

CC7220-1

LA WEB DE DATOS

PRIMAVERA 2022

LECTURE 4: WEB ONTOLOGY LANGUAGE (OWL) [I]

Aidan Hogan

aidhog@gmail.com

LAST TIME ...

SEMANTIC WEB: LOGIC

DATA:

Ireland



(Ireland,partOf,Europe)
 (Ireland,isA,Country)
 (Ireland,capital,Dublin)

Dublin

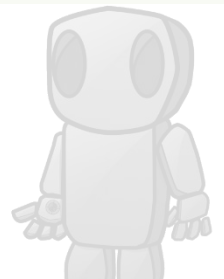


(Ireland,capital,Dublin)
 (Dublin,population,1000000)

LOGIC: $(b, \text{capital}, a) \rightarrow (a, \text{partOf}, b)$
 $(a, \text{partOf}, b), (b, \text{partOf}, c) \rightarrow (a, \text{partOf}, c)$

QUERY: $(x, \text{partOf}, y)?$

OUTPUT: $\{(x \mapsto \text{Ireland}, y \mapsto \text{Europe}),$
 $(x \mapsto \text{Dublin}, y \mapsto \text{Ireland}),$
 $(x \mapsto \text{Dublin}, y \mapsto \text{Europe})\}$



RDF SCHEMA: RDFS

Class c is a **sub-class** of Class d

If $(x, \text{rdf:type}, c)$ then $(x, \text{rdf:type}, d)$

Property p is a **sub-property** of q

If (x, p, y) then (x, q, y)

Property p has **domain** class c

If (x, p, y) then $(x, \text{rdf:type}, c)$


Property p has **range** class c

If (x, p, y) then $(y, \text{rdf:type}, c)$

TODAY'S TOPIC ...

SEMANTIC WEB: LOGIC

DATA:

Ireland 

(Ireland,partOf,Europe)
 (Ireland,isA,Country)
 (Ireland,capital,Dublin)

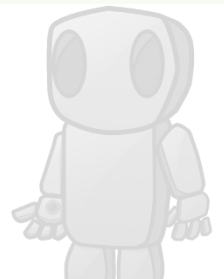
Dublin 

(Ireland,capital,Dublin)
 (Dublin,population,1000000)

LOGIC: $(b, \text{capital}, a) \rightarrow (a, \text{partOf}, b)$
 $(a, \text{partOf}, b), (b, \text{partOf}, c) \rightarrow (a, \text{partOf}, c)$

QUERY: $(x, \text{partOf}, y)?$

OUTPUT: $\{(x \mapsto \text{Ireland}, y \mapsto \text{Europe}),$
 $(x \mapsto \text{Dublin}, y \mapsto \text{Ireland}),$
 $(x \mapsto \text{Dublin}, y \mapsto \text{Europe})\}$



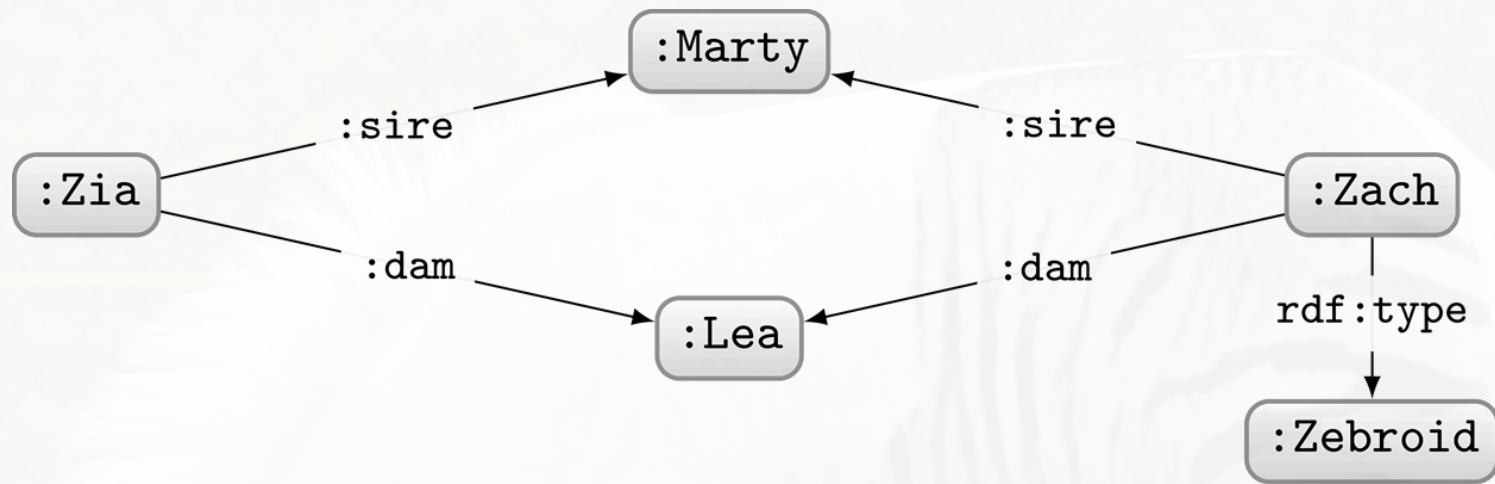
An iceberg floating in the ocean. The tip of the iceberg is visible above the water surface, while the much larger, submerged part is visible below. The sky is blue with light clouds, and the water is a deep blue. The text '← RDFS' is positioned to the right of the visible tip of the iceberg.

← RDFS

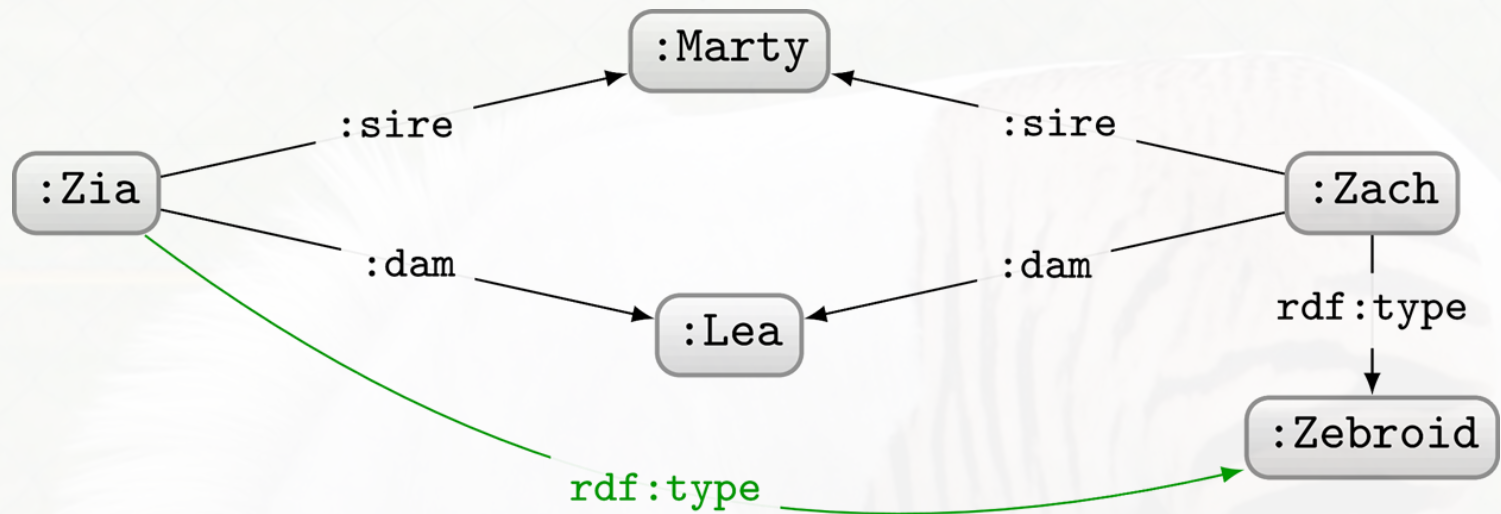
A cartoon illustration of a brown owl wearing a blue snorkel mask and a blue snorkel tube. The owl is looking forward with large, yellow eyes. The background is dark blue.

← OWL





What can we intuitively conclude about Zia?

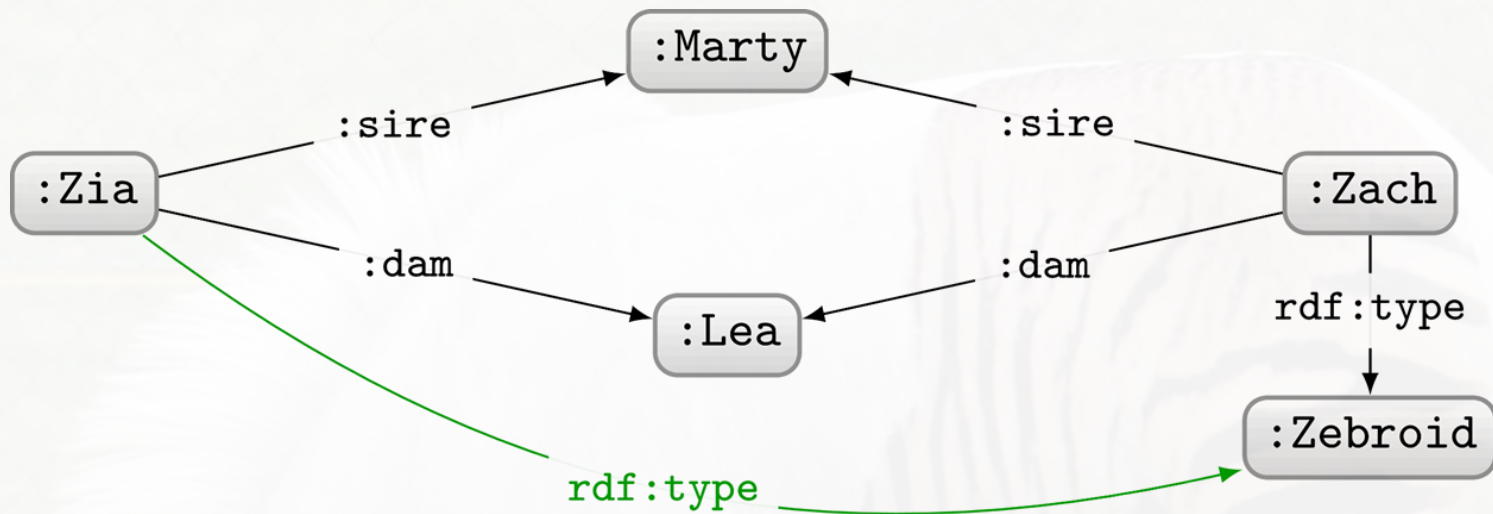


What can we intuitively conclude about Zia?

Zia is also a Zebroid!

What kind of reasoning are we using here?

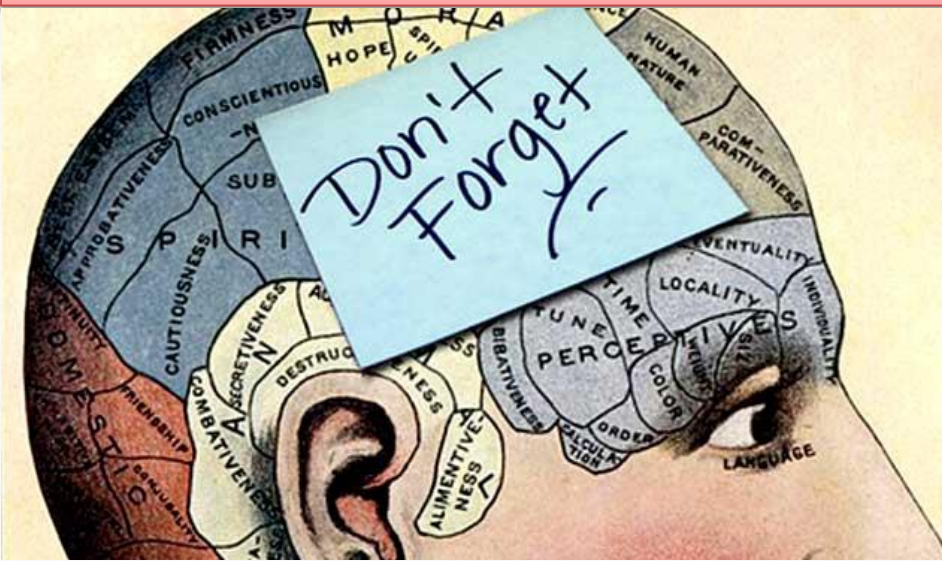
Deductive (mostly)

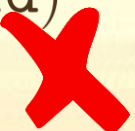


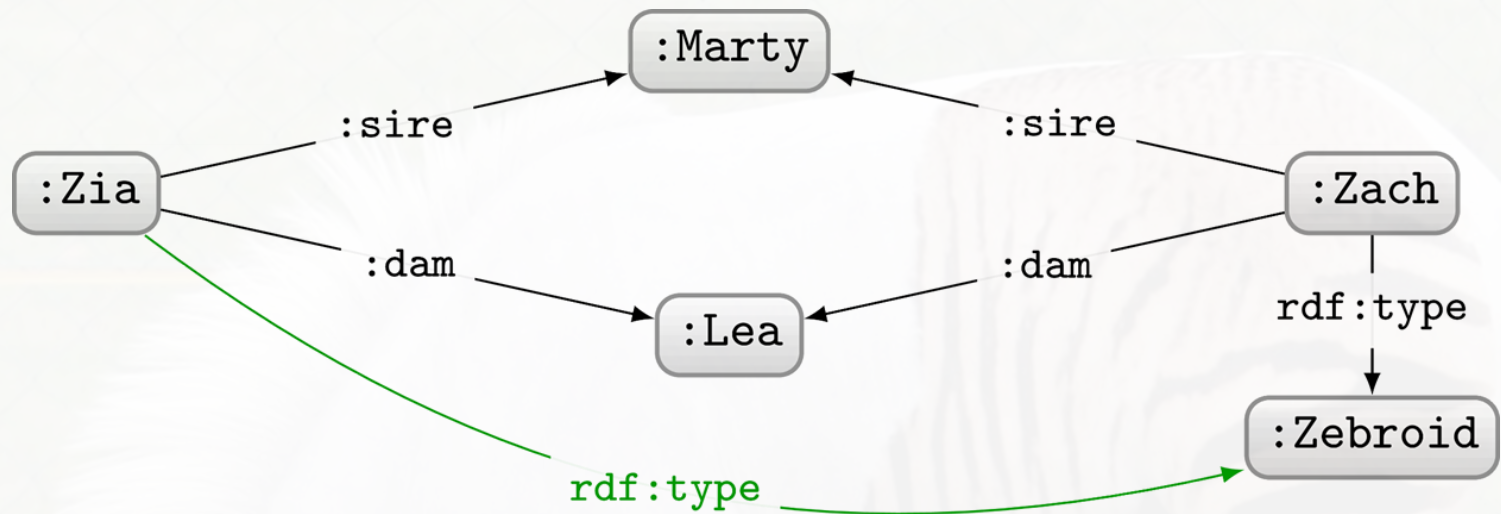
What assumptions do we make to conclude that Zia is a Zebroid?

If x has same sire and dam as y and y is a Zebroid then x is a Zebroid!

