
Tutorial: Protégé-OWL (Universidade)

José Carlos Ramalho

jcr@di.uminho.pt

Dezembro 2011

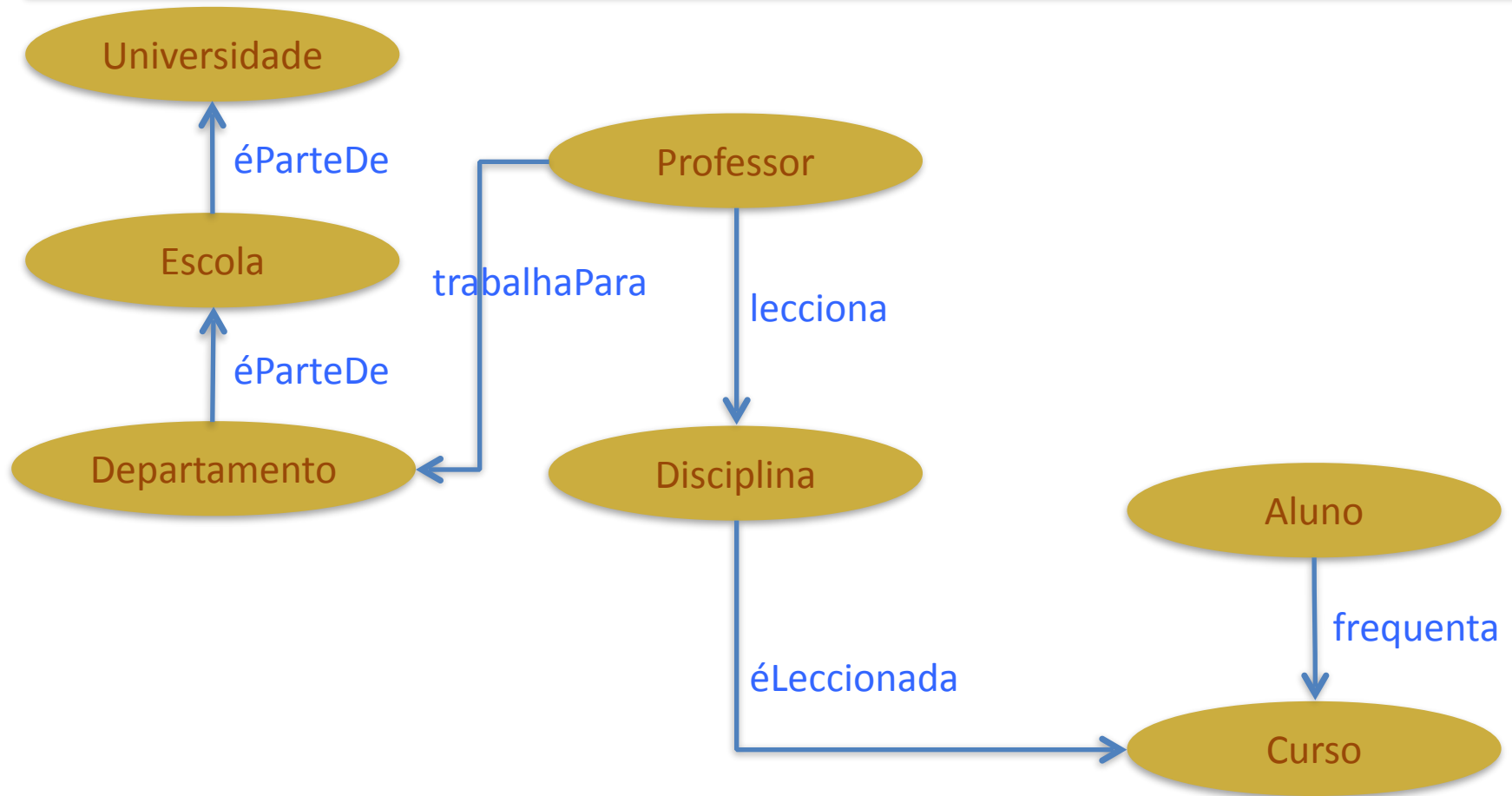
Conteúdo

- Condições necessárias e suficientes:
 - Classes primitivas e classes definidas;
- Classificação Automática;
- Restrições universais;
- Axiomas de fecho;
- Partições e axiomas de cobertura;
- Restrições de cardinalidade;
- Restrições de cardinalidade qualificadas;

Criação de Ontologias no Protégé

- Desenho conceptual da ontologia;
- Especificação das classes;
- Especificação das propriedades;
- Especificação/criação de indivíduos;
- Especificação de restrições.

