

# Editing e ragionamento con Protégé

Strumenti di editing

Strumenti per il ragionamento automatico

# Ragionamento su classi

La gerarchia delle classi  
asserita mediante gli  
assiomi di sottoclasse  
può essere diversa da  
quella inferita

Il reasoner può inferire  
delle relazioni di  
*sussunzione* a partire  
dalla descrizione delle  
classi

The screenshot shows a software interface with two main panels. The left panel, titled 'Class hierarchy: Genitore', displays a tree structure: 'Thing' (expanded) contains 'Persona' (expanded), which contains 'Genitore\_di\_due' (expanded), which contains 'Genitore'. The 'Genitore' class is highlighted with a blue selection bar. The right panel is divided into two sections: 'Annotations: Genitore' and 'Description: Genitore'. The 'Annotations' section is currently empty. The 'Description' section shows two properties: 'Equivalent To' with a value of 'haFiglio min 1 Persona' and 'SubClass Of' with a value of 'Persona'.

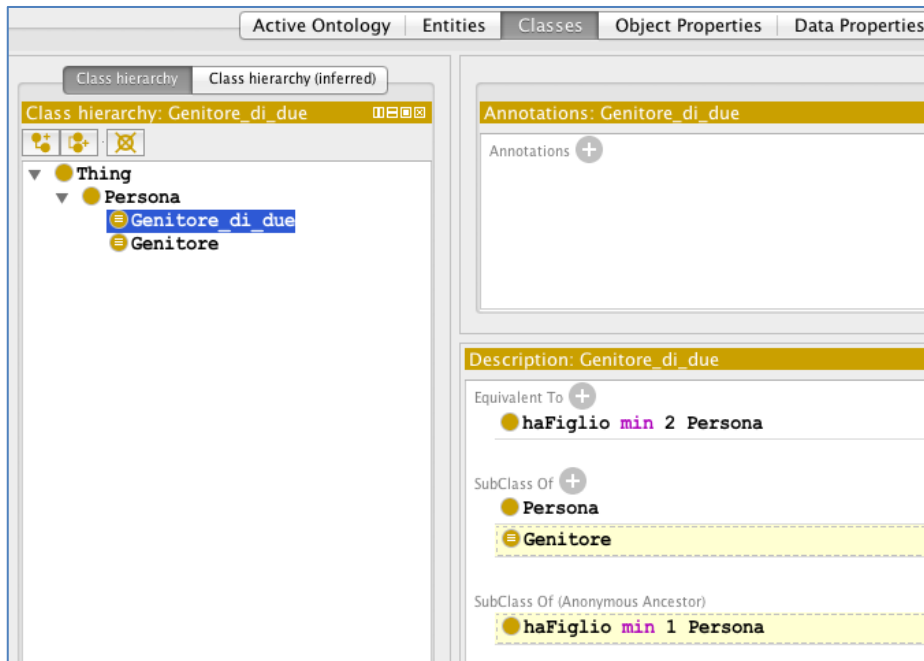
La proprietà  
haFiglio ha come  
dominio Persona e  
come range Persona

La classe Genitore è  
vincolata a avere  
almeno un figlio

The screenshot shows the same software interface as above, but now the left panel is titled 'Class hierarchy: Genitore\_di\_due'. The tree structure is: 'Thing' (expanded) contains 'Persona' (expanded), which contains 'Genitore\_di\_due' (expanded) and 'Genitore'. The 'Genitore\_di\_due' class is highlighted with a blue selection bar. The right panel is divided into two sections: 'Annotations: Genitore\_di\_due' and 'Description: Genitore\_di\_due'. The 'Annotations' section is currently empty. The 'Description' section shows two properties: 'Equivalent To' with a value of 'haFiglio min 2 Persona' and 'SubClass Of' with a value of 'Persona'.