Very specific to this example

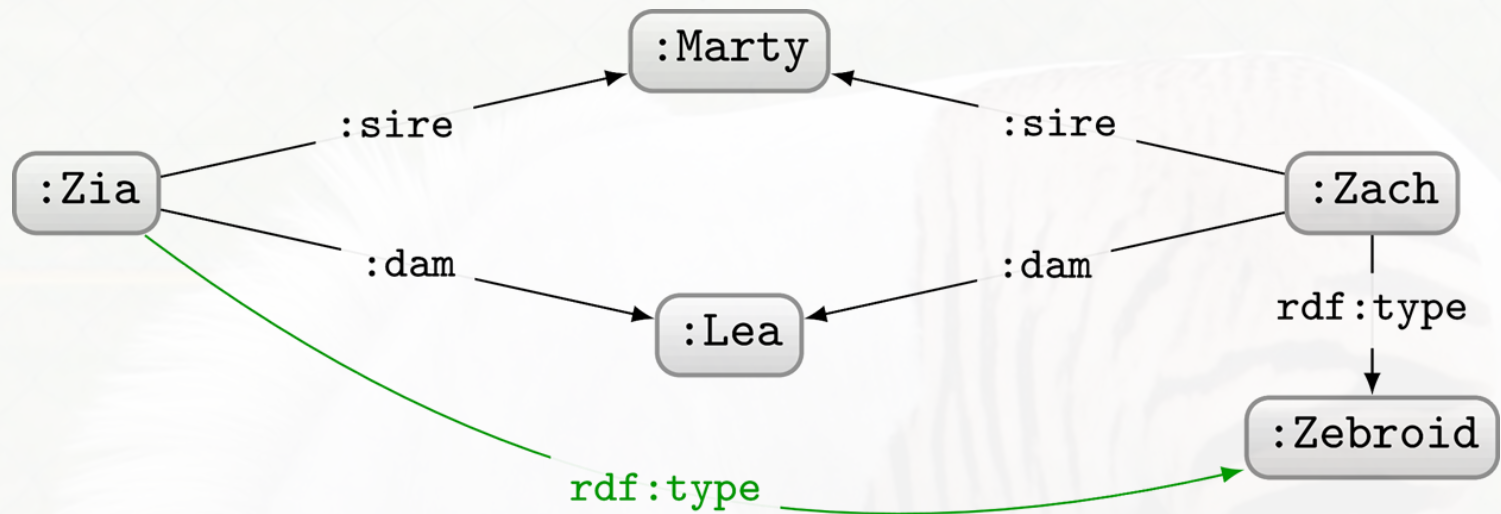


$$\begin{aligned}
 & (x, :dam, z_1), (x, :sire, z_2), \\
 & (y, :dam, z_1), (y, :sire, z_2), \\
 & (y, rdf:type, :Zebroid) \\
 \rightarrow & (x, rdf:type, :Zebroid)
 \end{aligned}$$




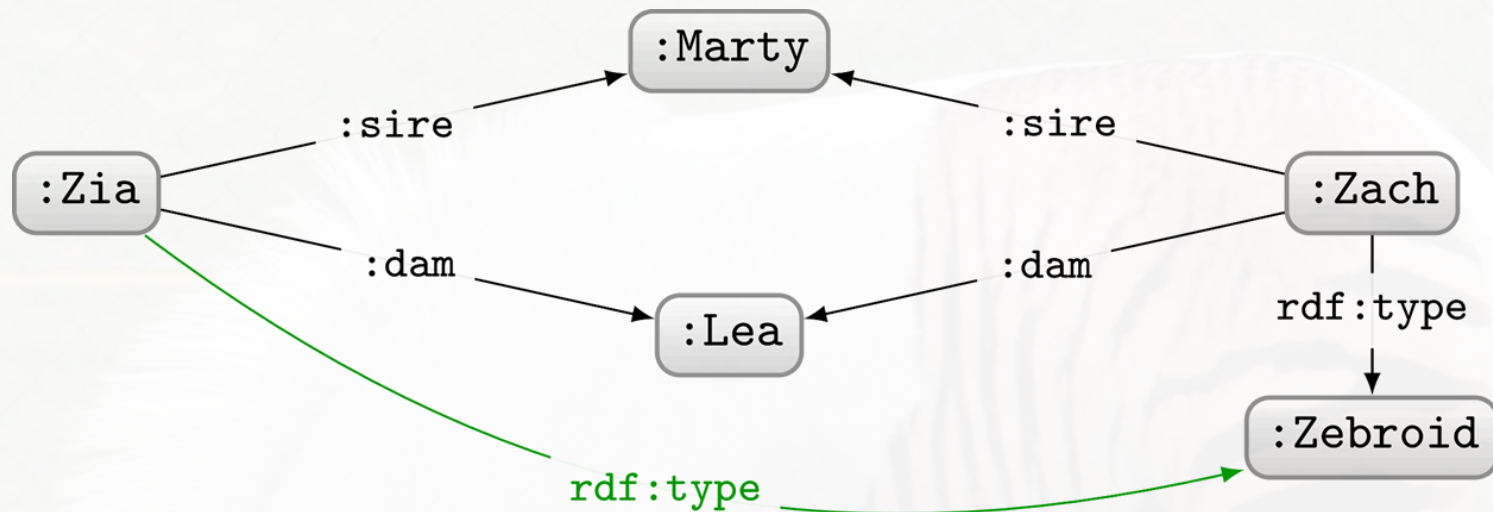
What assumptions do we make to conclude that Zia is a Zebroid?

- sire is a sub-property of parent
- dam is a sub-property of parent



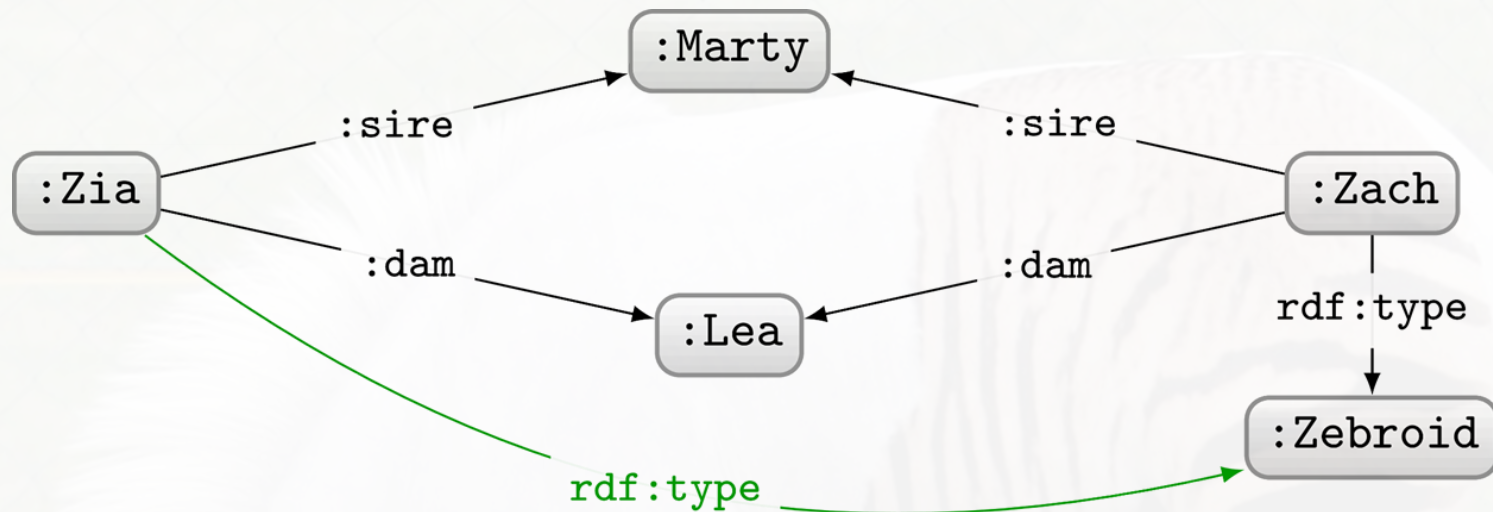
What assumptions do we make to conclude that Zia is a Zebroid?

- sire is a sub-property of parent
- dam is a sub-property of parent
- A Zebroid has exactly one parent a Zebra
- A Zebroid has exactly one parent a (-Zebra and a Equine)



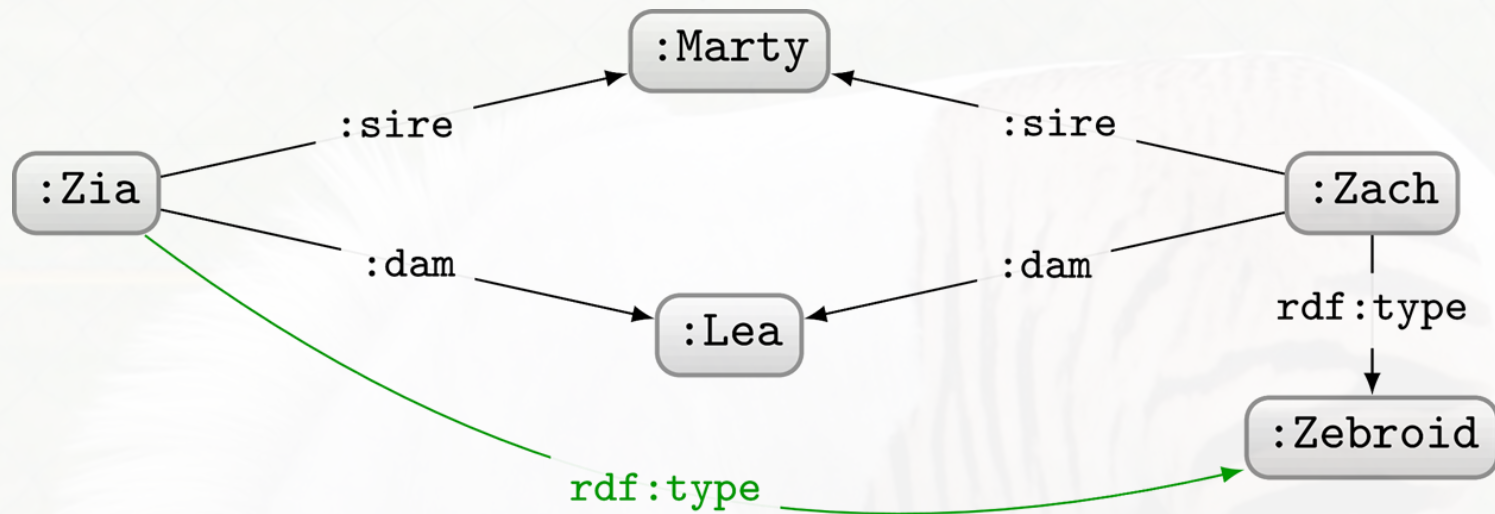
What assumptions do we make to conclude that Zia is a Zebroid?

- sire is a sub-property of parent
- dam is a sub-property of parent
- A Zebroid has exactly one parent a Zebra
- A Zebroid has exactly one parent a (-Zebra and a Equine)
- A Zebroid is a sub-class of Equine
- An Equine has exactly two parents



What assumptions do we make to conclude that Zia is a Zebroid?

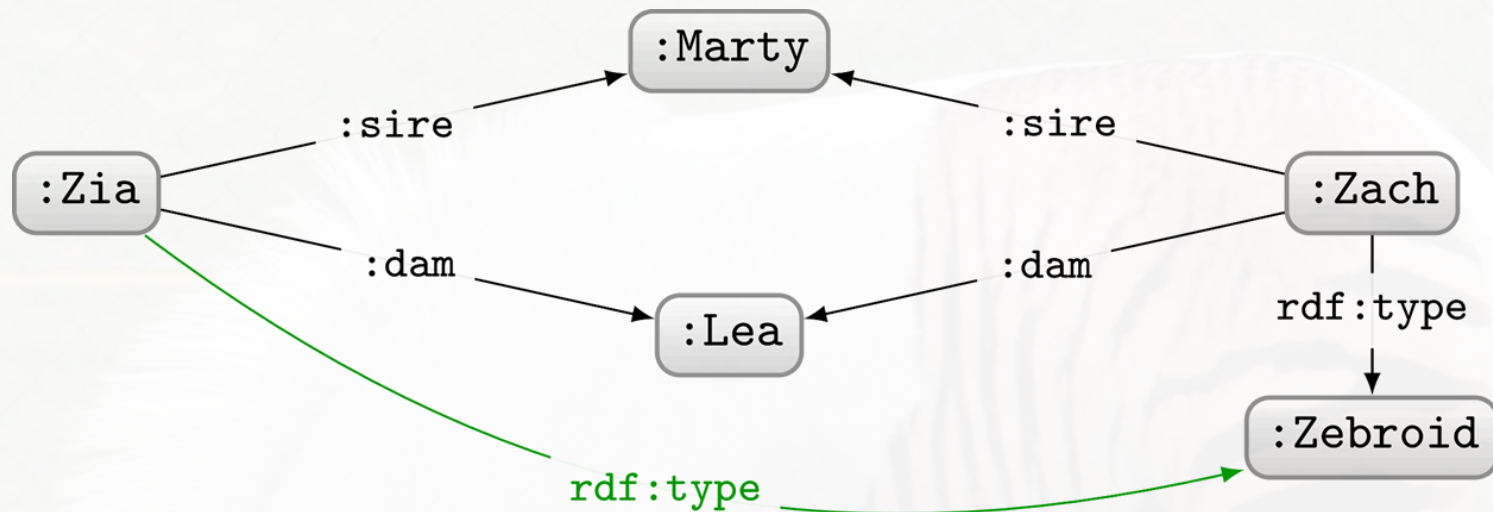
- sire is a sub-property of parent
- dam is a sub-property of parent
- A Zebroid has exactly one parent a Zebra
- A Zebroid has exactly one parent a (-Zebra and a Equine)
- A Zebroid is a sub-class of Equine
- An Equine has exactly two parents
- Two things cannot be related by sire and dam at the same time



What assumptions do we make to conclude that Zia is a Zebroid?

- sire is a sub-property of parent
- dam is a sub-property of parent
- A Zebroid has exactly one parent a Zebra
- A Zebroid has exactly one parent a (-Zebra and a Equine)
- A Zebroid is a sub-class of Equine
- An Equine has exactly two parents
- Two things cannot be related by sire and dam at the same time

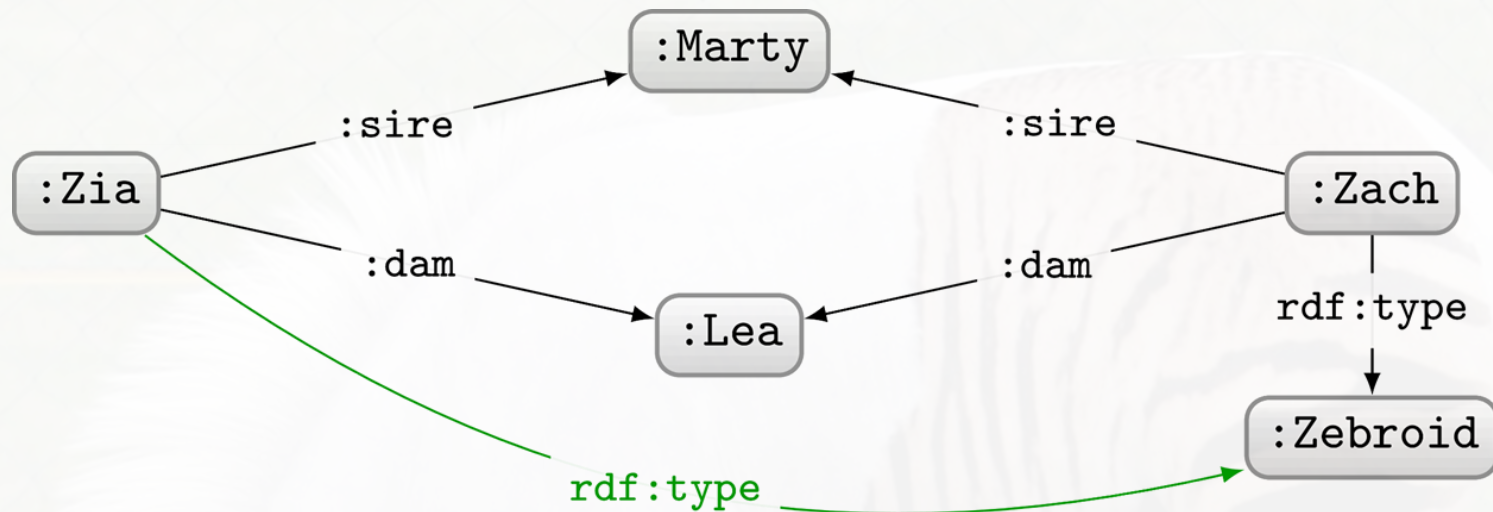
Which are expressible in RDFS?