Novo caso de estudo: MIECOM

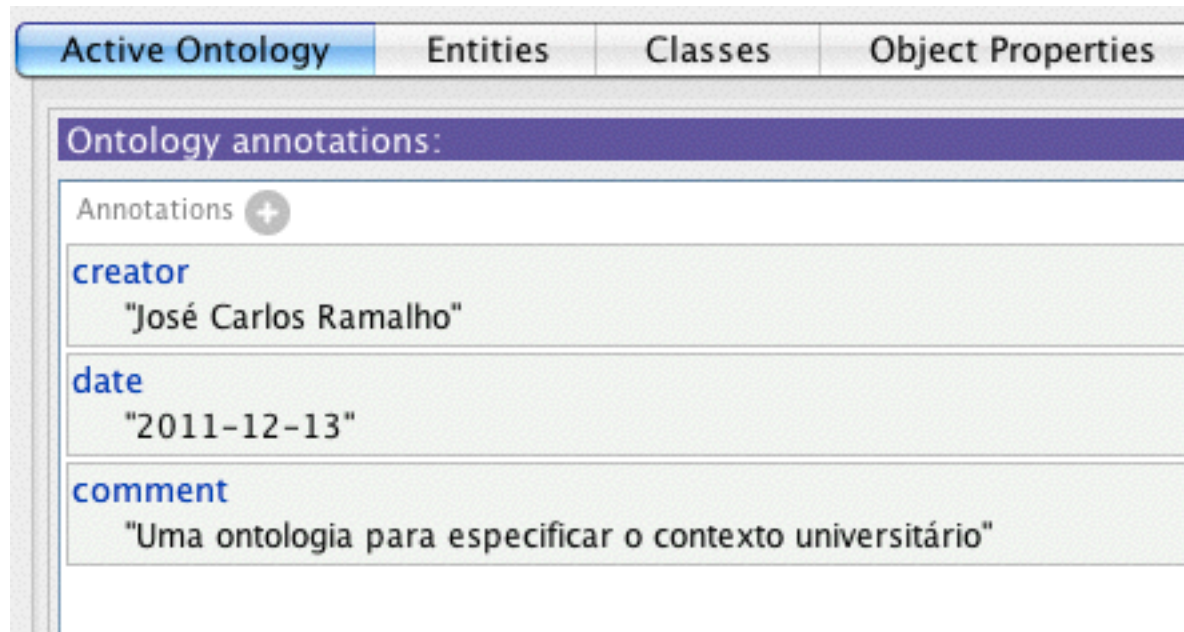


Criar as classes

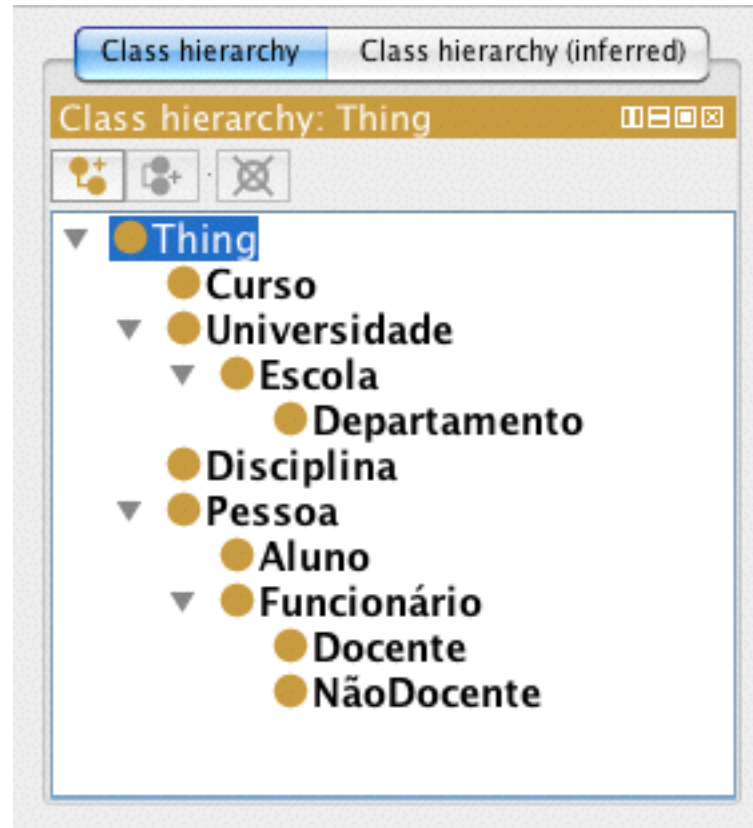
- Universidade
- Escola
- Departamento
- Funcionário: Docente, NãoDocente
- Aluno
- Curso
- Disciplina

Documentação da Ontologia

- Metainformação sobre o que se está a fazer



Especificação das Classes

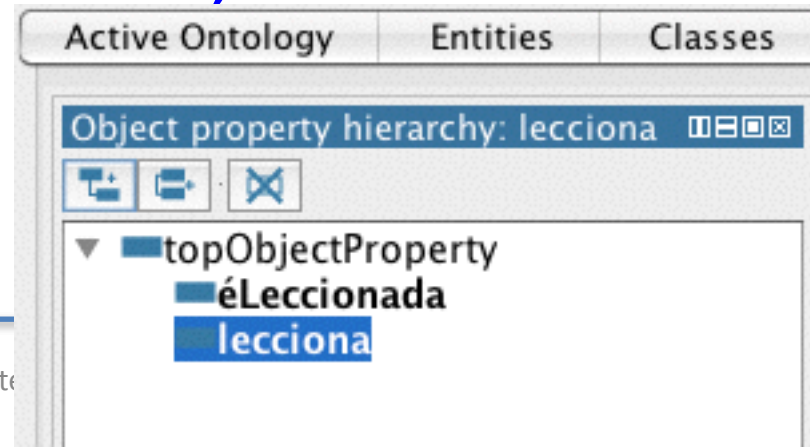


Mais classes...

- Acrescentar filhos a Curso:
 - 1ºciclo, 2ºciclo e 3ºciclo;
- Acrescentar classe Publicação com subclasses:
 - Tese, Livro, ArtigoRevista, ArtigoConferência, ...
- Acrescentar a classe Competência

Especificação de Propriedades: “Object Properties”

- Relacionam dois conceitos
- Criar:
 - lecciona / éLeccionada;
 - frequenta / éFrequentado;
 - fazPartePlanoCurricular;
 - trabalhaPara / temTrabalhador;