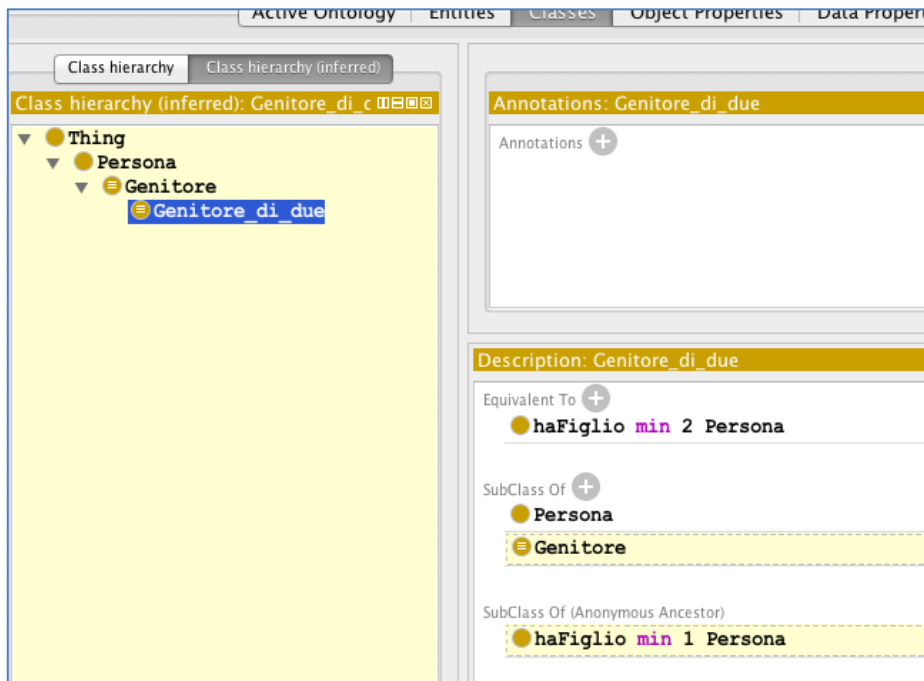
La classe  
Genitore\_di\_due è  
vincolata a averne  
almeno due

Le due classi sono  
definite come  
sottoclassi di Persona.



Il reasoner inferisce che la classe Genitore\_di\_due è sottoclasse di Genitore

Nel pannello delle classi, la gerarchia inferita mostra che la classe Genitore\_di\_due è stata **spostata** sotto la classe Genitore



Il reasoner ha inferito la relazione di sussunzione

The background of the slide features several thin, curved lines in shades of gray, some solid and some dashed, creating a sense of motion or flow. On the left side, there is a blue rectangular area with a white border and a small white triangle pointing downwards at the bottom center, resembling a speech bubble or a callout box.

## Inserimento di individui

- Gli individui vengono inseriti direttamente in una classe
- Dopo averli inseriti è possibile predicarne le caratteristiche, cioè specificarne le proprietà che li mettono in relazione con altri individui o con un dato (class e property assertions)
- Il ragionamento può eventualmente ricollocare un certo individuo in un'altra classe

# Relazione classi - individui



Aggiunta individui  
(nella classe  
selezionata)

# Inserimento Individui

The screenshot shows the Protégé ontology editor interface. The top navigation bar includes tabs for Object Properties, Data Properties, Annotation Properties, Individuals, DL Query, Ontology Differences, and SPARQL Query. The 'Individuals' tab is active. On the left, the 'Class hierarchy' panel shows a tree structure with 'Ristorante' selected. The 'Members list' panel is empty. The 'Annotations' panel shows a search bar and a plus icon. The 'Description' panel shows 'Types' and 'Same Individual As' sections. The 'Property assertions' panel shows sections for Object property assertions, Data property assertions, Negative object property assertions, and Negative data property assertions. A dialog box titled 'Create a new OWLNamedIndividual' is open in the foreground, with the following fields:

- Name: ristorante da Beppe
- IRI: g/rossana/ontologies/2014/11/untitled-ontology-48#ristorante\_da\_Beppe
- New entity options... button
- Buttons: Annulla, OK

At the bottom of the window, there is a status bar with the text: To use the reasoner click Reasoner->Start reasoner ☒ Show Inferences

# Asserzioni sugli individui

The screenshot displays the Protégé ontology editor interface. The main window shows the 'untitled-ontology-48' with tabs for Object Properties, Data Properties, Annotation Properties, Individuals, DL Query, and Ontology. The 'Class hierarchy' panel on the left shows a tree structure with 'Ristorante' as the root, containing sub-classes like 'CucinaItaliana', 'CucinaOrientale', 'CucinaCinese', and 'CucinaGiapponese'. The 'Members list: ristorante\_da\_Beppe' panel shows a list of individuals, including 'bistecca\_alla\_Fiorentina' and 'ristorante\_da\_Beppe'. The 'Property assertions: ristorante\_da\_Beppe' panel shows a list of property assertions, including 'serveCucina'. A red text box with an arrow points to the 'serveCucina' property, stating: 'Attribuisco all'individuo selezionato una proprietà che lo collega a un altro individuo'. Another red text box with an arrow points to the 'Object property assertions' button, stating: 'Aggiunta asserzioni sull'individuo selezionato'. At the bottom, a footer text reads: 'To use the reasoner click Reasoner->Start reasoner' and 'Show Inferences'.

