítulo.

Exercício E.1:

Considere as seguintes frases e a ontologia anexa `animais.owl` (comece por analisar essa ontologia, que é uma possível resolução dos exercícios do guião 6).
Use o Protégé para estender essa ontologia de modo a representar também as seguintes afirmações:

1. As costelas protegem quer o coração quer os pulmões (note-se que um orgão não pode ser simultaneamente coração e pulmão)
2. Os animais de rua não são domésticos
3. Há também animais abandonados que são animais de rua e têm dono
4. Os animais têm um único coração
5. Os animais têm normalmente dois rins, mas podem ter (no mínimo) só um

Apresente um ficheiro de texto `exerc1.owl` contendo essa ontologia na sintaxe de Manchester.

2 Ontologias e Inferência

Vamos agora focar-nos mais sobre aquilo que se pode inferir a partir de uma ontologia, usando um *Reasoner*.
Vamos usar uma outra ontologia, contida no ficheiro anexo `aldeiaBarjas.owl`, que poderíamos descrever pelas seguintes frases:

- Há uma aldeia chamada Barjas.
- As aldeias podem ter habitantes que são pessoas.
- As pessoas podem ser caracterizadas por várias propriedades: temIdade, temParceiro e temAmigo.
- As aldeias também podem ter amigos.

A ontologia aldeiaBarjas tem a seguinte descrição, na sintaxe de Manchester (repare no domínio da propriedade temAmigo):

```
Prefix: : <http://www.semanticweb.org/gg/ontologies/2020/3/aldeiaBarjas#>
Prefix: dc: <http://purl.org/dc/elements/1.1/>
Prefix: owl: <http://www.w3.org/2002/07/owl#>
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
```

```
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>

Ontology: <http://www.semanticweb.org/gg/ontologies/2020/3/aldeiaBarjas>

ObjectProperty: temAmigo
    Domain:
        Aldeia or Pessoa
    Range:
        Pessoa
ObjectProperty: temHabitante
    Domain:
        Aldeia
    Range:
        Pessoa
ObjectProperty: temParceiro
    Domain:
        Pessoa
    Range:
        Pessoa

DataProperty: temIdade
   Domain:
        Pessoa
    Range:
        xsd:int

Class: Aldeia
Class: Pessoa

Individual: Barjas
```

Os exemplos deste capítulo são apresentados sob a forma de diálogos entre dois interlocutores, o Tagarela e o Pensativo:

O Tagarela começa o diálogo, dando alguma informação.

O Pensador responde-lhe, dizendo o que conseguiu concluir.

Na verdade, queremos encarar o Pensador como correspondendo ao Reasoner a trabalhar sobre a ontologia em que se introduziu a informação dada pelo Tagarela.

Vamos ver o que necessitamos de introduzir na ontologia, para que se obtenham as conclusões pretendidas.

Diálogo 1:

```
Tagarela: A aldeia de Barjas tem homens e mulheres.  AL vive em Barjas.
Pensador: Isso quer dizer que AL é uma pessoa com mais de 60 anos,  homem ou
mulher.
```

Para que essa conclusão seja possível temos que:

1. Criar 2 novas classes: homens e mulheres.
2. Indicar que a classe das pessoas é a união disjunta dos homens e das mulheres.
3. Indicar que o Individual Barjas é do tipo ((tem some Homem) and (tem some Mulher))
4. Criar um Individual AL
5. Indicar que o Individual Barjas tem como habitante o AL
6. Definir a classe MaiorQue60 de quem tem mais de 60 anos
7. Indicar que o Individual Barjas é do tipo (temHabitante only MaiorQue60)

Os primeiros pontos são simples de resolver.

Os pontos 6 e 7 são mais complicados. Pode haver a tendência de representar a classe MaiorQue60 da seguinte forma que está errada:

```
Class: MaiorQue60
    SubClassOf:
        Pessoa and (temIdade min 60 xsd:int)
```

Porque é que está errada? O seu significado é "Pessoa que tem no mínimo 60 valores na propriedade temIdade" !!!

Então como representar?

- Queremos que uma pessoa só possa ter uma idade, logo temIdade será uma propriedade funcional.
- Para a classe MaiorQue60 queremos restringir o Range dessa propriedade: em vez de ser xsd:int, deverá ser um tipo que corresponda aos inteiros > 60.

Ainda não vimos como podemos restringir os valores de um DataType. O OWL permite fazer isso e o Protégé ajuda-nos a definir esses subtipos: Vamos usar o Tab Datatypes, criar um novo tipo dando-lhe um nome, por exemplo intMaiorQue60, e na sua descrição, clicar em Datatype Definitions e escrever a seguinte expressão:

```
xsd:int[> "60"^^xsd:int]
```

O ficheiro `aldeiaBarjas1.owl` tem o resultado da representação deste diálogo 1.

Experimente carregá-lo no Protégé para confirmar se o *Reasoner* consegue de facto inferir o pretendido.

Não se admire se nem todas as conclusões surgirem: a causa pode ser faltar explicitar alguma informação importante, mas também pode acontecer que nem todas as conclusões obtidas pelo Reasoner estejam a ser mostradas no Protégé nas janelas de descrição das várias entidades.