What assumptions do we make to conclude that Zia is a Zebroid?

- *sire* is a sub-property of parent
- *dam* is a sub-property of parent
- A Zebroid has exactly one parent a Zebra
- A Zebroid has exactly one parent a (-Zebra and a Equine)
- *A Zebroid is a sub-class of Equine*
- An Equine has exactly two parents
- Two things cannot be related by *sire* and *dam* at the same time

Which are expressible in *RDFS*?



What assumptions do we make to conclude that Zia is a Zebroid?

- sire is a sub-property of parent
- dam is a sub-property of parent
- A Zebroid has exactly one parent a Zebra
- A Zebroid has exactly one parent a (-Zebra and a Equine)
- A Zebroid is a sub-class of Equine
- An Equine has exactly two parents
- Two things cannot be related by sire and dam at the same time

Which are expressible in RDFS?

The rest we can express in OWL

WEB ONTOLOGY LANGUAGE: OWL

OWL (2): A WEB STANDARD



<https://www.w3.org/TR/owl2-overview/>

OWL 2 Web Ontology Language Document Overview (Second Edition)

W3C Recommendation 11 December 2012

This version:

<http://www.w3.org/TR/2012/REC-owl2-overview-20121211/>

Latest version (series 2):

<http://www.w3.org/TR/owl2-overview/>

Latest Recommendation:

<http://www.w3.org/TR/owl-overview>

Previous version:

<http://www.w3.org/TR/2012/PER-owl2-overview-20121018/>

Editors:

W3C OWL Working Group (see [Acknowledgements](#))

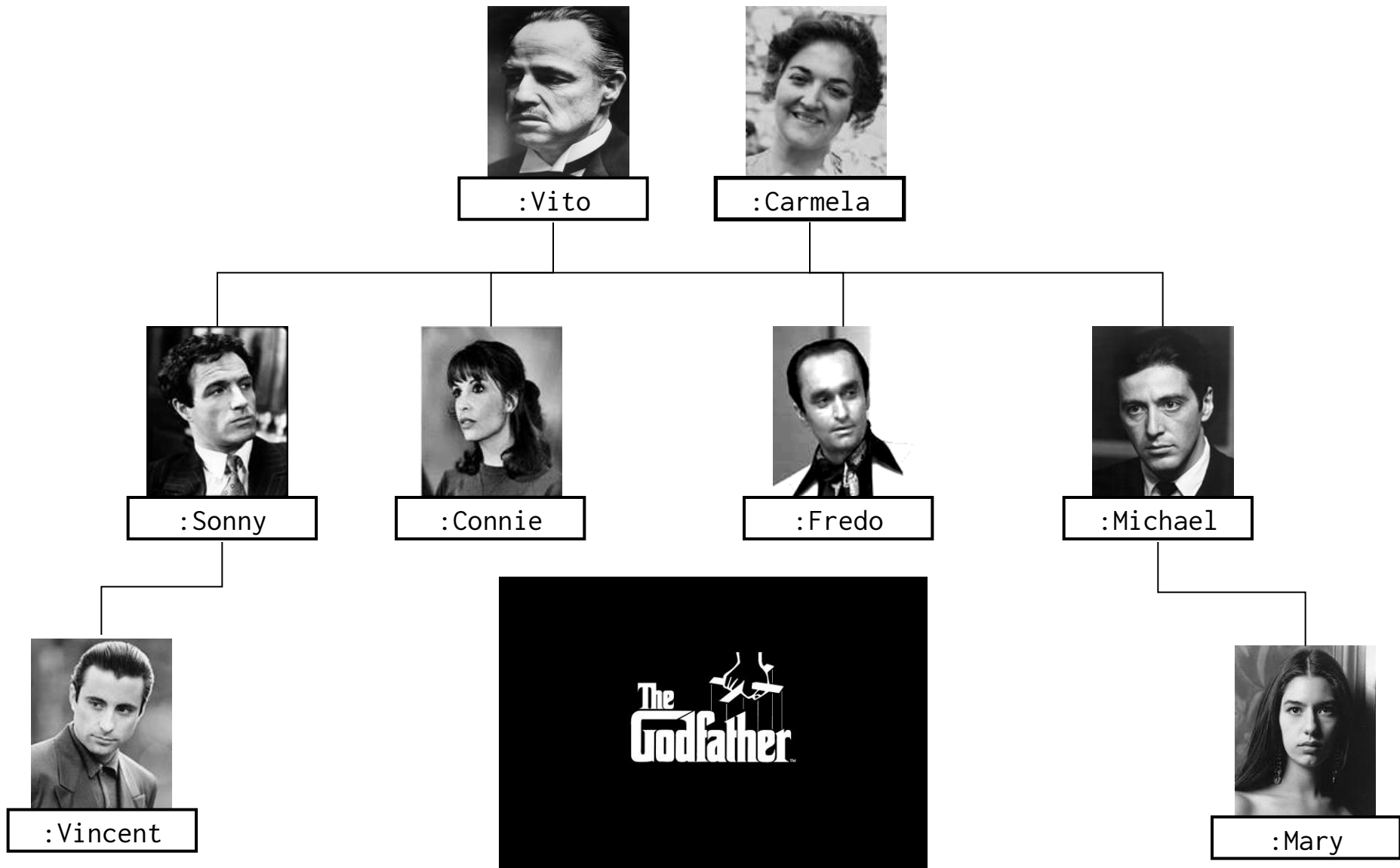
Please refer to the [errata](#) for this document, which may include some normative corrections.

A [color-coded version of this document showing changes made since the previous version](#) is also available.

FORMAL UNDERPINNINGS: DESCRIPTION LOGICS

Name	Syntax	OWL key-term	DL
CONCEPT DEFINITIONS			
Atomic Concept	A	owl:Class	\mathcal{ALC}
Top Concept	\top	owl:Thing	\mathcal{ALC}
Bottom Concept	\perp	owl:Nothing	\mathcal{ALC}
Concept Negation	$\neg C$	owl:complementOf	\mathcal{ALC}
Concept Intersection	$C \sqcap D$	owl:intersectionOf	\mathcal{ALC}
Concept Union	$C \sqcup D$	owl:unionOf	\mathcal{ALC}
Nominal	$\{a_1, \dots, a_n\}$	owl:oneOf	\mathcal{O}
Existential Restriction	$\exists R.C$	owl:someValuesFrom	\mathcal{ALC}
Universal Restriction	$\forall R.C$	owl:allValuesFrom	\mathcal{ALC}
Self Restriction	$\exists R.\text{Self}$	owl:hasSelf	\mathcal{R}
Number Restriction	$\leq n R, \geq n R, = n R$	owl:*cardinality	\mathcal{N}
Qualified Number Restriction	$\leq n R.C, \geq n R.C, = n R.C$	owl:*qualifiedCardinality	\mathcal{Q}
CONCEPT AXIOMS (T-Box)			
Concept Inclusion	$C \sqsubseteq D$	rdfs:subClassOf	\mathcal{ALC}
ROLE DEFINITIONS			
Role	R	owl:*Property	\mathcal{ALC}
Inverse Role	R^-	owl:inverseOf	\mathcal{I}
Universal Role	U	owl:top*Property	\mathcal{R}
ROLE AXIOMS (R-Box)			
Role Inclusion	$R \sqsubseteq S$	rdfs:subPropertyOf	\mathcal{H}
Complex Role Inclusion	$R_1 \circ \dots \circ R_n \sqsubseteq S$	owl:propertyChainAxiom	\mathcal{R}
Transitive Roles	Trans(R)	owl:TransitiveProperty	\mathcal{S}
Functional Roles	Func(R)	owl:FunctionalProperty	\mathcal{F}
Reflexive Roles	Ref(R)	owl:ReflexiveProperty	\mathcal{R}
Irreflexive Roles	Irref(R)	owl:IrreflexiveProperty	\mathcal{R}
Symmetric Roles	Sym(R)	owl:SymmetricProperty	\mathcal{I}
Asymmetric Roles	Asym(R)	owl:AsymmetricProperty	\mathcal{R}
Disjoint Roles	Disj(R, S)	owl:disjointPropertyWith	\mathcal{R}
ASSERTIONAL DEFINITIONS			
(Named) Individual	a	(RDF IRI or Literal)	\mathcal{ALC}
ASSERTIONAL AXIOMS (A-Box)			
Role Assertion	$R(a, b)$	(RDF triple)	\mathcal{ALC}
Negative Role Assertion	$\neg R(a, b)$	owl:NegativePropertyAssertion	\mathcal{ALC}
Concept Assertion	$C(a)$	rdf:type	\mathcal{ALC}
Equality	$a = b$	owl:sameAs	\mathcal{ALC}
Inequality	$a \neq b$	owl:differentFrom	\mathcal{ALC}

FOR TODAY: A RUNNING EXAMPLE

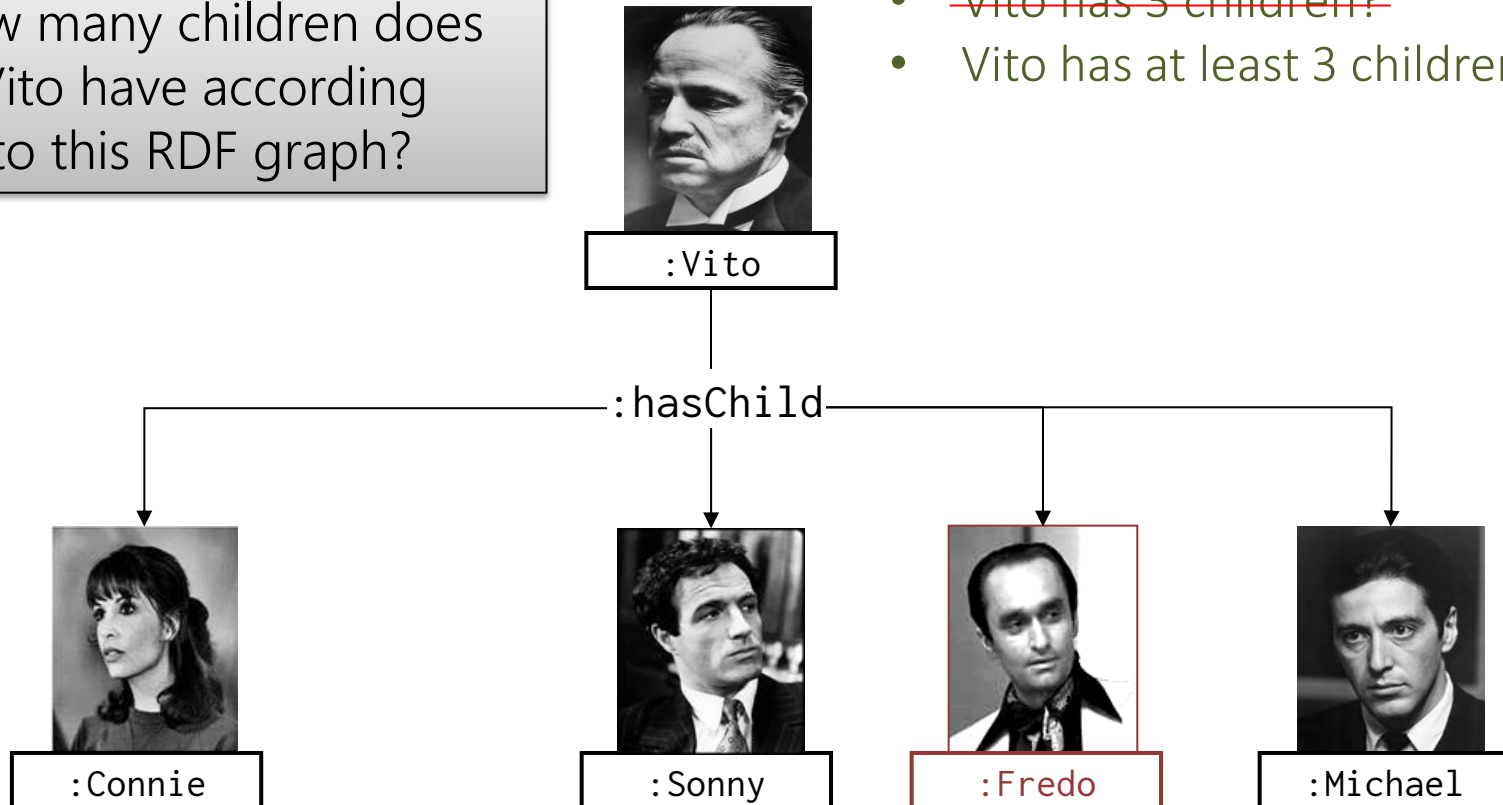


LOGICAL ASSUMPTIONS

OPEN WORLD ASSUMPTION (OWA)

How many children does Vito have according to this RDF graph?

- ~~Vito has 3 children?~~
- Vito has at least 3 children?



```
:Vito :hasChild :Connie , :Sonny , :Michael .
```

```
:Vito :hasChild :Fredo .
```

```
... ?
```

OPEN WORLD ASSUMPTION

- RDF(S) and OWL:
 - Take an **Open World Assumption (OWA)**:
 - Anything not known is not assumed to be false, simply unknown
 - Without further information, Vito may have children that we don't know about!

Why might this assumption be important for the Web?

OWA: Assuming Web data to be complete a **Bad Idea**®.

No UNIQUE NAME ASSUMPTION (No UNA)

How many children does Vito have according to this RDF graph?