Especificação de Propriedades: inversas

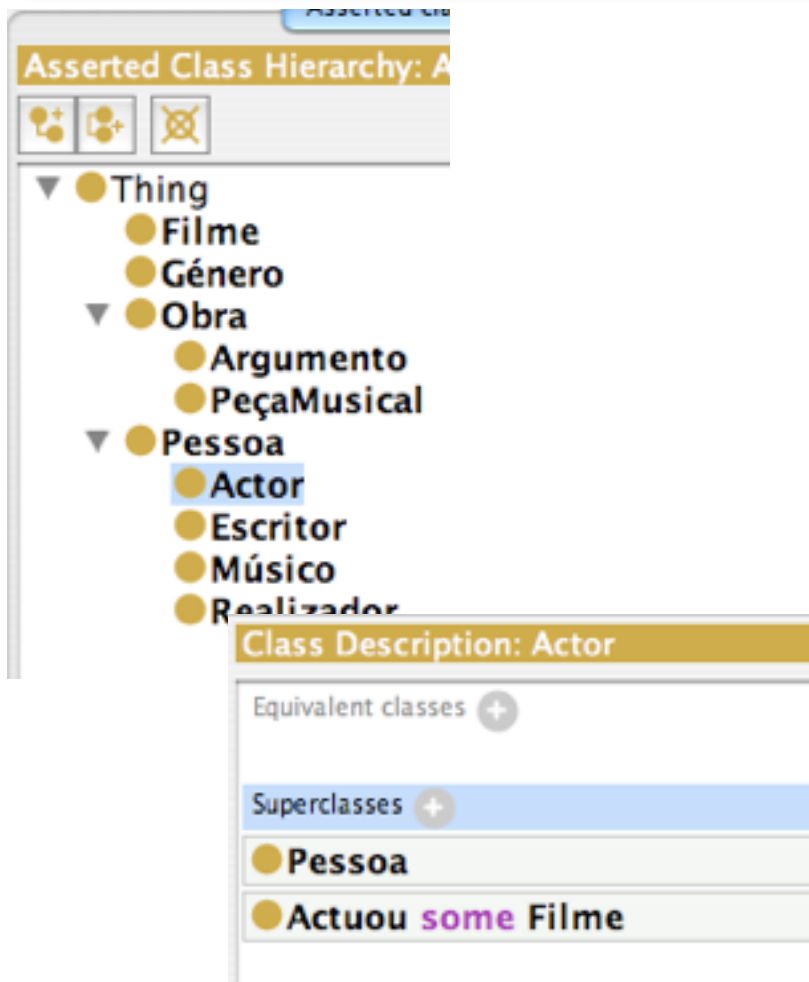
Indicar que estes pares correspondem a propriedades inversas:

- lecciona / éLeccionada;
- frequenta / éFrequentado;
- trabalhaPara / temTrabalhador;

Vamos criar alguns indivíduos

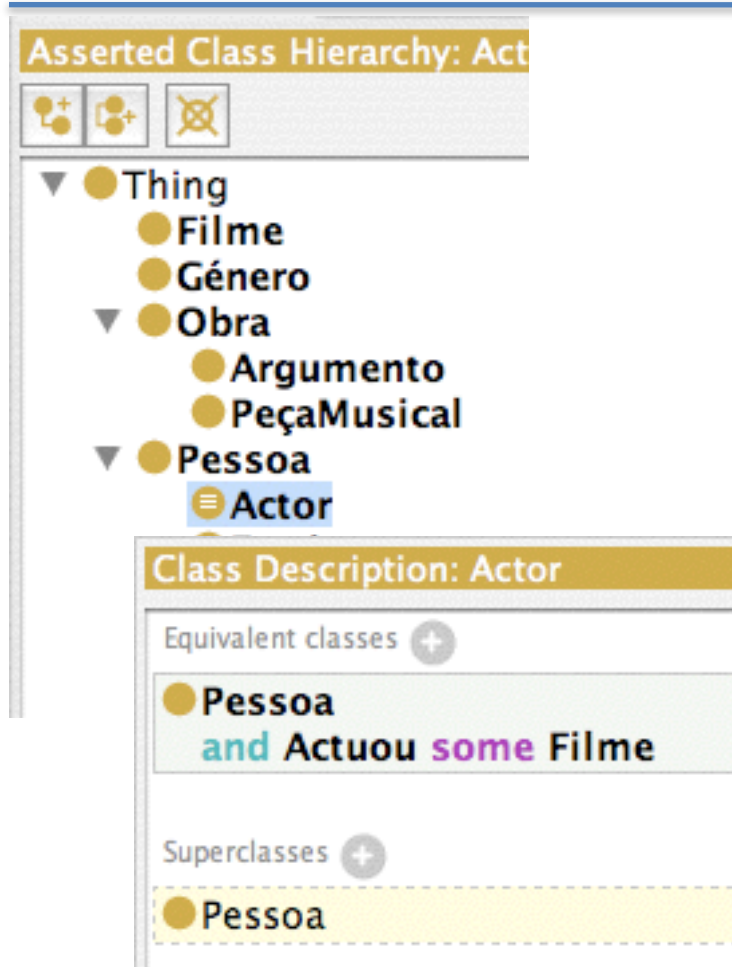
- Universidade: Minho;
- Pessoas: eu, alguns de vocês, ...
 - relacioná-las com as propriedades criadas;
- Verificar se o sistema consegue inferir que algumas pessoas são da classe **Docente** e outras da classe **Aluno**.

Classe primitiva: Actor



- Classe primitiva: condições necessárias;
- Para ser Actor é necessário que se tenha actuado nalgum filme;
- Se houver um indivíduo da classe Pessoa que tenha actuado num Filme o sistema não pode concluir que é um Actor.