Para confirmar se é a segunda hipótese, pode usar o Tab DL Query (escolhendo Window $>$ Views $>$ Query Views $>$ DL Query) e indicar na caixa de texto a expressão que corresponde ao que queria concluir:

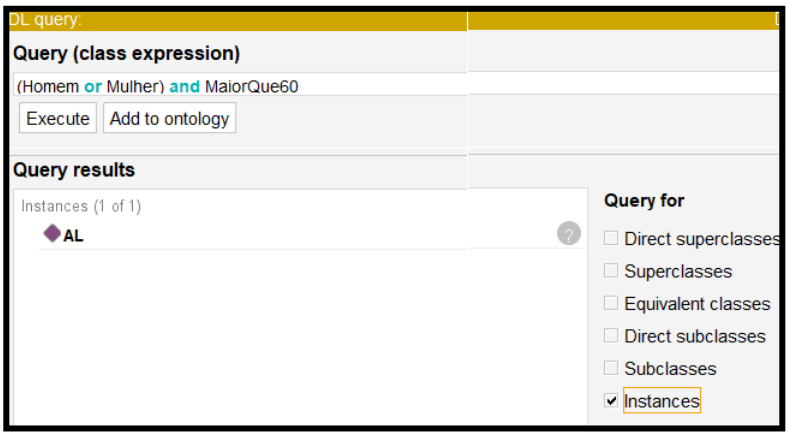

Figura 3: DL Query relativa ao diálogo 1 para obter as instâncias da classe

Diálogo 2:

```
Tagarela: Os habitantes de Barjas são monogâmicos (têm um único parceiro) e os
parceiros são sempre do sexo oposto. Há pessoas em Barjas sem parceiro, mas não
é o caso do AL.
Pensador: Então o Al tem uma parceira.
```

Para que essa conclusão seja possível temos que:

1. Definir uma subclasse de Pessoa chamada PessoaMonogamica
2. Caracterizar essa classe como tendo no máximo um valor na propriedade temParceiro.
3. Acrescentar em Barjas que é do tipo (temHabitante only PessoaMonogamica)
4. Acrescentar no Individual AL que é do tipo Homem e também que é do tipo (temParceiro some Pessoa). Este segundo tipo é fundamental para podermos concluir que AL tem de facto um parceiro.
5. Representar o facto de, na aldeia de Barjas, o parceiro de uma pessoa ter que ser do sexo oposto.

É fácil resolver os primeiros pontos.
Já o último parece mais difícil de representar. Podemos, por exemplo definir uma nova classe comParceiroSexoOposto:

```
Class: comParceiroSexoOposto
    SubClassOf:
        (Homem
         and (temParceiro only Mulher))
        or (Mulher
         and (temParceiro only Homem))
```

E em seguida acrescentar em Barjas que é do tipo (temHabitante only comParceiroSexoOposto).

Encontra no ficheiro `aldeiaBarjas2.owl` o resultado da representação deste diálogo 2 (incluindo tudo o que já vinha do diálogo anterior).

Diálogo 3:

`Tagarela: A parceira do AL é a Laura, mas há outra mulher, a Lia, que é também de Barjas.`
`Pensador: Então a Lia não é parceira do AL.`

Para representar este diálogo, temos que ter presente uma questão muito importante em OWL: Assume-se sempre que a informação pode estar incompleta.
Isso corresponde à adopção de dois pressupostos, que têm um grande impacto no que se consegue inferir:

- Pressuposto do **Mundo Aberto** (*Open World Assumption*): o que não se consegue provar não é considerado falso. Apenas não se sabe se é verdadeiro ou falso.
- Pressuposto de que **os nomes não são únicos** (*No Unique-Names Assumption*): Dois (ou mais) Individuals podem corresponder a (diferentes nomes para) uma mesma instância. Só são considerados distintos se isso for explicitamente indicado.

Para representar este diálogo é fundamental lembrarmo-nos do pressuposto de que os nomes não são únicos: Não basta criarmos dois Individuals Laura e Lia (poderiam ser dois nomes para a mesma entidade).
Temos que indicar explicitamente que Laura é diferente de Lia: tendo uma dessas instâncias seleccionada, no seu Tab Description clicamos em Different Individuals e seleccionamos a outra instância.

Encontra no ficheiro `aldeiaBarjas3.owl` o resultado da representação deste diálogo 3 (incluindo tudo o que já vinha dos diálogos anteriores).

Exercício 2:

Partindo da ontologia em `aldeiaBarjas3.owl`, estenda-a de modo a representar os seguintes diálogos:

1. `Tagarela: Os parceiros dos habitantes de Barjas habitam também em Barjas.`
   `Pensador: Isso quer dizer que Barjas tem pelo menos 3 habitantes.`

2. `Tagarela: Os amigos da aldeia de Barjas são amigos de todos os que habitam em Barjas. Nem todos os habitantes de Barjas são amigos da aldeia de Barjas.`
   `A Laura não é amiga de ninguém.`
   `Pensador: Isso quer dizer que há pelo menos 1 habitante de Barjas que não é amigo da aldeia.`

Exercício 3:

Dada a ontologia anexa `equipas.owl`, explicite o seguinte:

1. A razão para a inferência de que `AvenidasFC` é instância de `EquipaMista`.

2. A razão para a não inferência de que `AvenidasFC` é instância de `EquipaNaoIndividual`.