untitled-ontology-48 (http://www.semanticweb.org/rossana/ontologies/2014/11/untitled-ontology-48) : [http://www.semanticweb.org/rossana/ontologies/2014/11/untitled-ontology-48]

Object Properties | Data Properties | Annotation Properties | Individuals | DL Query | Ontology

Class hierarchy | Class hierarchy (inferred)

Class hierarchy: Ristorante

- Thing
  - Cucina
    - CucinaItaliana
    - CucinaOrientale
      - CucinaCinese
      - CucinaGiapponese
  - Ristorante

Members list: ristorante\_da\_Beppe

- ristorante\_da\_Beppe

Annotations: ristorante\_da\_Beppe

Annotations +

Description: ristorante\_da\_Beppe

Types +

- Ristorante

Property assertions: ristorante\_da\_Beppe

- Object property assertions +
- Data property assertions +
- Negative object property assertions -
- Negative data property assertions +

ristorante\_da\_Beppe

topObjectProperty

- serveCucina

bistecca\_alla\_Fiorentina

ristorante\_da\_Beppe

Attribuisco all'individuo selezionato una proprietà che lo collega a un altro individuo

Aggiunta asserzioni sull'individuo selezionato

Annulla OK

To use the reasoner click Reasoner->Start reasoner ☒ Show Inferences



# Esempio: disgiunzione

- Colloco lo stesso individuo in classi disgiunte
- Il reasoner mi segnala che l'ontologia è inconsistente

The screenshot shows a web-based ontology editor interface. On the left, under the 'Classes' tab, the 'Class hierarchy: Studenti' is displayed. It shows a tree structure starting from 'owl:Thing', with 'StudentiLM' as a child, and 'StudentiLT' as a child of 'StudentiLM'. The 'StudentiLT' class is highlighted. Below the hierarchy, there are icons for adding, deleting, and asserting classes, and a dropdown menu set to 'Asserted'. On the right, the 'Annotations: StudentiLT' and 'Description: StudentiLT' panels are visible. The 'Annotations' panel has a '+' icon. The 'Description' panel has several sections: 'Equivalent To' (+), 'SubClass Of' (+), 'General class axioms' (+), 'SubClass Of (Anonymous Ancestor)', 'Instances' (+) with 'Maria' listed below it, 'Target for Key' (+), and 'Disjoint With' (+) with 'StudentiLM' listed below it.

Dato che le classi StudentiLT e StudentiLM sono disgiunte, non posso collocare lo stesso individuo in entrambe

*Reason for inconsistency:  
individual Maria is forced to  
belong to class “StudentiLT”  
and its complement*

## Inserimento di individui

### Description: Genitore

Equivalent To +

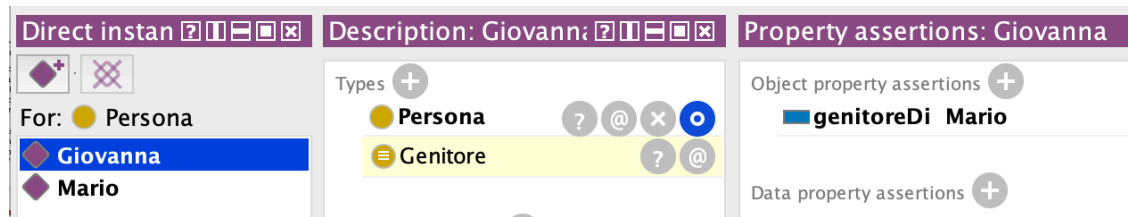
● **genitoreDi** some **Persona**

SubClass Of +

● **Persona**

- Inserendo un individuo con le caratteristiche di una certa classe, l'individuo viene classificato come appartenente alla classe dal *reasoner*
- Anche se è stato collocato manualmente in una classe più generale