:Vito

:hasChild



:Connie



:Sonny



:Fredo



:Michael

- ~~Vito has 3 children?~~
- ~~Vito has at least 3 children?~~
- Vito has at least one child!

```
:Vito :hasChild :Connie , :Sonny , :Michael .
```

```
:Vito :hasChild :Fredo .
```

```
... ?
```


NO UNIQUE NAME ASSUMPTION (NO UNA)

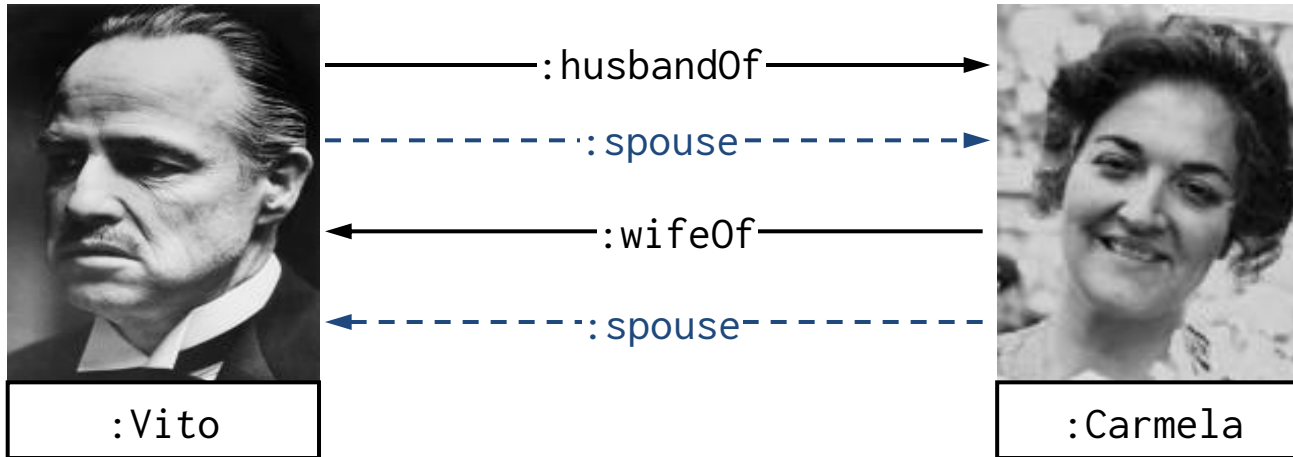
- RDF(S) and OWL:
 - Do not take a **Unique Name Assumption**:
 - Two or more IRIs may refer to the same thing!
 - Without further information, the IRIs we know to be Vito's children may refer to one real-world thing!

Why might this assumption be important for the Web?

No **UNA**: Assuming strict naming agreement on the Web a **Bad Idea**[®].

LET'S START WITH SOME RDFS ...

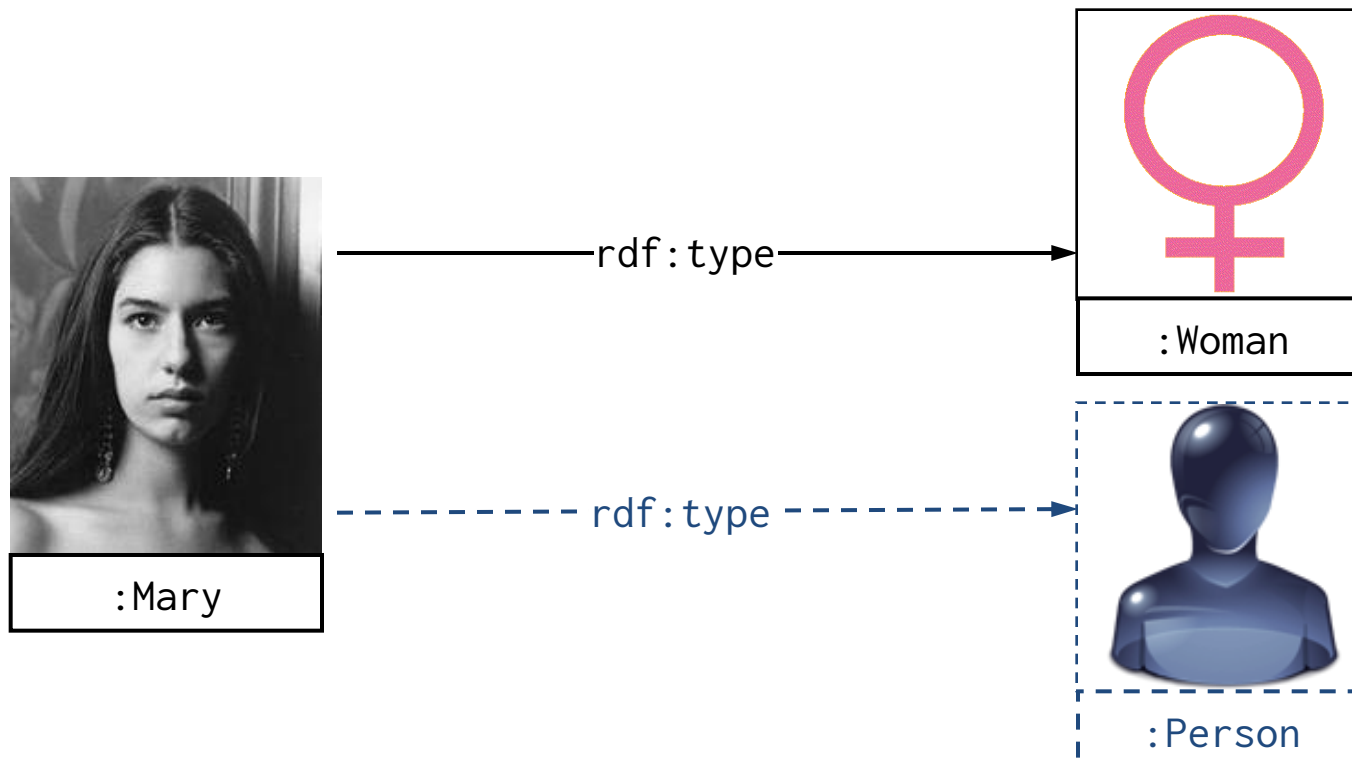
rdfs:subPropertyOf



```
:Vito :husbandOf :Carmela .  
:husbandOf rdfs:subPropertyOf :spouse .  
⇒ :Vito :spouse :Carmela .
```

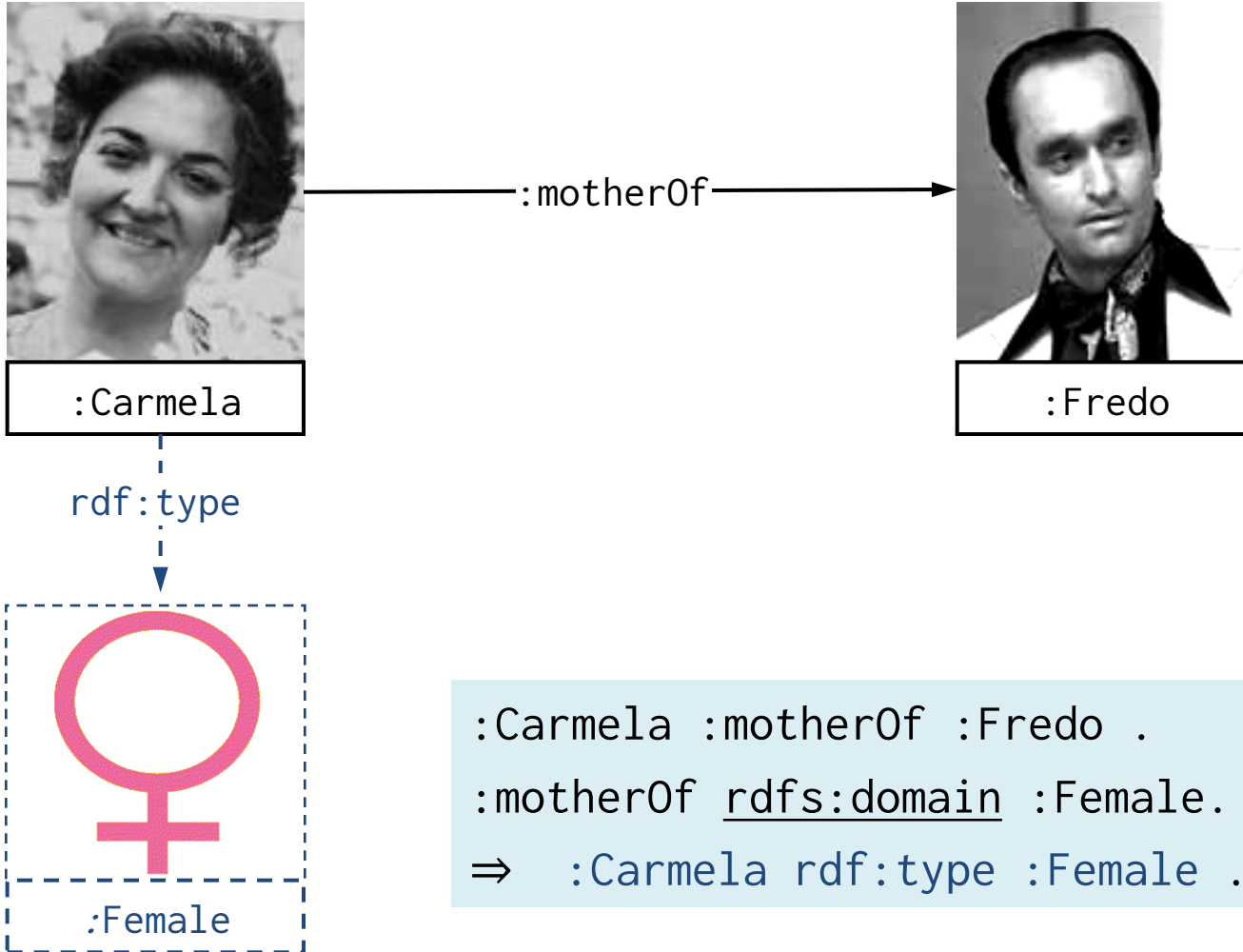
```
:Carmela :wifeOf :Vito .  
:wifeOf rdfs:subPropertyOf :spouse .  
⇒ :Carmela :spouse :Vito .
```

rdfs:subClassOf



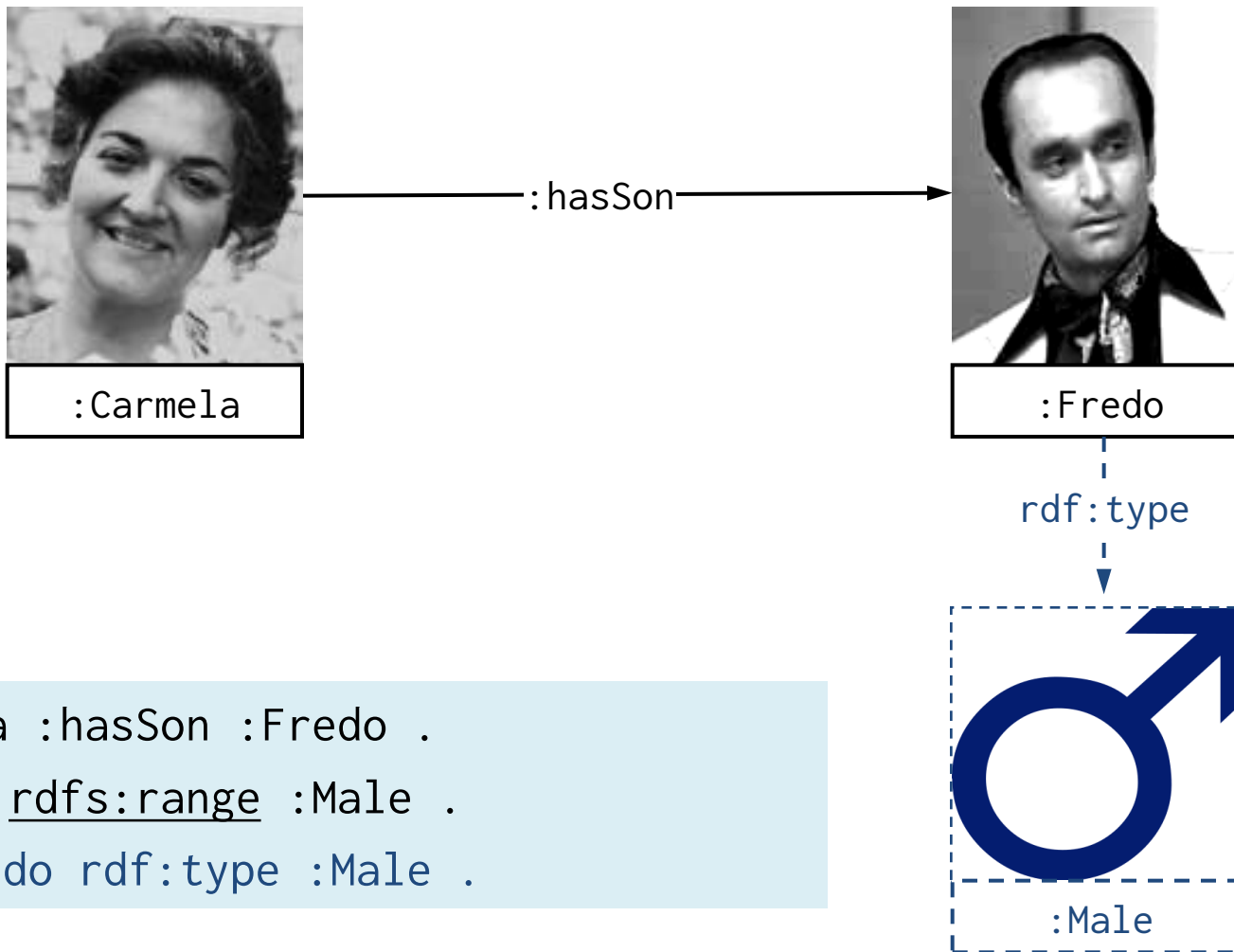
```
:Mary rdf:type :Woman .  
:Woman rdfs:subClassOf :Person .  
⇒ :Mary rdf:type :Person .
```

rdfs:domain



```
:Carmela :motherOf :Fredo .  
:motherOf rdfs:domain :Female .  
⇒ :Carmela rdf:type :Female .
```

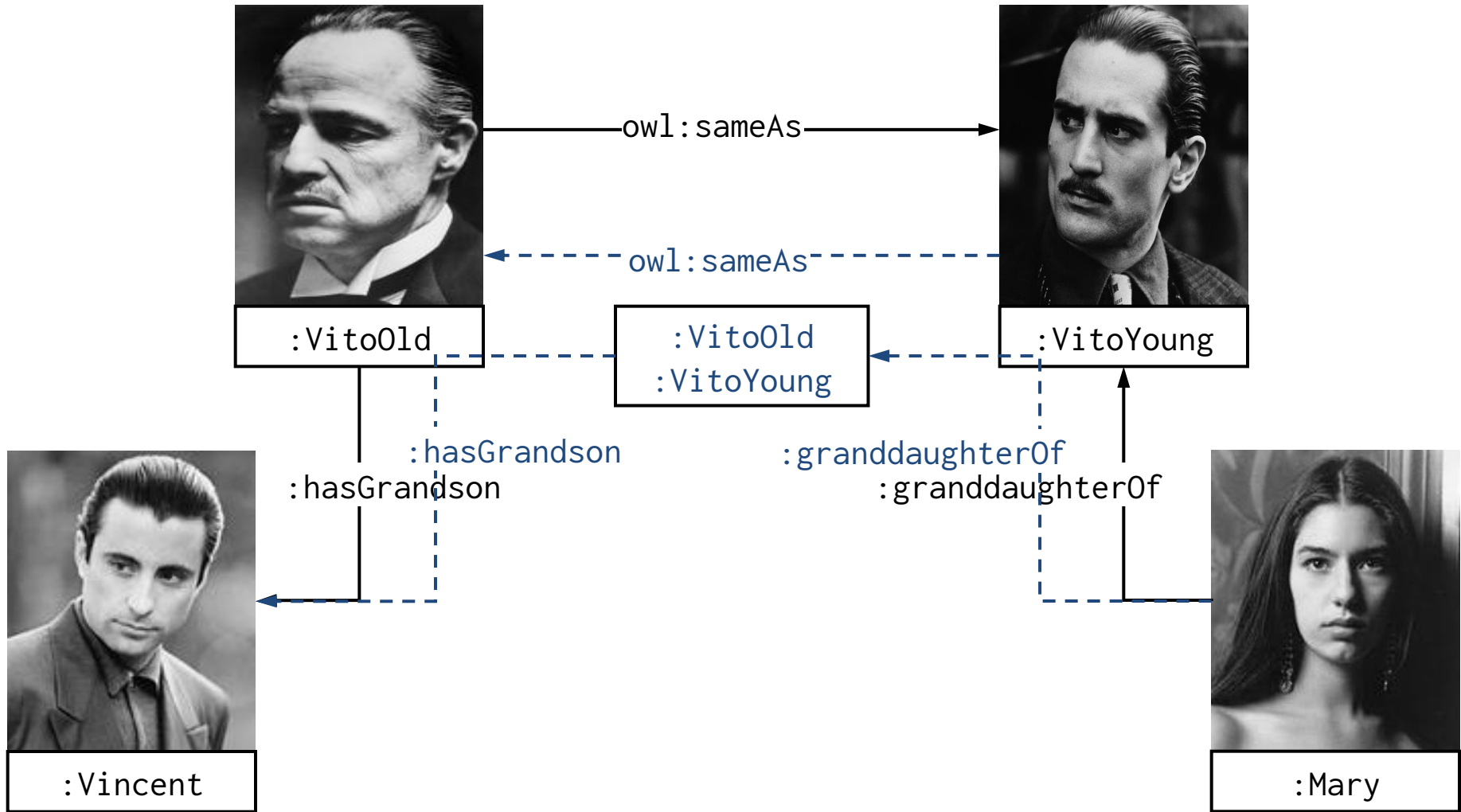
rdfs:range



```
:Carmela :hasSon :Fredo .  
:hasSon rdfs:range :Male .  
⇒ :Fredo rdf:type :Male .
```