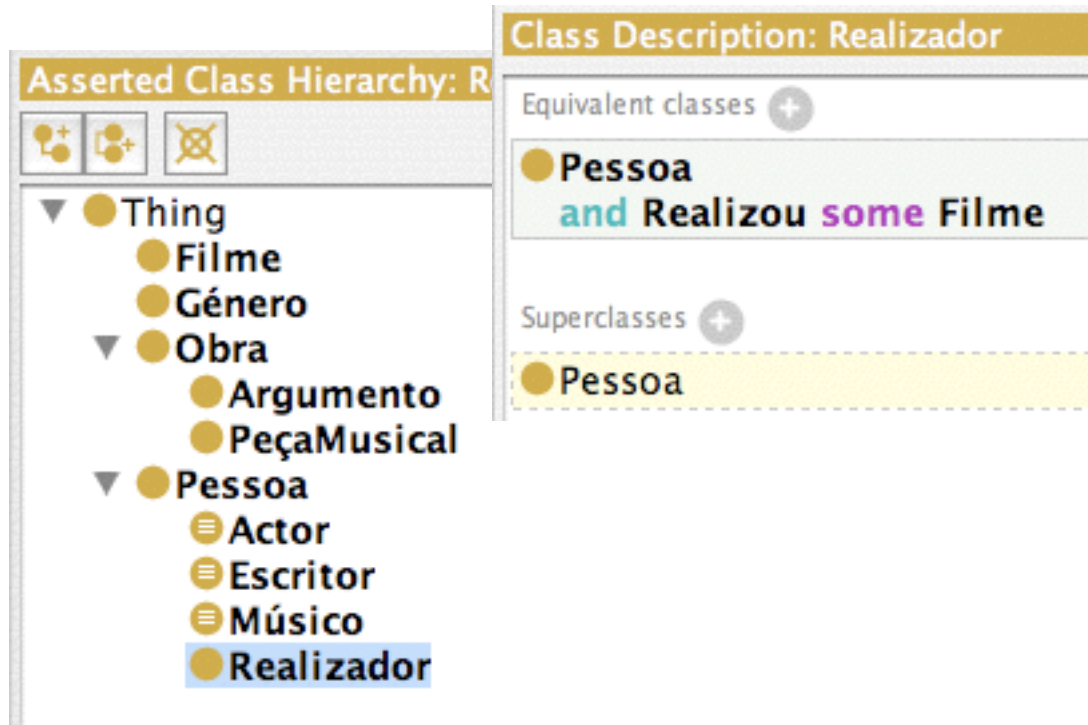
Transformar Actor numa classe definida



- Classe definida: as condições são agora necessárias e suficientes;
- Para ser Actor é agora necessário e suficiente que se tenha actuado nalgum filme;
- Se houver um indivíduo da classe Pessoa que tenha actuado num Filme o sistema pode agora concluir que é um Actor.

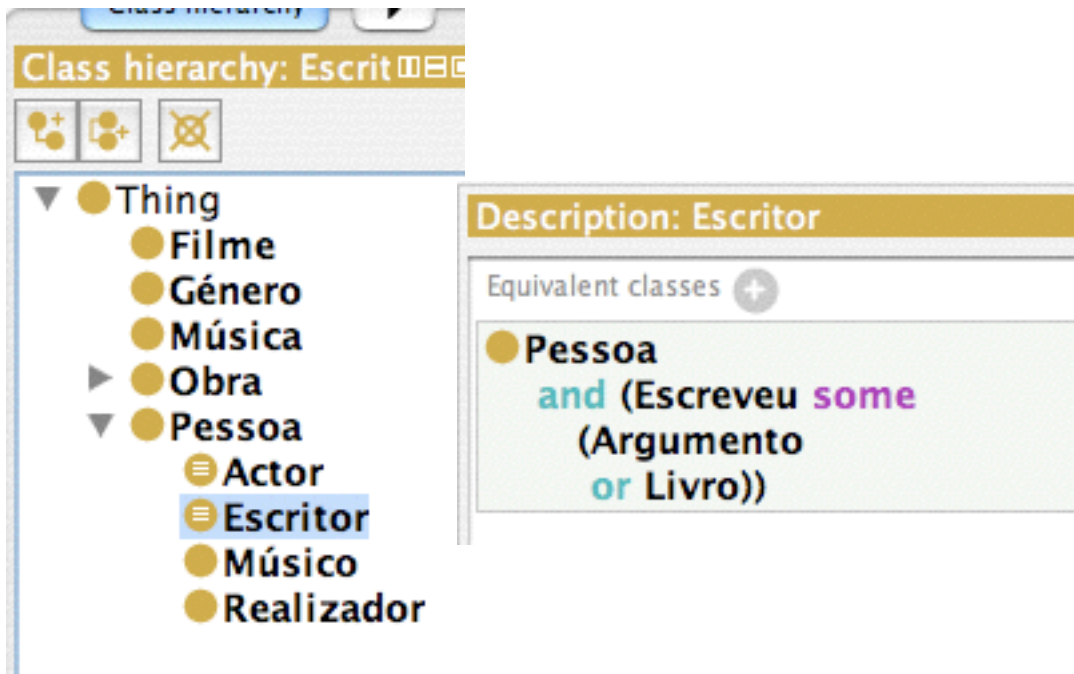
Definição de classes

- Definir a classe: Realizador



Definição de classes

- Definir a classe: Escritor



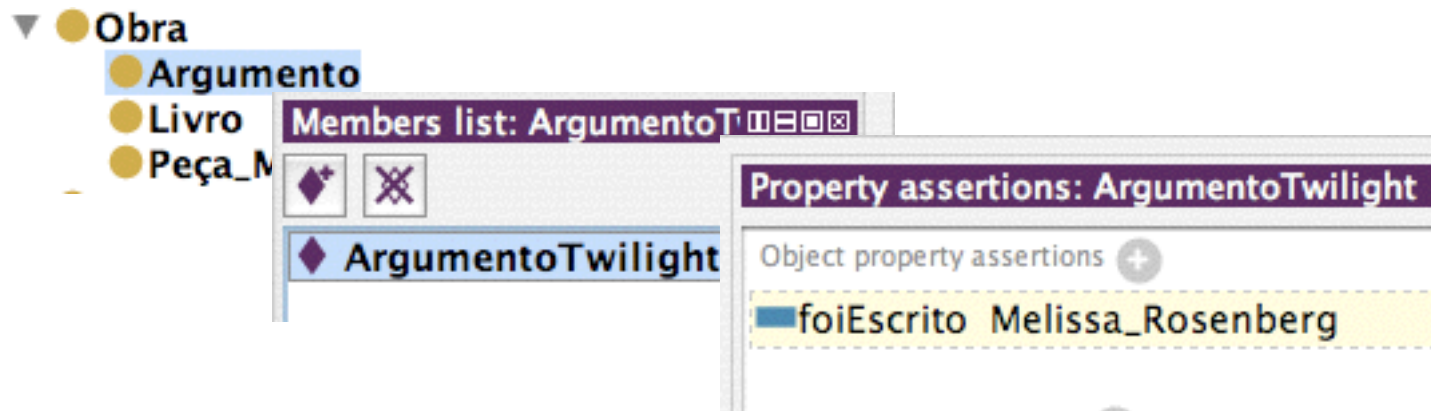
Definição de classes

- Definir a classe: Músico



Mais indivíduos...