# Classificazione automatica

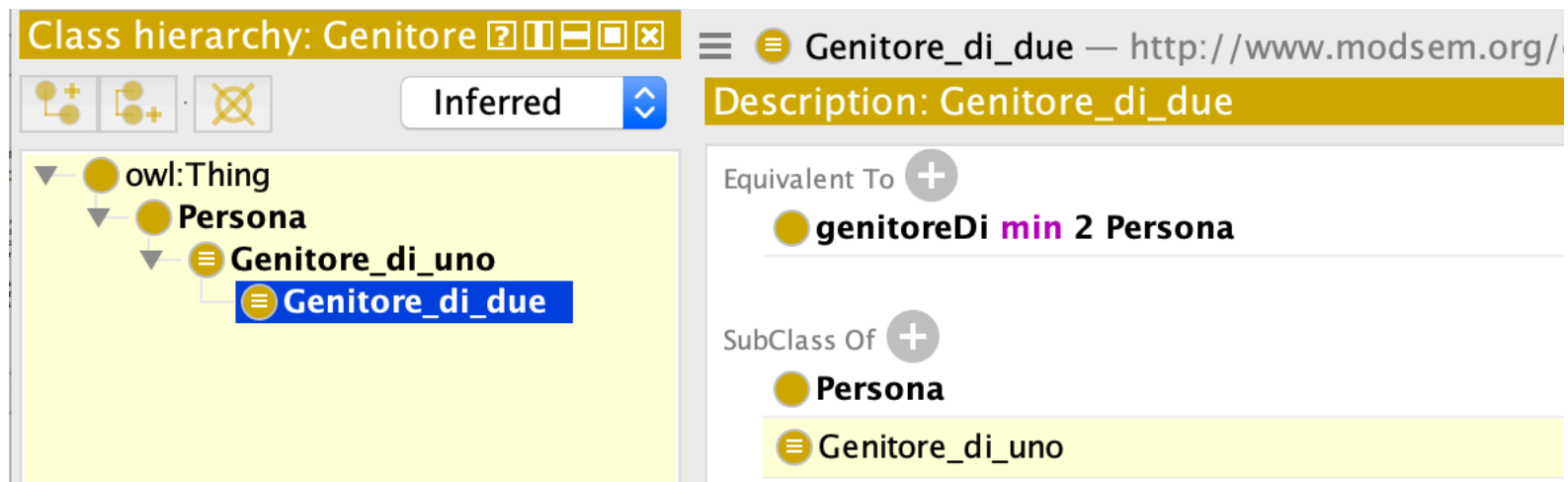


- L'individuo Giovanna è stato spostato nella classe genitore (classe più specifica di Persona in cui era stato collocato)
  - Perché Giovanna “genitoreDi” Mario
- L'inferenza scatta perché la classe Genitore è una classe *definita*, a cui sono associate condizioni necessarie e sufficienti

# Assunzione di mondo aperto

- Le inferenze effettuate dal reasoner possono sembrare poco intuitive in alcuni casi
- La maggior parte dei sistemi di ragionamento automatico segue l'assunzione di mondo chiuso:
  - Ciò che non è rappresentato esplicitamente nel sistema viene assunto falso
  - Provate a salire su un volo se non siete nella lista dei passeggeri
- Il ragionamento sulle ontologie OWL segue invece l'assunzione di mondo aperto:
  - Il fatto che un'informazione non sia rappresentata nel sistema non determina che essa sia assunta falsa.

# Individui e classi: esempio



La classe Genitore\_di\_due è una classe definita su cui vale il vincolo che per i suoi appartenenti la relazione genitoreDi abbia almeno due individui diversi

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

- Classes:** A sidebar on the left showing a hierarchy. Under 'Individuals: Car', there are two entries: 'Carlo' (highlighted in blue) and 'Maria'.
- Description: Carlo:** The central panel shows the description of the individual 'Carlo'. It includes a 'Types' section with a plus icon and a list containing 'Genitore\_di\_due' (preceded by a menu icon). Below this is a 'Same Individual As' section with a plus icon. To the right of the type list are four circular icons: a question mark, an at-sign, a cross, and a circle.
- Property assertions: Carlo:** The right panel shows property assertions for 'Carlo'. It has two sections: 'Object property assertions' (highlighted in blue) and 'Data property assertions'. Under 'Object property assertions', there is a plus icon and a single assertion: 'genitoreDi Maria' (where 'genitoreDi' is preceded by a blue bar icon).