(IN)EQUALITY IN OWL ...

owl:sameAs

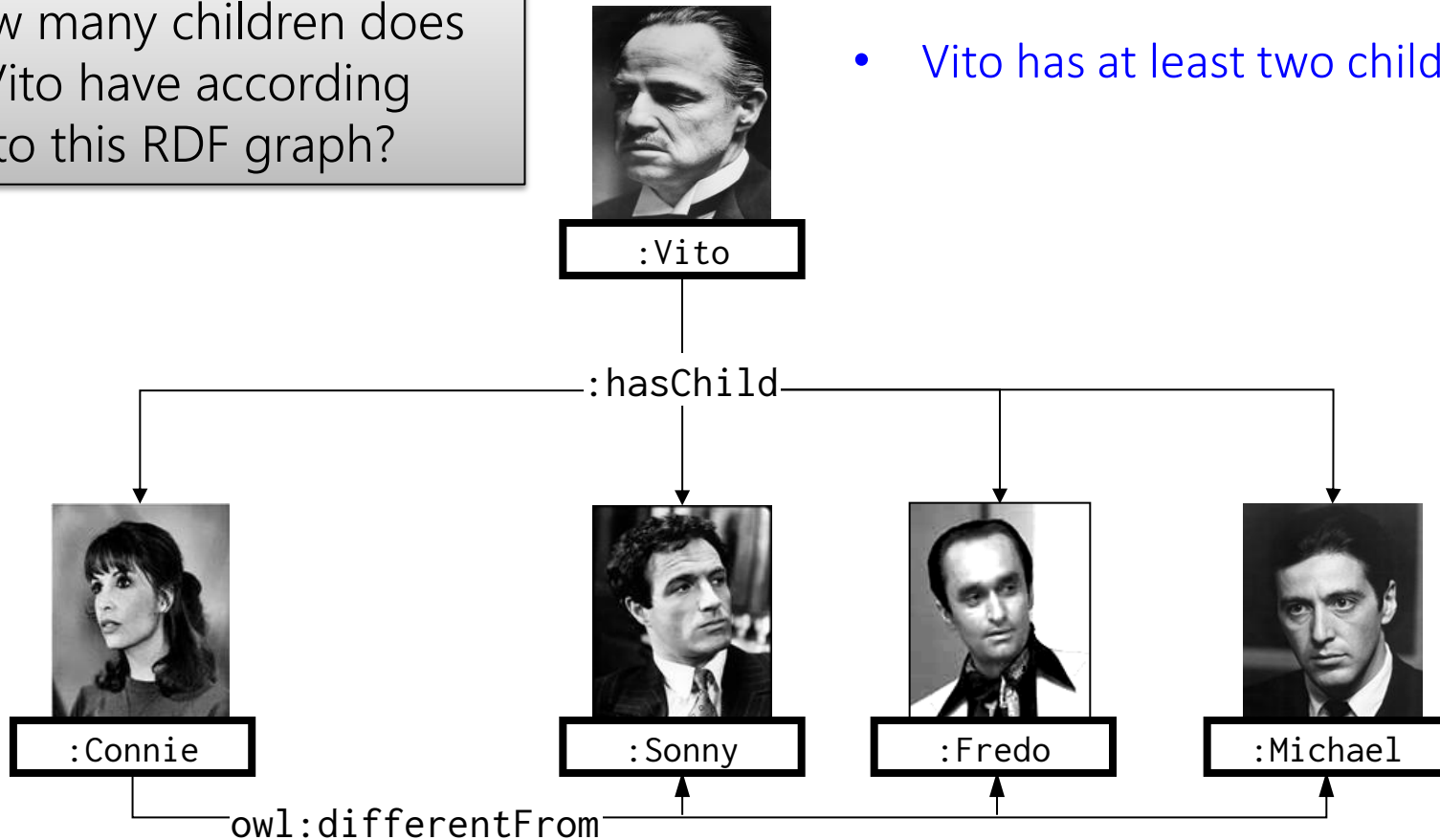


```
:VitoOld owl:sameAs :VitoYoung .
```


owl:differentFrom

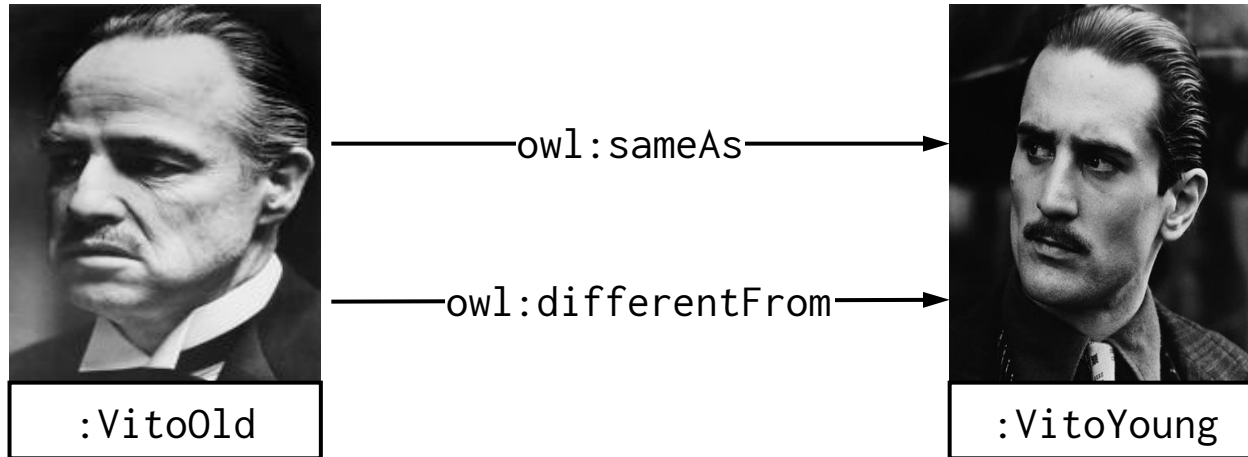
How many children does Vito have according to this RDF graph?

- Vito has at least two children!



```
:Vito :hasChild :Connie, :Sonny, :Michael, :Fredo .  
:Connie owl:differentFrom :Sonny, :Michael, :Fredo .
```

INCONSISTENCY IN OWL ...

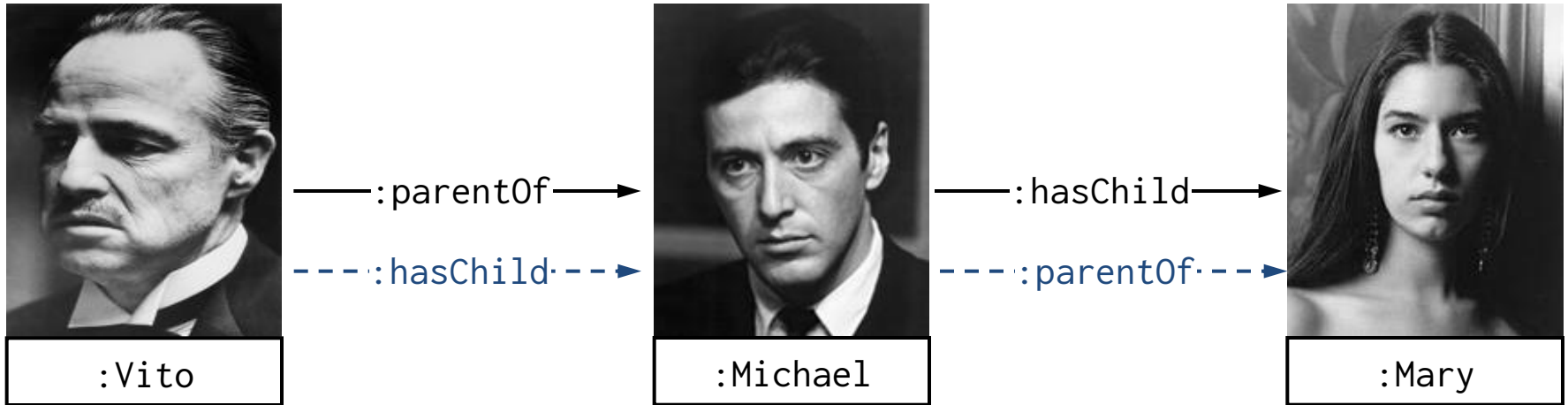


```
:VitoOld owl:sameAs :VitoYoung .  
:VitoOld owl:differentFrom :VitoYoung .  
⇒ FALSE
```



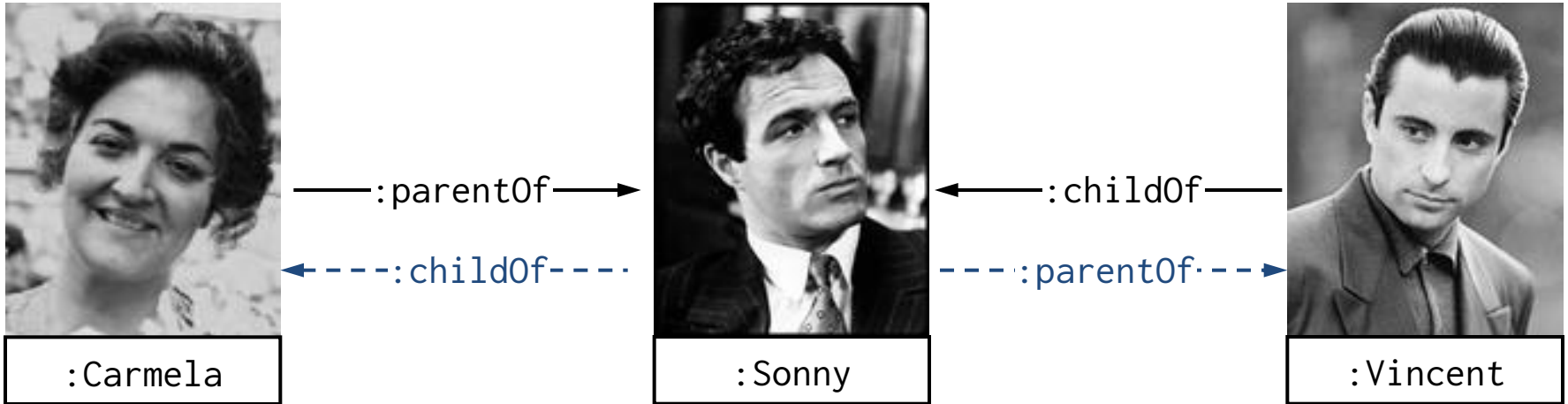
PROPERTY AXIOMS IN OWL ...

owl:equivalentProperty



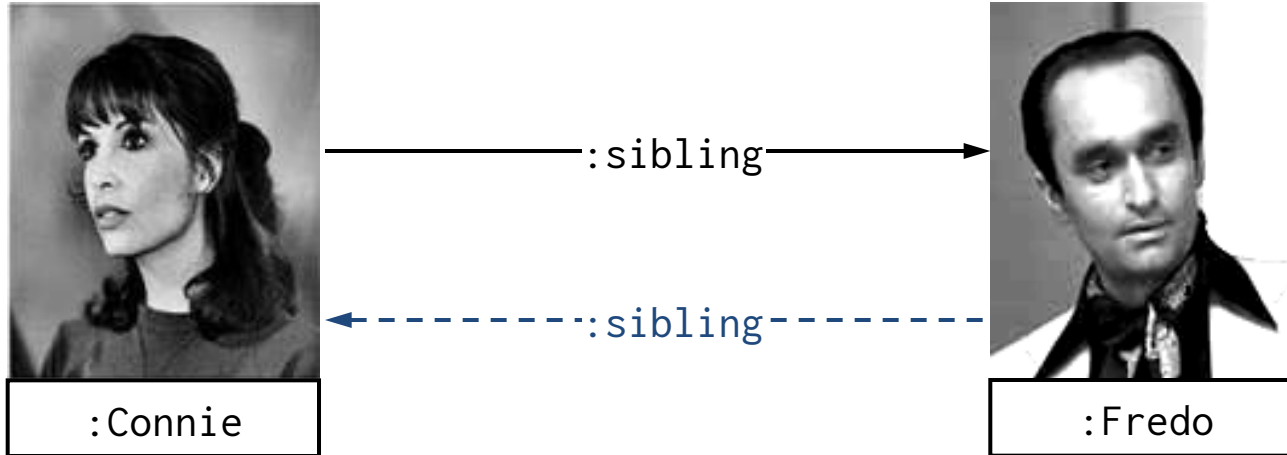
```
:Vito :parentOf :Michael .  
:Michael :hasChild :Mary .  
:parentOf owl:equivalentProperty :hasChild .  
⇒ :Vito :hasChild :Michael .  
⇒ :Michael :parentOf :Mary .
```

owl:inverseOf



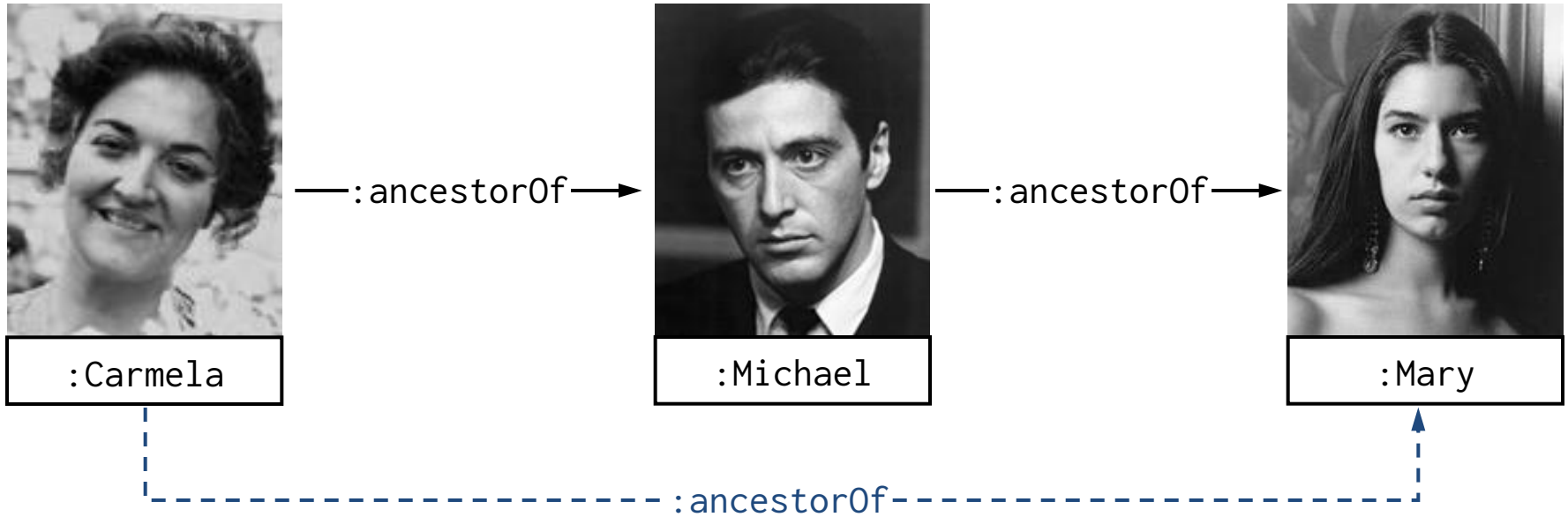
```
:Carmela :parentOf :Sonny .  
:Vincent :childOf :Sonny .  
:parentOf owl:inverseOf :childOf .  
⇒ :Sonny :parentOf :Vincent .  
⇒ :Sonny :childOf :Carmela .
```

owl:SymmetricProperty



```
:Connie :sibling :Fredo .  
:sibling rdf:type owl:SymmetricProperty .  
⇒ :Fredo :sibling :Connie .
```