- Indivíduos: Catherine Hardwicke (**Pessoa que Realizou Twilight**), Melissa Rosenberg (**Pessoa que Escreveu o ArgumentoTwilight**);
- Usar o “Reasoner” para ver se as propriedades estão a ser inferidas: Argumento



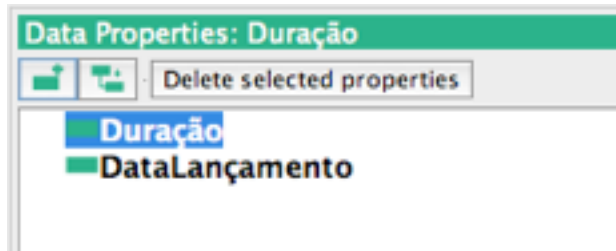
Novas classes e propriedades

- Propriedades: temArgumento, temPaísOrigem (membro da classe País) e temLíngua (membro da classe Língua);
- Mais indivíduos: Língua = {Inglês, Francês, Português}, País = {Estados Unidos, Inglaterra, França , Portugal}.

“Data properties”: características

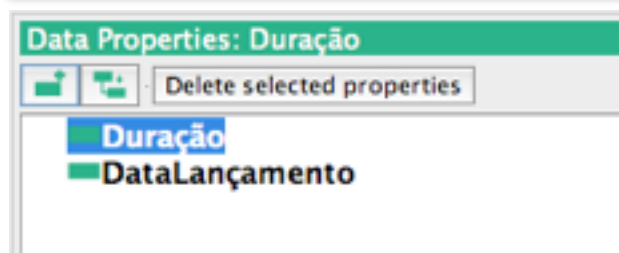
- Filme
 - tem Data (date);
 - e Duração (integer = min.).

“Data Properties”

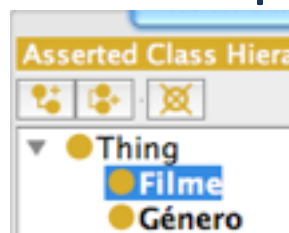


- Depois de as criarmos temos de as associar às respectivas classes para as podermos usar.

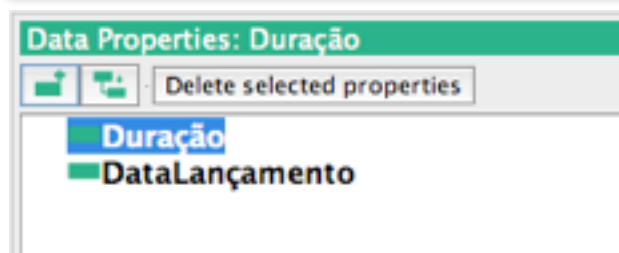
“Data Properties”



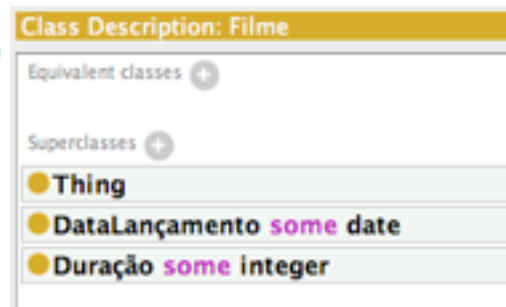
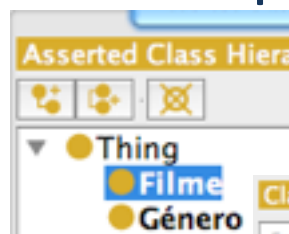
- Depois de as criarmos temos de as associar às respectivas classes para as podermos usar.



“Data Properties”



- Depois de as criarmos temos de as associar às respectivas classes para as podermos usar.