Il fatto che un certo individuo, Carlo, sia `genitoreDi` un solo individuo, Maria, non genera una inconsistenza, anche se Carlo è stato collocato nella classe `Genitore_di_due`

Il ragionatore infatti, applica **l'assunzione di mondo aperto**: non viene asserito che Carlo non abbia altri figli oltre a Maria e quindi l'appartenenza alla classe, se asserita esplicitamente, non è in conflitto con le proprietà dell'individuo Carlo

# Same e different: conseguenze

The screenshot shows a Semantic Web browser interface. On the left, a sidebar lists 'Object properties', 'Classes', and 'Individuals: Carlo, Giuseppe, Lisa'. The main area is titled 'Carlo — http://www.modsem.org/esempi#Carlo'. It has two tabs: 'Description: Carlo' and 'Property assertions: Carlo'. The 'Description' tab shows 'Types' with 'Persona' and 'Genitore\_di\_due'. The 'Property assertions' tab shows 'Object property assertions' with 'genitore Lisa' and 'genitore Giuseppe'. There are also 'Same Individual As' and 'Data property assertions' sections.

Il reasoner inferisce correttamente che Giovanni, in quanto genitore di due Samuele e Carla, è appartenere alla classe Genitore\_di\_due  
Ma solo se Samuele e Carla sono stati specificati esplicitamente come due diversi individui.

Altrimenti (mondo aperto) potrebbero essere lo stesso individuo e quindi il vincolo non sarebbe soddisfatto.

# Spiegazioni (funzione Explain inferences)

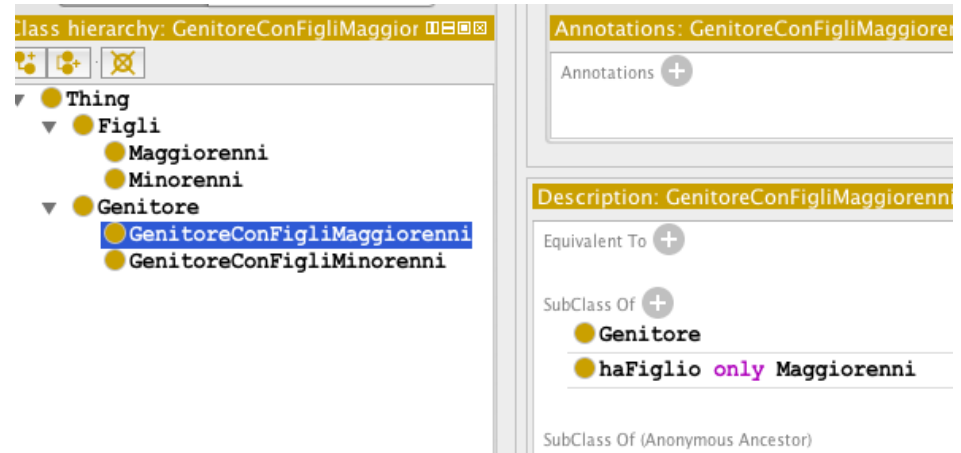
Explanation for: Carlo Type Genitore\_di\_due

1)	Carlo <b>Type</b> Persona	In <b>NO</b> other justifications	?
2)	Carlo genitore Lisa	In <b>ALL</b> other justifications	?
3)	Carlo genitore Giuseppe	In <b>ALL</b> other justifications	?
4)	Giuseppe <b>Type</b> Persona	In <b>ALL</b> other justifications	?
5)	Persona <b>and</b> (genitore <b>min</b> 2 Persona) <b>SubClassOf</b> Genitore_di_due	In <b>ALL</b> other justifications	?
6)	Giuseppe <b>DifferentFrom</b> Lisa	In <b>ALL</b> other justifications	?
7)	Lisa <b>Type</b> Persona	In <b>ALL</b> other justifications	?

- Normalmente sono presenti Explanations diverse, costruite a partire da sequenze di assiomi diverse.
- Le giustificazioni sono diverse ma in parte sovrapposte
- Utile osservare gli assiomi che fanno parte di tutte le giustificazioni.