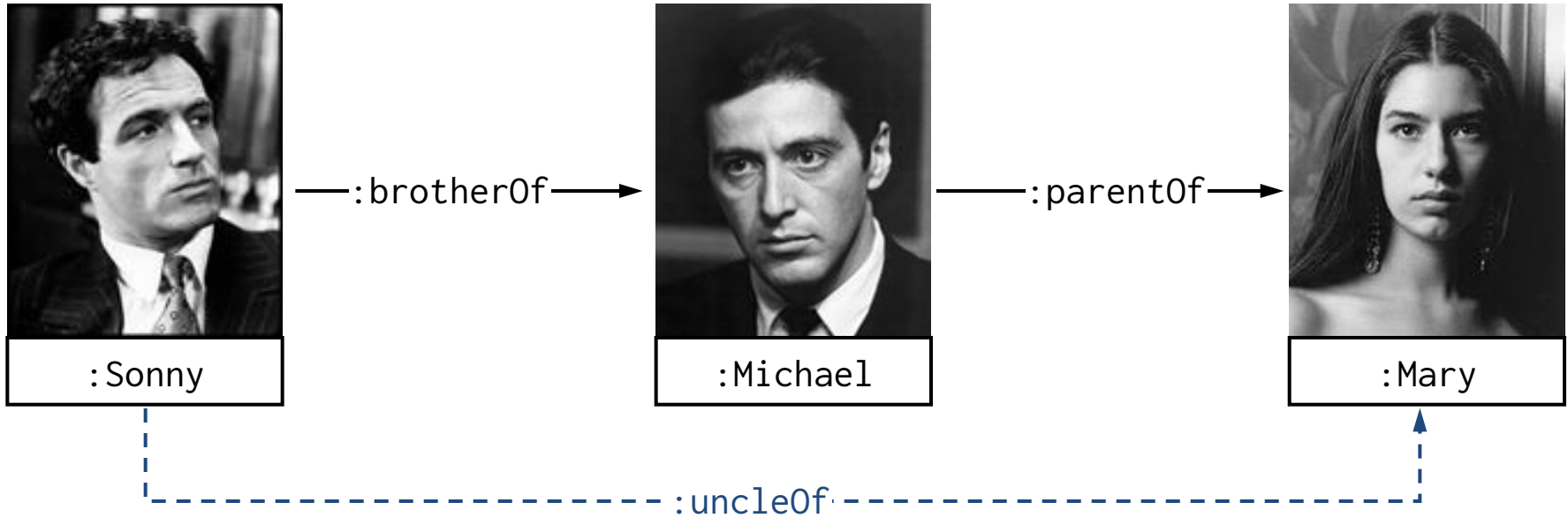
owl:TransitiveProperty



```
:Carmela :ancestorOf :Michael .  
:Michael :ancestorOf :Mary .  
:ancestorOf rdf:type owl:TransitiveProperty .  
⇒ :Carmela :ancestorOf :Mary .
```


owl:propertyChainAxiom

Means new to OWL version 2.0!



```
:Sonny :brotherOf :Michael .  
:Michael :parentOf :Mary .  
:uncleOf owl:propertyChainAxiom (:brotherOf :parentOf) .  
⇒ :Sonny :uncleOf :Mary .
```

owl:ReflexiveProperty



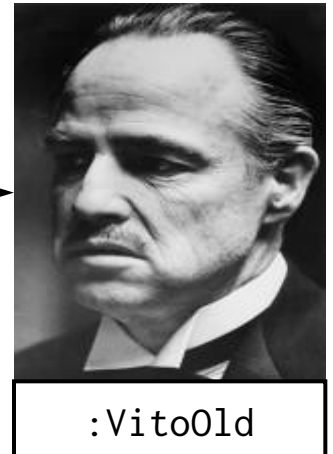
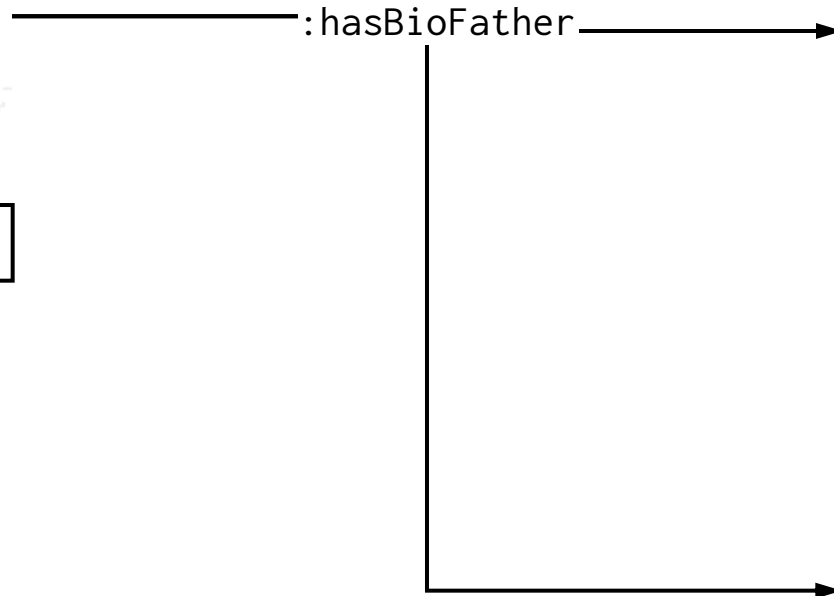
```
:similarTo rdf:type owl:ReflexiveProperty .
```

```
⇒ :Connie :similarTo :Connie .
```

```
   :Fredo :similarTo :Fredo .
```

```
# everything :similarTo itself
```

owl:FunctionalProperty



```
:Fredo :hasBioFather :VitoOld .  
:Fredo :hasBioFather :VitoYoung .  
:hasBioFather rdf:type owl:FunctionalProperty .  
⇒ :VitoOld owl:sameAs :VitoYoung .
```

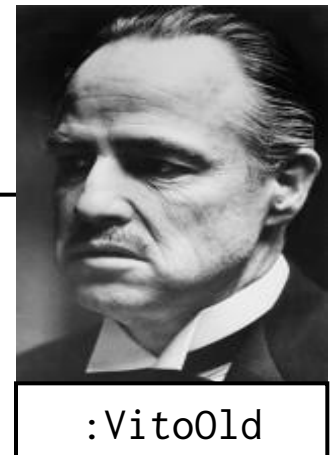
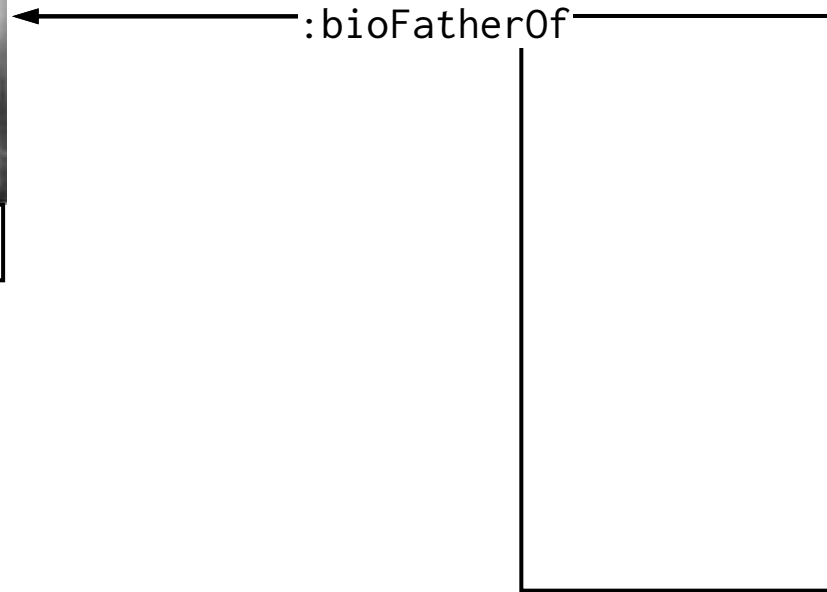
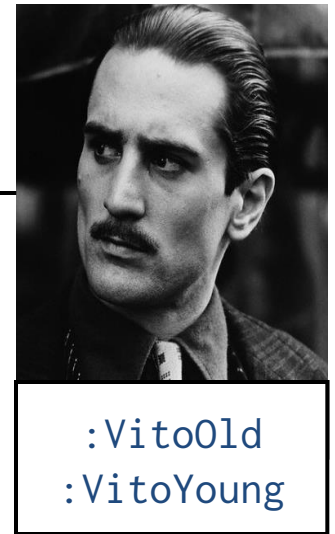
ASIDE ...

What if we said `:hasFather` was functional?



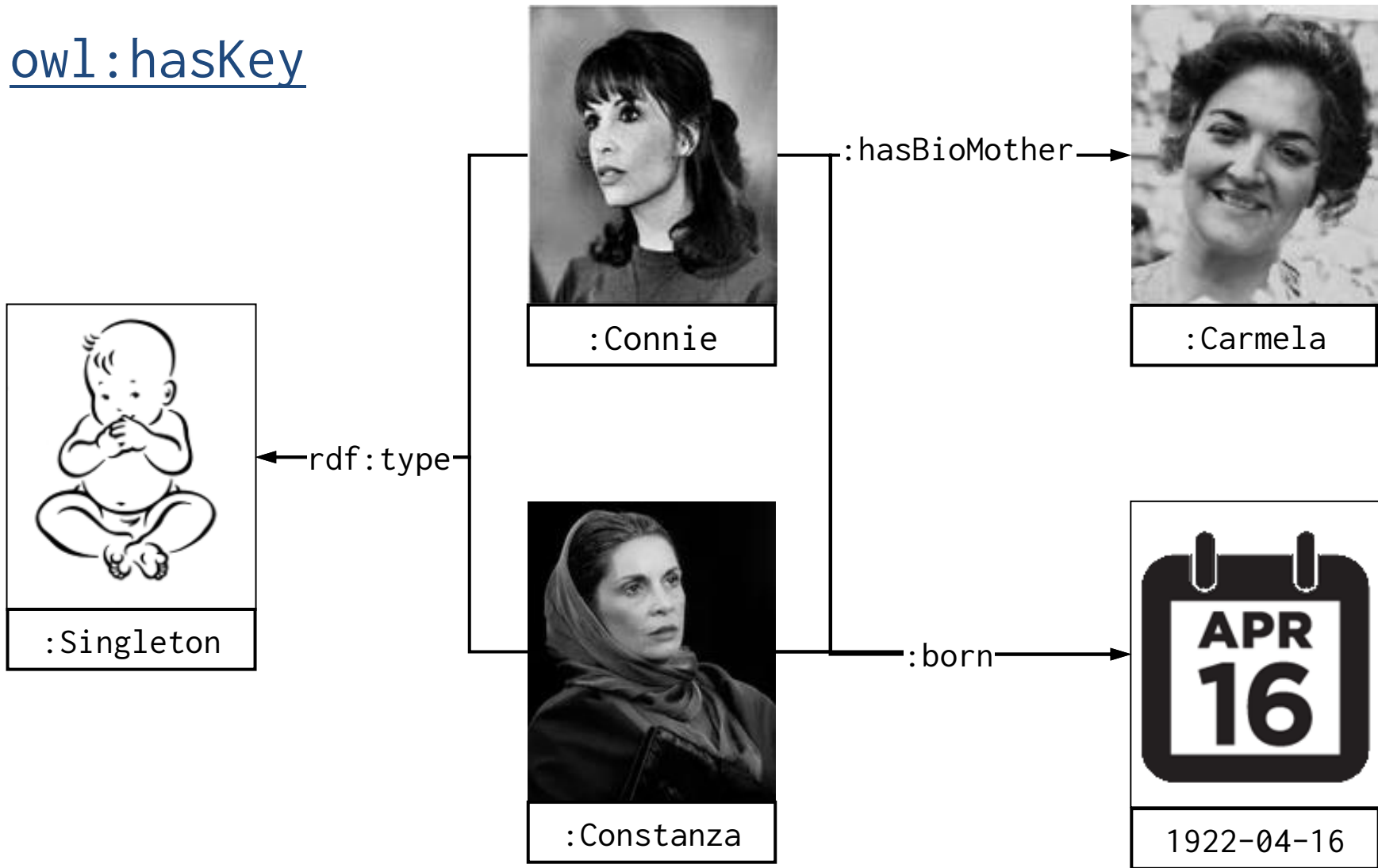
- Tom Hagen, the adopted son of Vito
 - Maybe he has two fathers?

owl:InverseFunctionalProperty



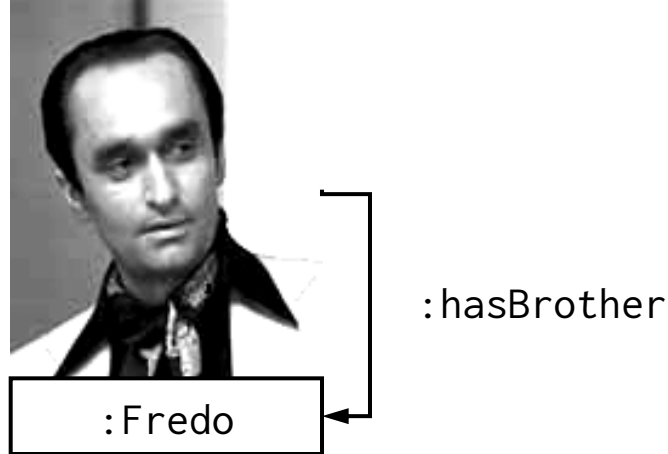
```
:VitoOld :bioFatherOf :Connie .  
:VitoYoung :bioFatherOf :Connie .  
:bioFatherOf rdf:type owl:InverseFunctionalProperty .  
⇒ :VitoOld owl:sameAs :VitoYoung .
```

owl:hasKey



```
:Connie a :Singleton ; :hasBioMother :Carmela ; :born "1922-04-16"^^xsd:date .
:Constanza a :Singleton ; :hasBioMother :Carmela ; :born "1922-04-16"^^xsd:date .
:Singleton owl:hasKey ( :hasBioMother :born ) .
⇒ :Connie owl:sameAs :Constanza .
```