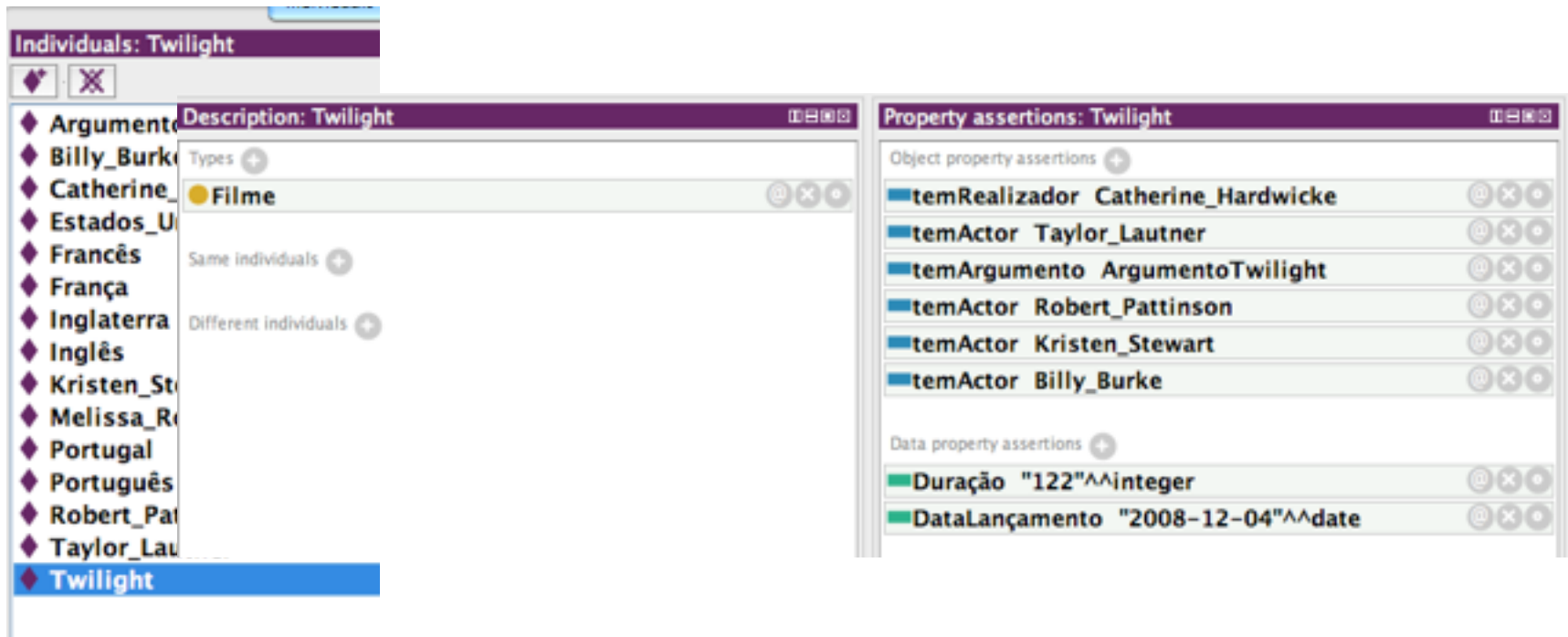


“Data Properties”: utilização

The screenshot displays a Semantic Web editor interface with three main panels:

- Left Panel (Individuals: Twilight):** A list of individuals including Argumento, Billy_Burke, Catherine, Estados_U, Francês, França, Inglaterra, Inglês, Kristen_St, Melissa_R, Portugal, Português, Robert_Pat, Taylor_Lau, and **Twilight** (highlighted in blue).
- Middle Panel (Description: Twilight):** Shows the type **Filme** (indicated by a yellow circle icon) and options for 'Same individuals' and 'Different individuals' (both with plus icons).
- Right Panel (Property assertions: Twilight):** Displays two sections of assertions:
 - Object property assertions:** A list of five assertions, each with a blue bar icon and a plus icon for expansion:
 - temRealizador Catherine_Hardwicke
 - temActor Taylor_Lautner
 - temArgumento ArgumentoTwilight
 - temActor Robert_Pattinson
 - temActor Kristen_Stewart
 - temActor Billy_Burke
 - Data property assertions:** A list of two assertions, each with a green bar icon and a plus icon for expansion:
 - Duração "122"^^integer
 - DataLançamento "2008-12-04"^^date

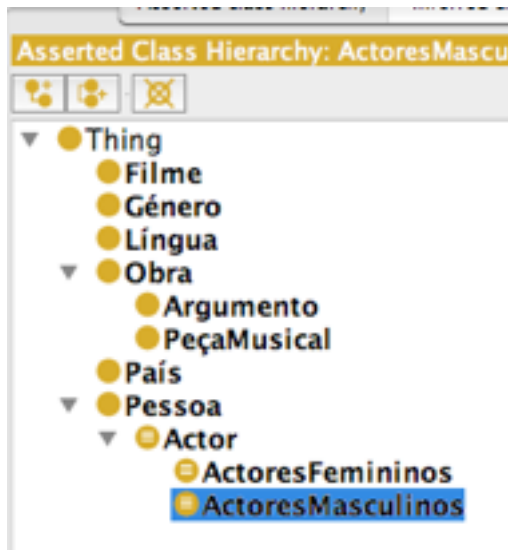
“Data Properties”: utilização



- Crie mais uma propriedade **temSexo** e preencha-a para todos os indivíduos da Classe **Pessoa**.

Restrições sobre valores

- Crie duas subclasses da classe Actor: ActoresMasculos e ActoresFemininos;
- $\text{ActorMascu} = \text{Actor} \wedge \text{temSexo "M"}$



Restrições sobre valores

- Crie duas subclasses da classe Actor: ActoresMasculos e ActoresFemininos;
- $\text{ActorMasculino} = \text{Actor} \wedge \text{temSexo "M"}$

The screenshot displays a software interface with two main panels. The left panel, titled "Asserted Class Hierarchy: ActoresMas", shows a tree structure of classes. The hierarchy starts with "Thing", which has children "Filme", "Género", "Língua", and "Obra". "Obra" has children "Argumento" and "PeçaMusical". "País" is also a child of "Thing". "Pessoa" is a child of "Obra", and "Actor" is a child of "Pessoa". Under "Actor", there are two subclasses: "ActoresFemininos" and "ActoresMasculos". The right panel, titled "Class Description: ActoresFemininos", provides details about the "ActoresFemininos" class. It lists "Equivalent classes" as "Actor and temSexo value 'F'", "Superclasses" as "Actor", "Inherited anonymous classes" as "Pessoa and Actuou some Filme", "Members" as "Kristen_Stewart", and "Disjoint classes" as "ActoresMasculos".

Asserted Class Hierarchy: ActoresMas

- Thing
 - Filme
 - Género
 - Língua
 - Obra
 - Argumento
 - PeçaMusical
 - País
 - Pessoa
 - Actor
 - ActoresFemininos
 - ActoresMasculos

Class Description: ActoresFemininos

- Equivalent classes
 - Actor and temSexo value "F"
- Superclasses
 - Actor
- Inherited anonymous classes
 - Pessoa and Actuou some Filme
- Members
 - Kristen_Stewart
- Disjoint classes
 - ActoresMasculos

Cobertura

- Vamos adicionar um novo actor ao filme “Twilight” sem indicar qual o seu sexo;
- Use o Calculador para inferir a ontologia e veja como foi classificado o novo indivíduo.