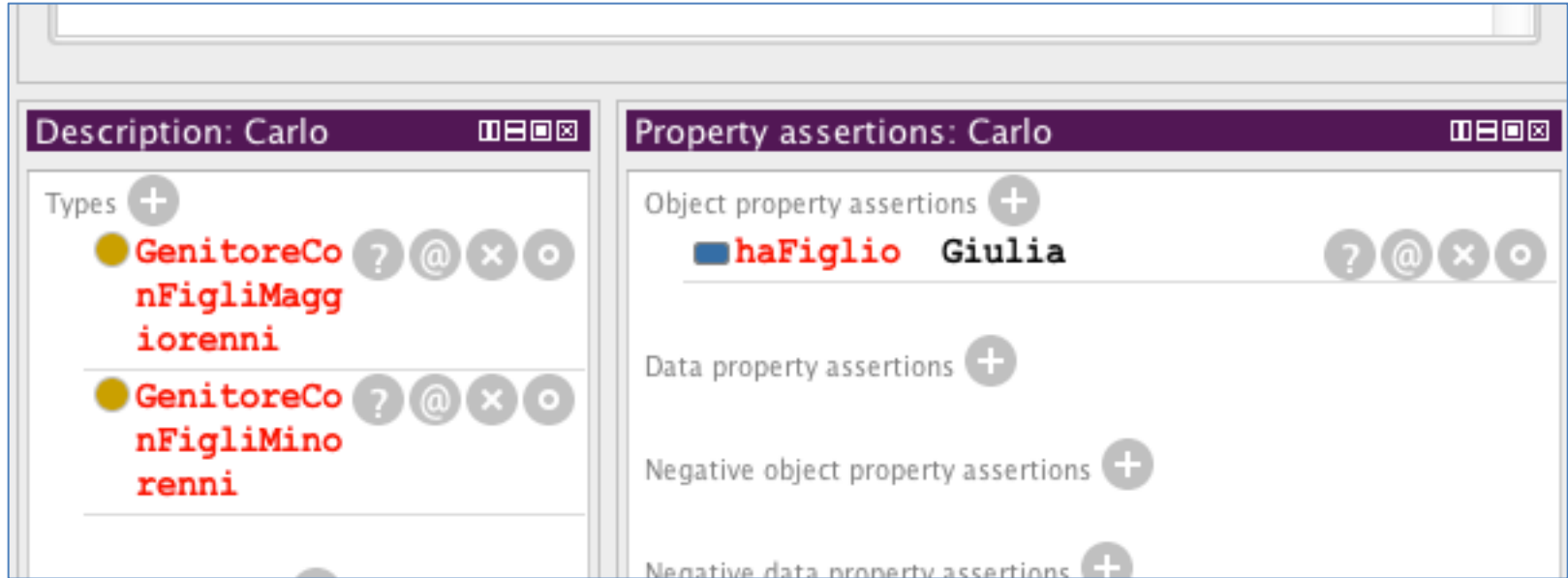


# Quantificatore universale (Only)



- Le classi GenitoreConFigliMaggiorenni e GenitoreConFigliMinorenni non sono disgiunte, ma lo sono le classi Maggiorenni e Minorenni
- Se un individuo viene collocato in entrambe le classi, il reasoner rileva una inconsistenza

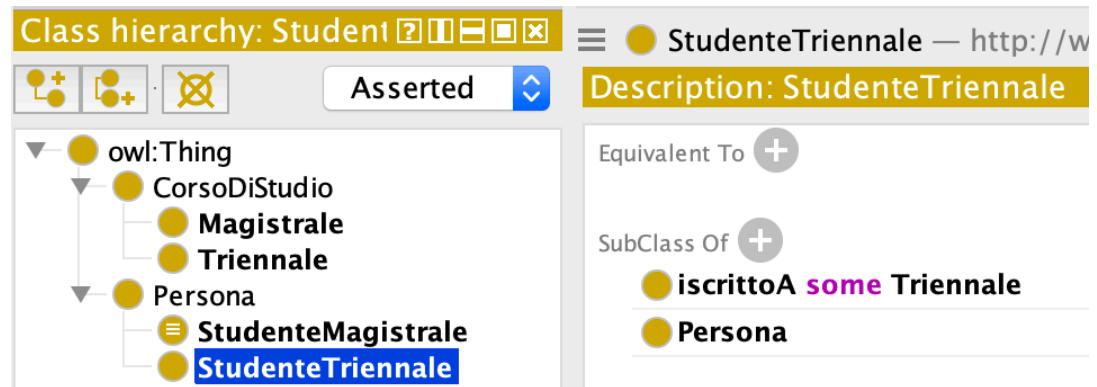
# Esempio



Se Carlo, che haFiglio Giulia (minorenne), viene dichiarato come membro di entrambe le classi, il reasoner rileva un'inconsistenza.