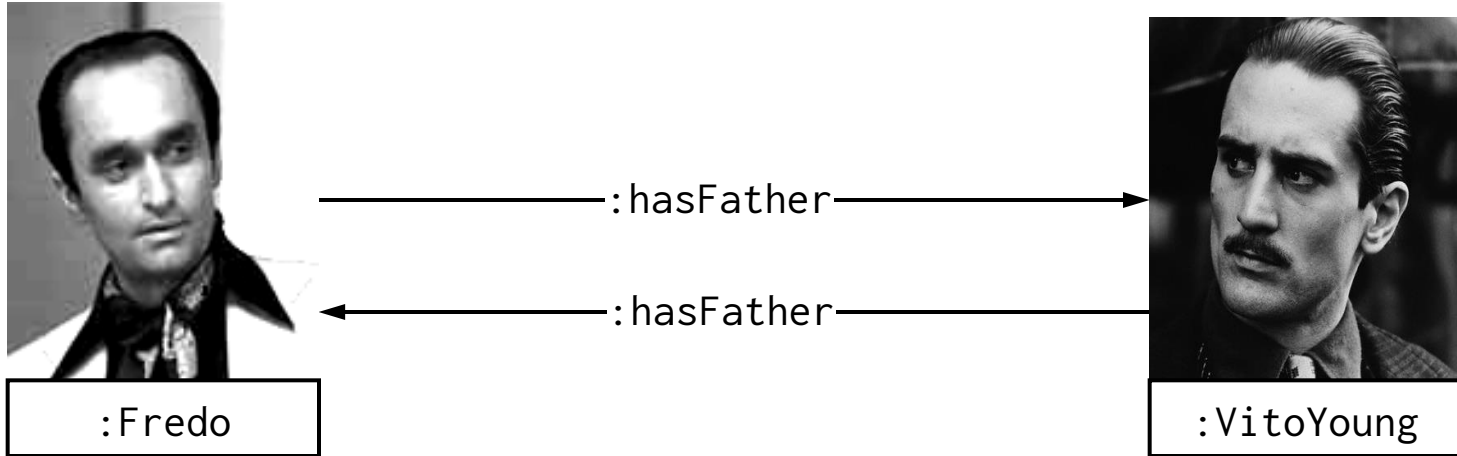
owl:IrreflexiveProperty



```
:Fredo :hasBrother :Fredo .  
:hasBrother rdf:type owl:IrreflexiveProperty .  
⇒ FALSE
```



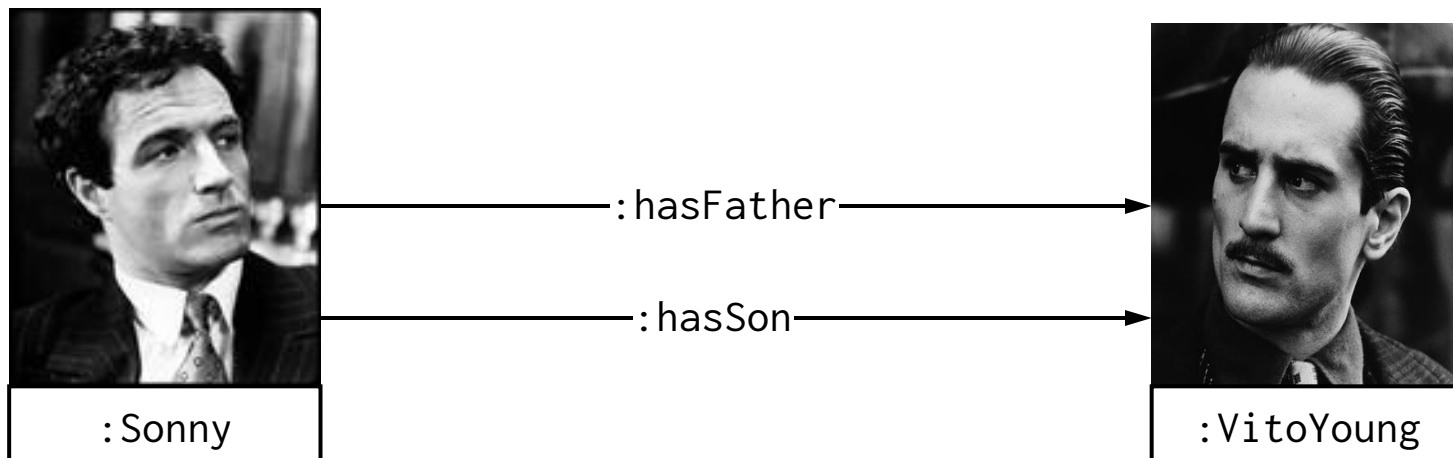
owl:AsymmetricProperty



```
:Fredo :hasFather :VitoYoung .  
:VitoYoung :hasFather :Fredo .  
:hasFather rdf:type owl:AsymmetricProperty .  
⇒ FALSE
```



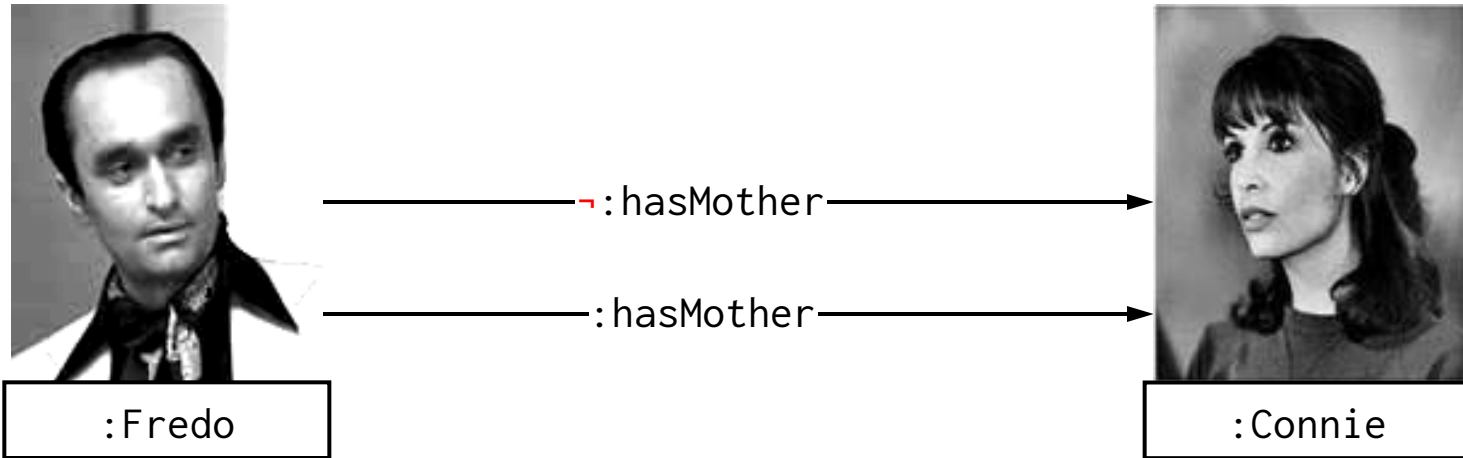
owl:propertyDisjointWith



```
:Sonny :hasFather :VitoYoung .  
:Sonny :hasSon :VitoYoung .  
:hasSon owl:propertyDisjointWith :hasFather .  
⇒ FALSE
```



NEGATIVE PROPERTY ASSERTIONS

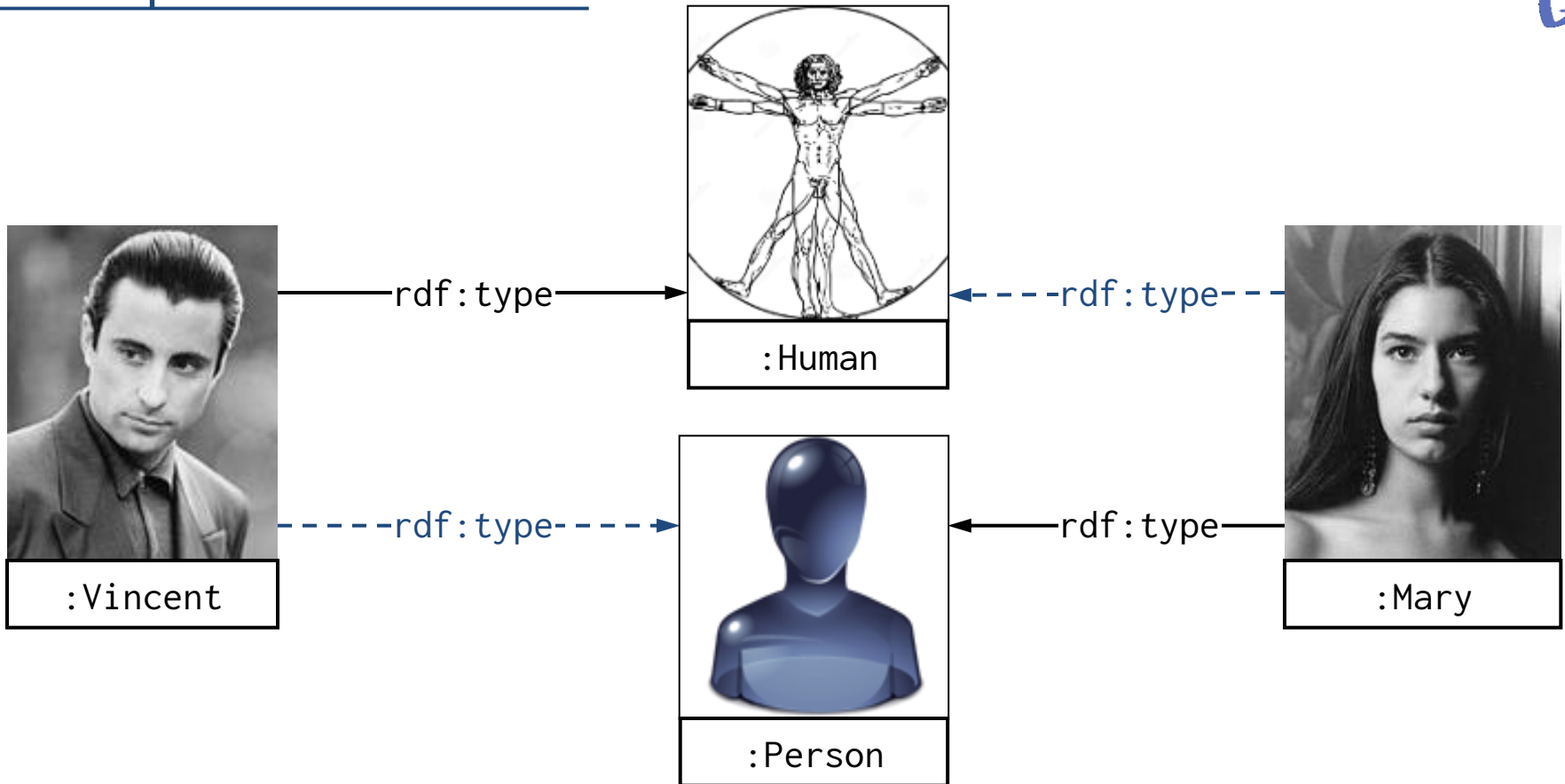


```
[ ] owl:sourceIndividual :Fredo ;  
    owl:assertionProperty :hasMother ;  
    owl:targetIndividual :Connie .  
:Fredo :hasMother :Connie .  
⇒ FALSE
```



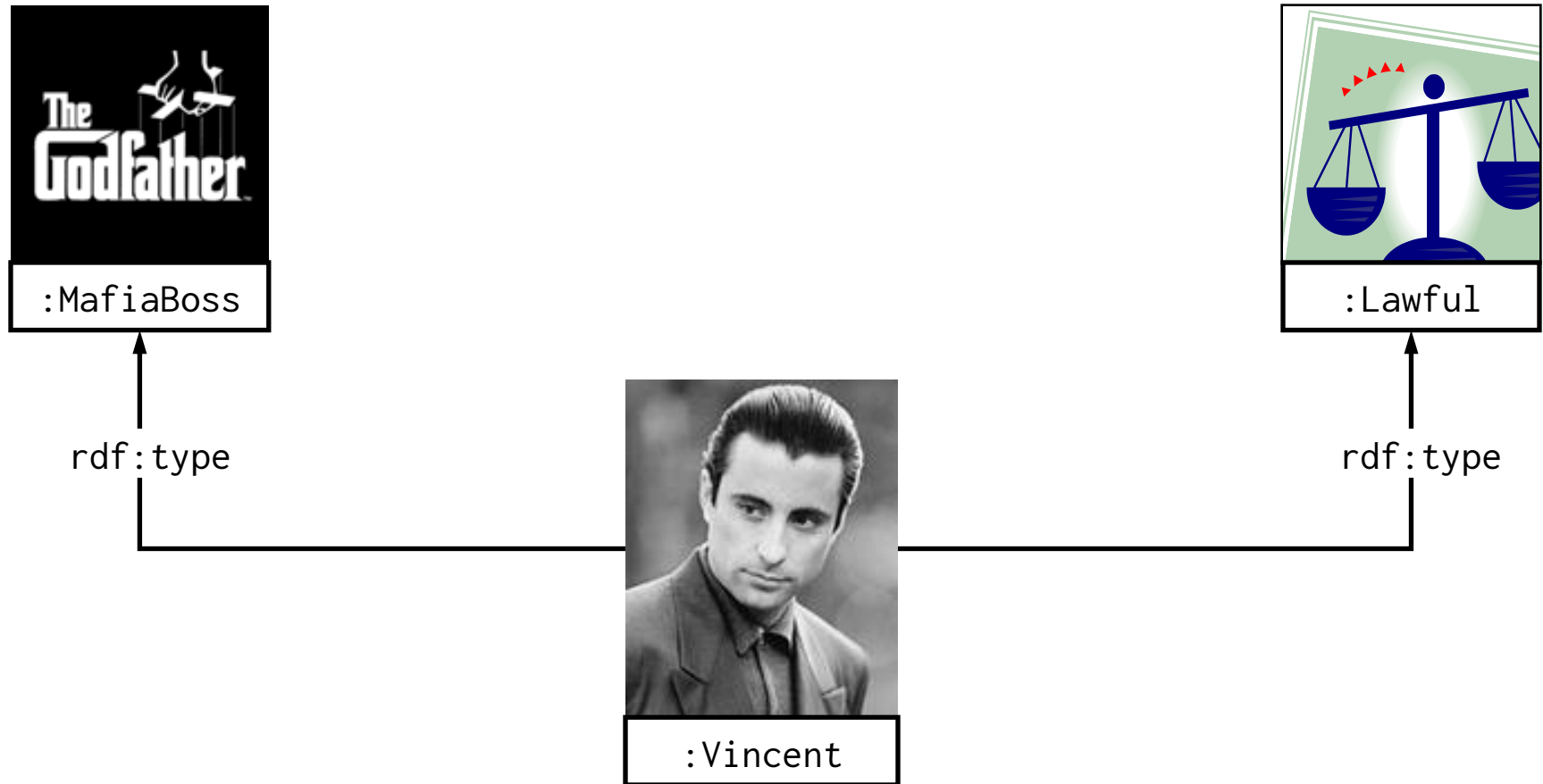
CLASS AXIOMS IN OWL

owl:equivalentClass



```
:Vincent rdf:type :Human .  
:Mary rdf:type :Person .  
:Human owl:equivalentClass :Person .  
⇒ :Vincent rdf:type :Person .  
   :Mary rdf:type :Human .
```