Cobertura

Class Description: Actor	Class Description: ActoresMasculinos
Equivalent classes +	Equivalent classes +
● Pessoa and Actuou some Filme	● Actor and temSexo value "M"
Superclasses +	Superclasses +
● Pessoa	● Actor
Inherited anonymous classes	Inherited anonymous classes
Members +	Members +
◆ Billy_Burke ◆ Gil_Birmingham ◆ Kristen_Stewart ◆ Robert_Pattinson ◆ Taylor_Lautner	◆ Billy_Burke ◆ Robert_Pattinson ◆ Taylor_Lautner
	Disjoint classes +
	● ActoresFemininos

Cobertura

Class Description: Actor	Class Description: ActoresMasculinos
Equivalent classes +	Equivalent classes +
● Pessoa and Actuou some Filme	● Actor and temSexo value "M"
Superclasses +	Superclasses +
● Pessoa	● Actor
Inherited anonymous classes	Inherited anonymous classes
Members +	Members +
◆ Billy_Burke ◆ Gil_Birmingham ◆ Kristen_Stewart ◆ Robert_Pattinson ◆ Taylor_Lautner	◆ Billy_Burke ◆ Robert_Pattinson ◆ Taylor_Lautner
	Disjoint classes +
	● ActoresFemininos

Se quisermos que o Calculador reporte esta situação como anómala temos de cobrir a classe **Actor** com as suas duas subclasses.

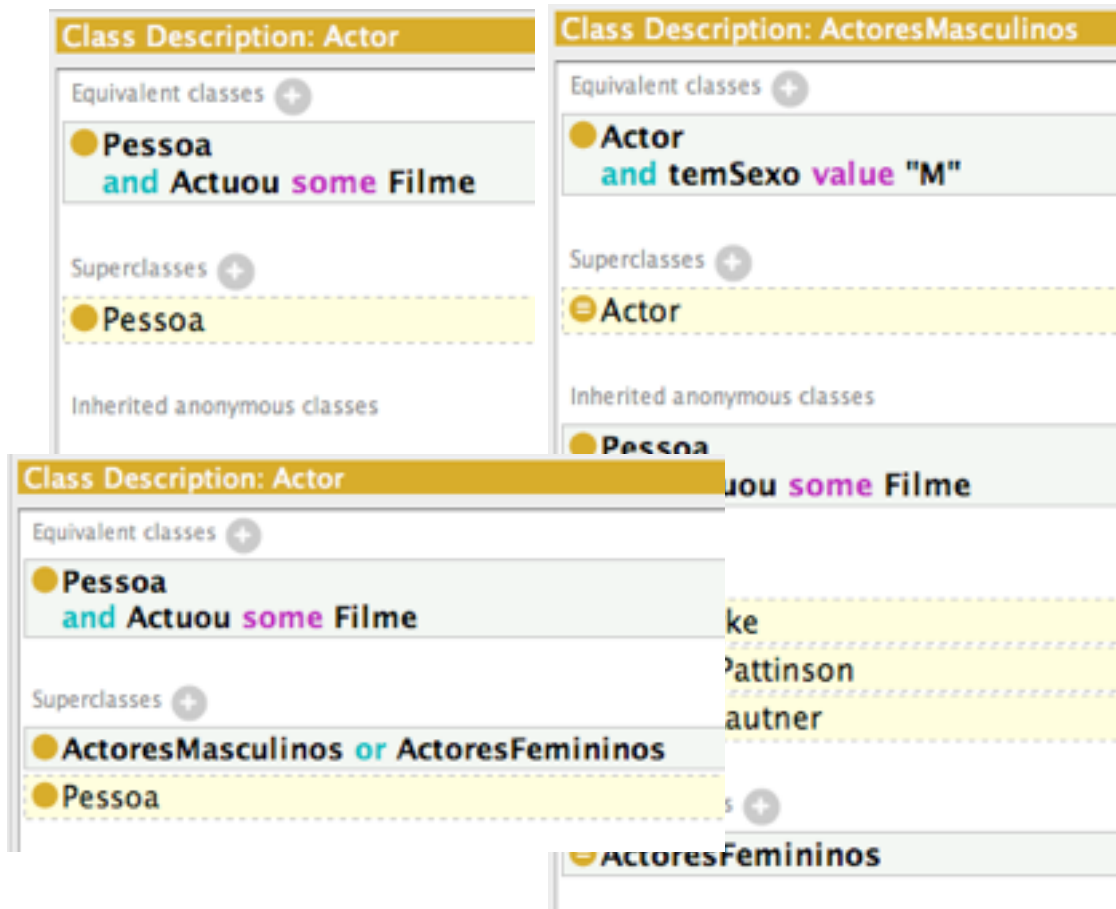
Cobertura

Class Description: Actor	Class Description: ActoresMasculos
Equivalent classes +	Equivalent classes +
● Pessoa and Actuou some Filme	● Actor and temSexo value "M"
Superclasses +	Superclasses +
● Pessoa	● Actor
Inherited anonymous classes	Inherited anonymous classes
Members +	Members +
◆ Billy_Burke ◆ Gil_Birmingham ◆ Kristen_Stewart ◆ Robert_Pattinson ◆ Taylor_Lautner	◆ Billy_Burke ◆ Robert_Pattinson ◆ Taylor_Lautner
	Disjoint classes +
	● ActoresFemininos

Se quisermos que o Calculador reporte esta situação como anómala temos de cobrir a classe **Actor** com as suas duas subclasses.

Há várias maneiras de resolver o problema: restrição existencial sobre a propriedade **temSexo**, ou, pedir ao sistema que crie um axioma de cobertura.

Cobertura



Se quisermos que o Calculador reporte esta situação como anómala temos de cobrir a classe **Actor** com as suas duas subclasses.

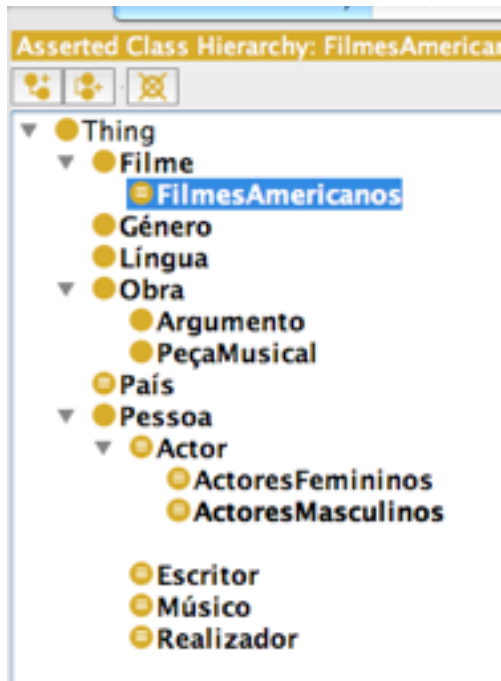
Há várias maneiras de resolver o problema: restrição existencial sobre a propriedade **temSexo**, ou, pedir ao sistema que crie um axioma de cobertura.