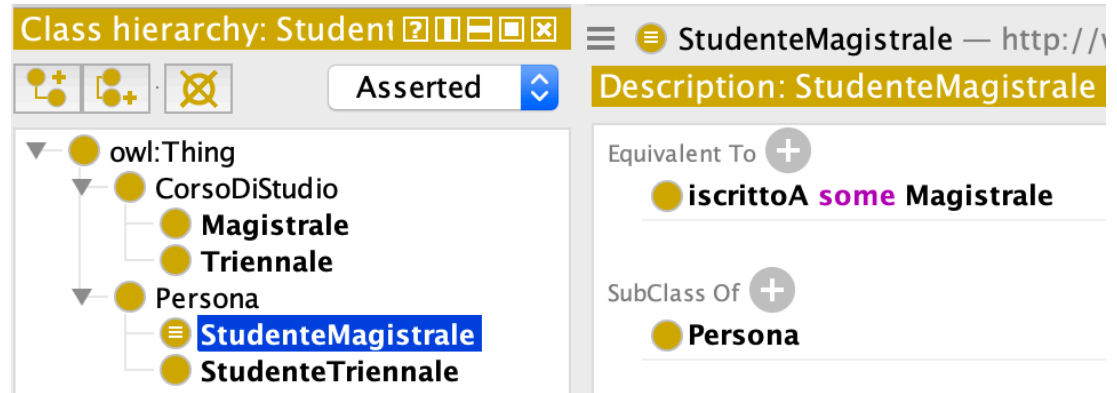
Giulia infatti si troverebbe a appartenere a due classi distinte, che sono state dichiarate come disgiunte.

Classe  
primitiva



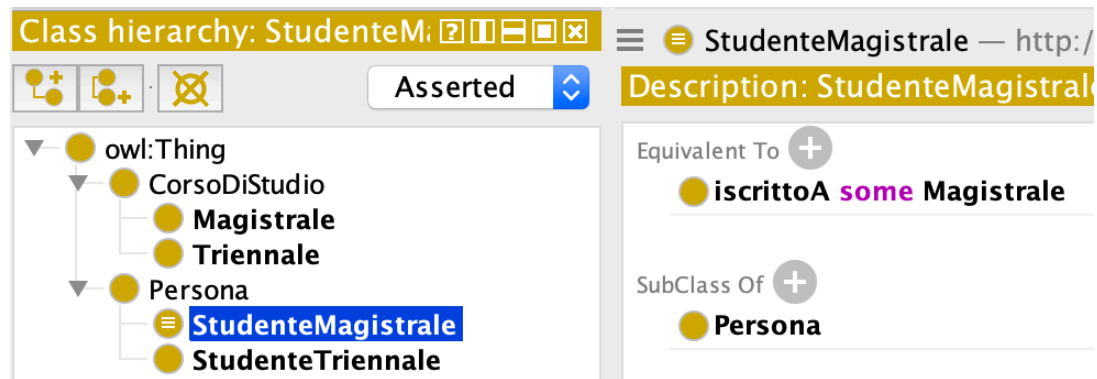
La classe primitiva può avere condizioni necessarie associate ad essa (qui, iscrittoA, vincolato al quantificatore esistenziale), dichiarate tramite lo slot SubClass Of.

Classe definita



La classe **definita** è una classe equivalente a un insieme di condizioni necessarie e sufficienti, specificate mediante **Equivalent To**

## Classi primitive e definite



- Le classi primitive (StudenteTriennale nell'esempio) hanno solo condizioni necessarie associate
- Le classi definite (Studente Magistrale) hanno condizioni necessarie e sufficienti associate → permettono al reasoner di inferire l'appartenenza alla classe di individui che hanno le caratteristiche elencate come condizioni necessarie e sufficienti della classe.

## Esempio

- Se un individuo che ha anche figli non adolescenti (e questi non possono essere assunti tali!) viene descritto come “genitore di figli solo adolescenti” il reasoner rileva un’inconsistenza

SubClass Of   
 **ha\_figli *only* Figli\_adolescenti**

Attenzione a  
quell'only!

- Only **non** significa almeno uno
- (Dal Primer:) “There is one particular **misconception** concerning the universal role restriction. As an example, consider the happiness axiom:” (una persona felice è una persona che ha solo figli felici):  
  
ObjectAllValuesFrom( :hasChild :HappyPerson )
- “The intuitive reading suggests that in order to be happy, a person must have *at least* one happy child. Yet, this is not the case: any individual that is not a “starting point” of the property hasChild is a class member of any class defined by universal quantification over hasChild.”

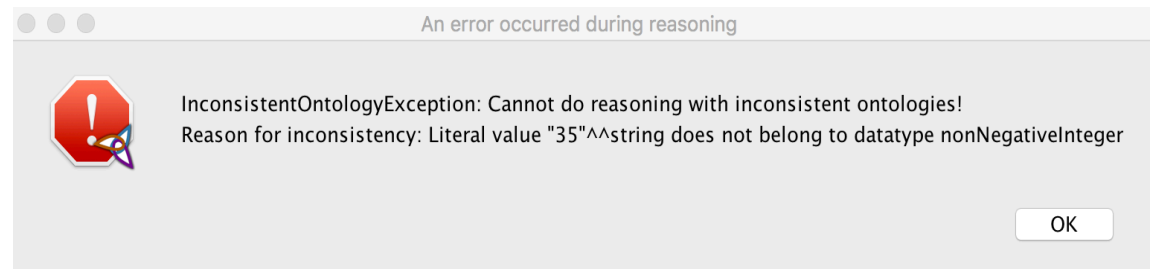
## Ragionamento su data properties

- I valori assegnati a un individuo per una certa data property devono appartenere al data type specificato come range
- Se il datatype è diverso da quello previsto, il reasoner rileva un'inconsistenza



# Esempio

- La proprietà *età* ha come range `xsd:nonNegativeInteger`
- Se la proprietà ha un valore appartenente a un datatype diverso (es. `xsd:string`), l'ontologia è inconsistente



## Restrizioni su data properties

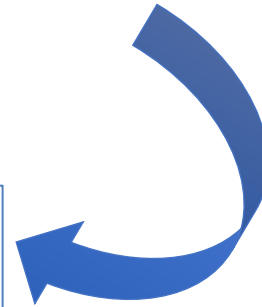
- E' possibile porre una restrizione su una data property
  - Usando quantificatori e cardinalità tuttavia si esprime un vincolo sul numero delle relazioni, non sul valore che assume la proprietà
- Per creare classi definite in molti casi si usano i pattern che individuano un range specifico di valori rispetto al tipo di dato
  - Per esempio le stringhe che cominciano con una certa lettera, gli interi superiori a un certo valore, ecc.
  - Valgono le specifiche su XSD Datatypes
  - E' possibile creare specifici datatypes (es. un range di valori)

# Restrizioni su data properties: esempio

```
owl:onProperty :età ;  
owl:someValuesFrom [ rdf:type rdfs:Datatype ;  
                    owl:onDatatype xsd:nonNegativeInteger ;  
                    owl:withRestrictions ( [ xsd:minInclusive "18"^^xsd:nonNegativeInteger  
                                              ]  
                                              )  
                    ]
```

Equivalent To 

 **età** **some** **xsd:nonNegativeInteger[>= 18]**



# Property chain

---

In OWL2 è possibile creare delle catene di proprietà

```
:hasGrandparent owl:propertyChainAxiom ( :hasParent :hasParent ) .
```

Se si asserisce che

- $a$  hasParent  $b$
- $b$  hasParent  $c$

Un reasoner inferisce che  $a$  hasGrandParent  $c$

## Esempio property chain

Domains (intersection) +

● **Persona**

Ranges (intersection) +

● **Residenza**

Disjoint With +

SuperProperty Of (Chain) +

■ **coinquilinoDi** o **risiedeIn** **SubPropertyOf:** **risiedePressoCoinquilino**

- Se una persona A è coinquilina di una persona B e la persona B risiede in un certo luogo, anche la persona A risiede in quel luogo

# Classi primitive e ragionamento

---



LE CLASSI PRIMITIVE NON PERMETTONO AI  
REASONER DI COLLOCARE GLI INDIVIDUI  
NELLE CLASSI IN MODO AUTOMATICO



TUTTAVIA, ESSE PERMETTONO AI REASONER  
DI INDIVIDUARE INCONSISTENZE CON LE  
ASSERZIONI SUGLI INDIVIDUI

# Esempio di modellazione (1)

- Esistono **città** e **nazioni**. Le nazioni possono essere **nazioni europee, asiatiche, africane**, ecc.
- Le città sono *situate* nelle nazioni. Hanno un *nome* e una *data di fondazione*.
- Le città *hanno* una **popolazione**. La popolazione può essere una **popolazione ampia, media o scarsa**
- Le **città grandi** sono quelle che hanno una popolazione grande, quelle **medie** sono le città che hanno una popolazione media, ecc.
- Le **città europee** sono quelle situate in una nazione europea, quelle asiatiche sono le città situate in una nazione asiatica, ecc.

## Esempio di modellazione (2)

- Torino è situata in Italia e ha una popolazione media
- Tokio è situata in Giappone e ha una popolazione ampia
- Bergamo è una città piccola
- Suggerimenti: codificare nome e data di fondazione come data properties