Classes definidas por enumeração



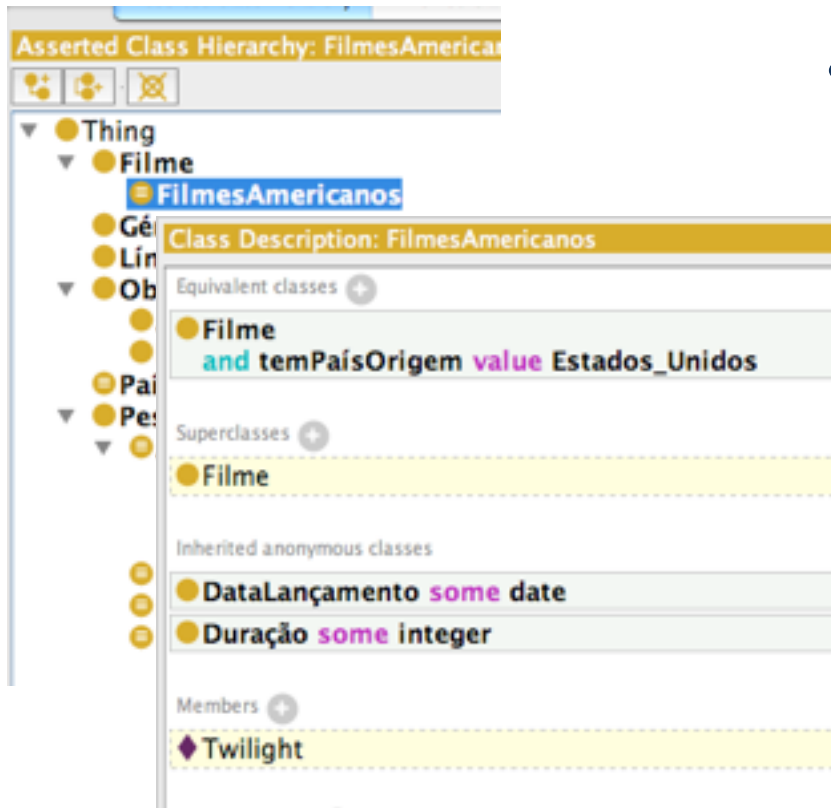
- Vamos alterar a definição de classe de País para uma enumeração;
- Adicionar uma classe equivalente com o seguinte conteúdo: {Estados_Unidos, França, Inglaterra, Portugal}

Classes definidas por enumeração



- Crie uma subclasse de **Filme** chamada **FilmesAmericanos** com uma restrição sobre a propriedade **temPaísOrigem**.

Classes definidas por enumeração



- Crie uma subclasse de **Filme** chamada **FilmesAmericanos** com uma restrição sobre a propriedade **temPaísOrigem**.

Completando a ontologia...

- Defina a classe **Género** como uma classe enumerada com os valores: Drama, Comédia, Thriller, Ficção, Terror, Infantil, Romance, Acção e Aventura;
- Instanciar o **Género** para o **Filme** Twilight: Drama, Romance e Thriller;
- Crie duas subclasses de **Filme**: **FilmesDramáticos** e **FilmesRomânticos**.

Exercício: novo filme

- Posicione-se na página do IMDB do filme Madagascar2;
- Crie o indivíduo **Madagascar2** com os seguintes campos: **Duração**, **DataLançamento**, **temTítulo**;
- Crie a classe **Personagem** com os indivíduos: **Alex**, **Gloria**, **Marty**, **Melman**;
- Crie as seguintes Propriedades: **éPersonagem** / **temPersonagem** e **representa**(**Actor**, **Personagem**);
- Acrescente os seguintes actores e relacione-os com os respectivos personagens: **Ben Stiller** (**Alex**), **Chris Rock** (**Marty**);
- Acrescente os géneros **Infantil** e **Aventura** a este filme;
- Crie as classes **FilmesInfantis** e **FilmesAventura**.

Restrições de Cardinalidade

- Vamos criar a classe **FilmesInteressantes** como aqueles que pertencem a mais de 2 géneros.



Restrições com intervalos

- Vamos criar a classe **LongasMetragens** que deverá conter os filmes com **Duração** superior a 60 minutos.

Restrições com intervalos

- Vamos criar a classe **LongasMetragens** que deverá conter os filmes com **Duração** superior a 60 minutos.

The screenshot displays a software interface for class management, divided into two main panels. The left panel, titled 'Asserted Class Hierarchy: LongaMetragem', shows a tree structure of classes. The root is 'Thing', which branches into 'Personagem', 'Filme', 'Gênero', 'Língua', 'Obra', 'País', and 'Pessoa'. Under 'Filme', there is a sub-tree including 'FilmesInteressantes', 'LongaMetragem' (highlighted with a blue selection box), 'FilmesInfantis', 'FilmesRomânticos', 'FilmesDramáticos', and 'FilmesAmericanos'. Under 'Pessoa', there is a sub-tree including 'Actor', 'ActoresFemininos', 'ActoresMasculinos', 'Escritor', 'Músico', and 'Realizador'. The right panel, titled 'Class Description: LongaMetragem', provides details for the selected class. It includes sections for 'Equivalent classes' (showing 'Filme that Duração some integer[>60]'), 'Superclasses' (showing 'Filme'), 'Inherited anonymous classes' (showing 'DataLançamento some date' and 'Duração some integer'), and 'Members' (showing 'Madagascar2' and 'Twilight').

Múltiplas Definições