owl:disjointWith



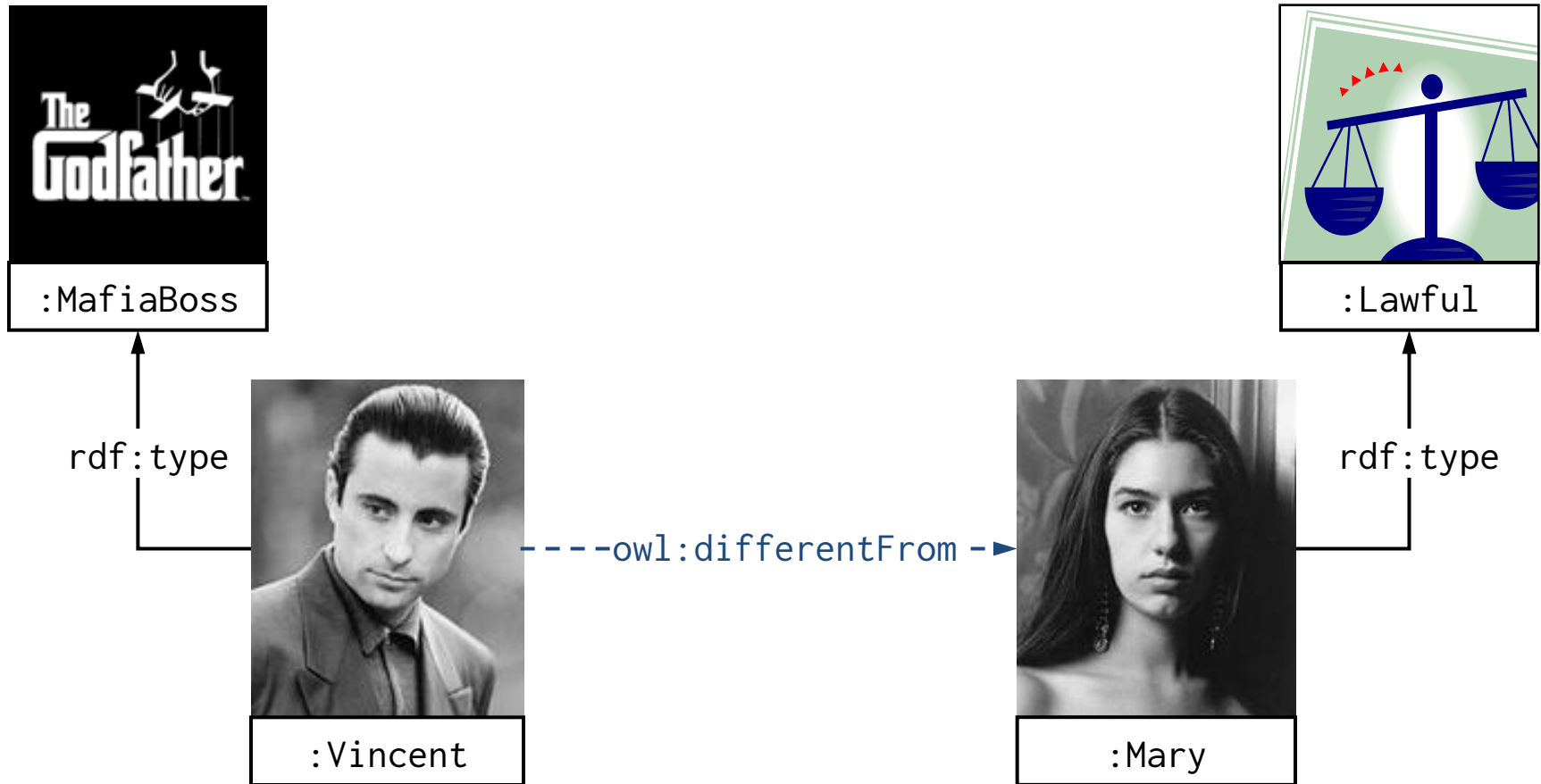
```
:Vincent rdf:type :MafiaBoss , :Lawful .
```

```
:MafiaBoss owl:disjointWith :Lawful .
```

⇒ FALSE



owl:disjointWith (II)



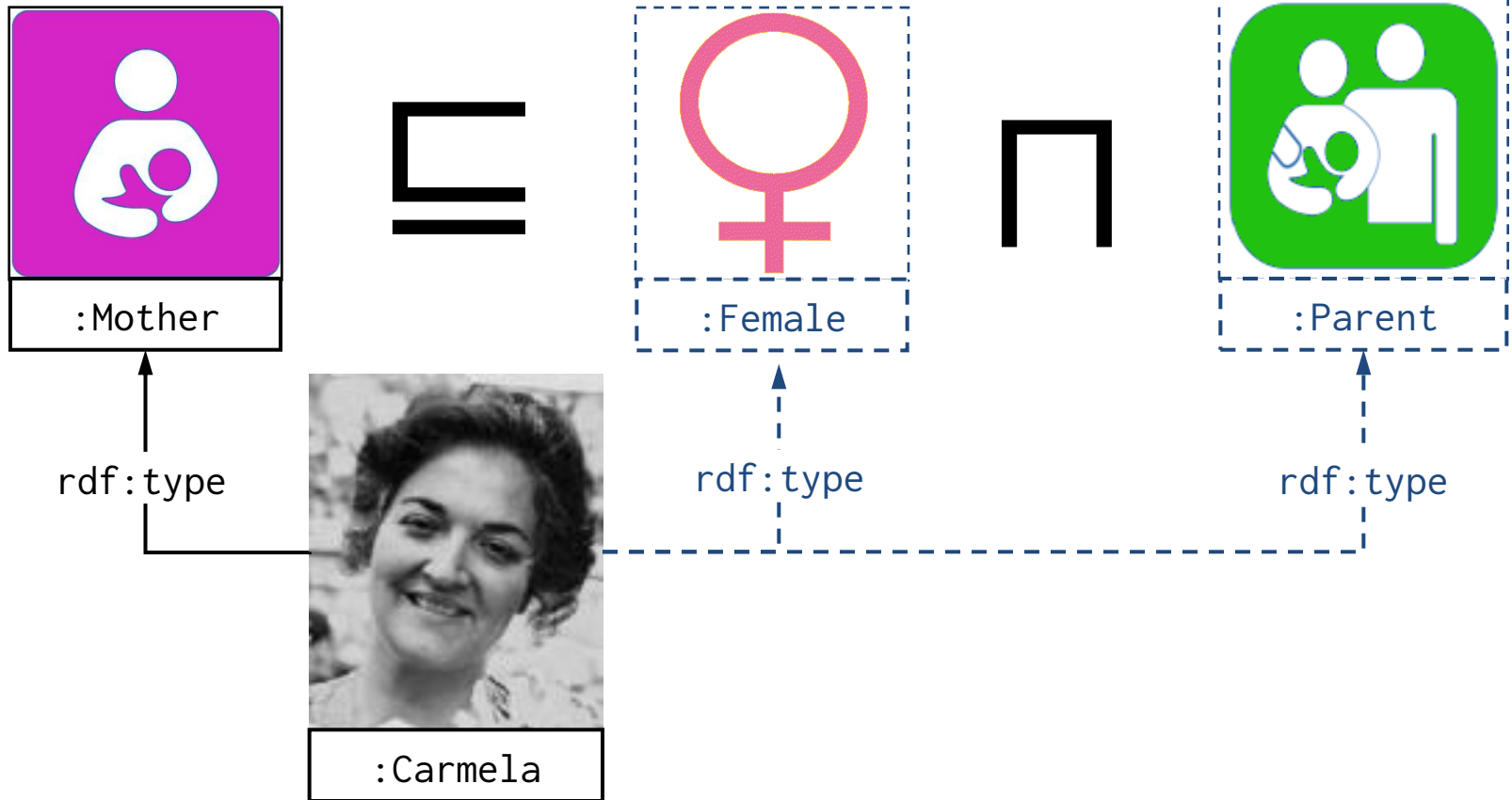
```
:Vincent rdf:type :MafiaBoss .  
:Mary rdf:type :Lawful .  
:MafiaBoss owl:disjointWith :Lawful .  
⇒ :Vincent owl:disjointWith :Mary
```


CLASS DEFINITIONS IN OWL

DESCRIPTION LOGICS

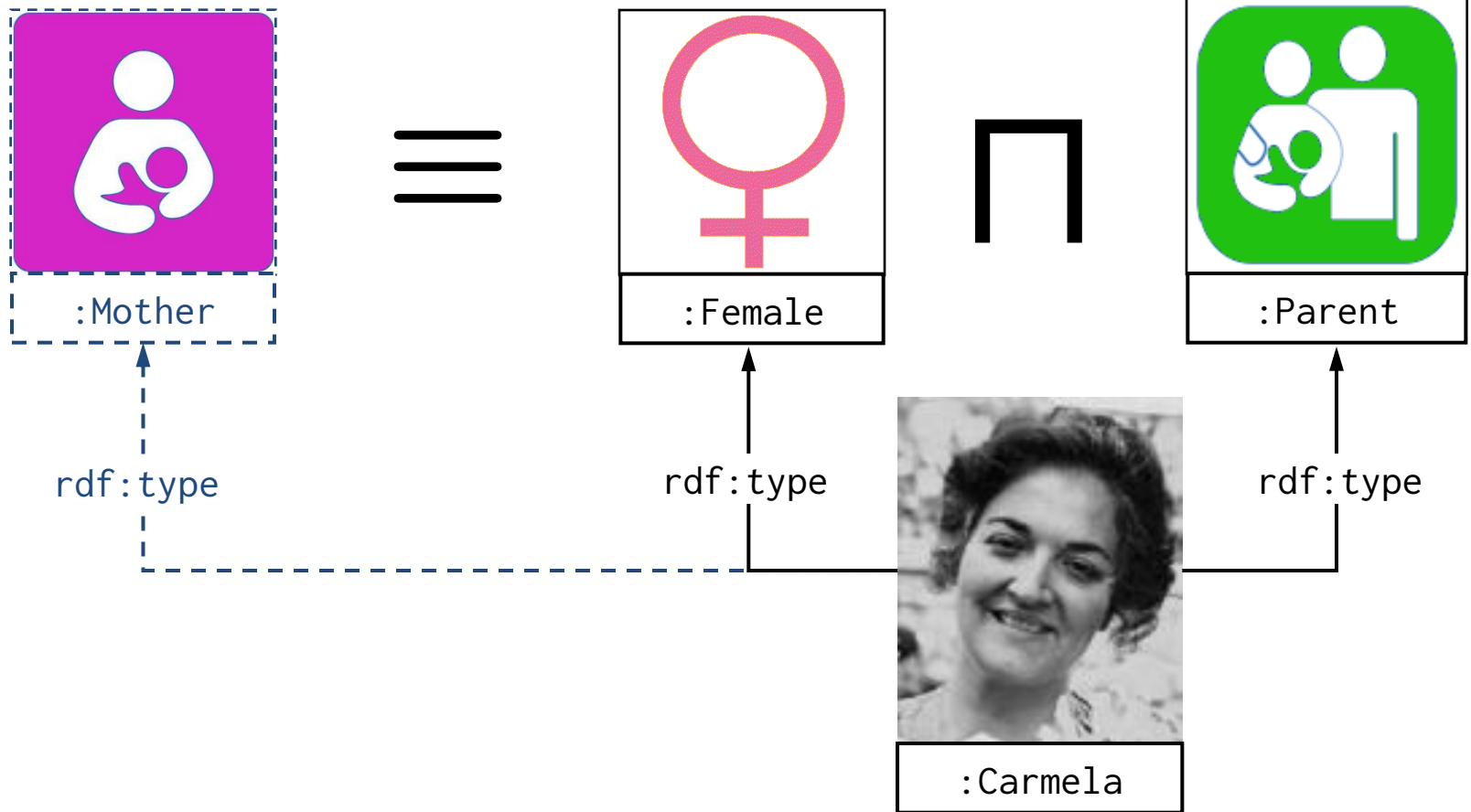
Name	Syntax	OWL key-term	DL
CONCEPT DEFINITIONS			
Atomic Concept	A	owl:Class	\mathcal{ALC}
Top Concept	\top	owl:Thing	\mathcal{ALC}
Bottom Concept	\perp	owl:Nothing	\mathcal{ALC}
Concept Negation	$\neg C$	owl:complementOf	\mathcal{ALC}
Concept Intersection	$C \sqcap D$	owl:intersectionOf	\mathcal{ALC}
Concept Union	$C \sqcup D$	owl:unionOf	\mathcal{ALC}
Nominal	$\{a_1, \dots, a_n\}$	owl:oneOf	\mathcal{O}
Existential Restriction	$\exists R.C$	owl:someValuesFrom	\mathcal{ALC}
Universal Restriction	$\forall R.C$	owl:allValuesFrom	\mathcal{ALC}
Self Restriction	$\exists R.\text{Self}$	owl:hasSelf	\mathcal{R}
Number Restriction	$\leq n R, \geq n R, = n R$	owl:*cardinality	\mathcal{N}
Qualified Number Restriction	$\leq n R.C, \geq n R.C, = n R.C$	owl:*qualifiedCardinality	\mathcal{Q}
CONCEPT AXIOMS (T-Box)			
Concept Inclusion	$C \sqsubseteq D$	rdfs:subClassOf	\mathcal{ALC}
ROLE DEFINITIONS			
Role	R	owl:*Property	\mathcal{ALC}
Inverse Role	R^-	owl:inverseOf	\mathcal{I}
Universal Role	U	owl:top*Property	\mathcal{R}
ROLE AXIOMS (R-Box)			
Role Inclusion	$R \sqsubseteq S$	rdfs:subPropertyOf	\mathcal{H}
Complex Role Inclusion	$R_1 \circ \dots \circ R_n \sqsubseteq S$	owl:propertyChainAxiom	\mathcal{R}
Transitive Roles	Trans(R)	owl:TransitiveProperty	\mathcal{S}
Functional Roles	Func(R)	owl:FunctionalProperty	\mathcal{F}
Reflexive Roles	Ref(R)	owl:ReflexiveProperty	\mathcal{R}
Irreflexive Roles	Irref(R)	owl:IrreflexiveProperty	\mathcal{R}
Symmetric Roles	Sym(R)	owl:SymmetricProperty	\mathcal{I}
Asymmetric Roles	Asym(R)	owl:AsymmetricProperty	\mathcal{R}
Disjoint Roles	Disj(R, S)	owl:disjointPropertyWith	\mathcal{R}
ASSERTIONAL DEFINITIONS			
(Named) Individual	a	(RDF IRI or Literal)	\mathcal{ALC}
ASSERTIONAL AXIOMS (A-Box)			
Role Assertion	$R(a, b)$	(RDF triple)	\mathcal{ALC}
Negative Role Assertion	$\neg R(a, b)$	owl:NegativePropertyAssertion	\mathcal{ALC}
Concept Assertion	$C(a)$	rdf:type	\mathcal{ALC}
Equality	$a = b$	owl:sameAs	\mathcal{ALC}
Inequality	$a \neq b$	owl:differentFrom	\mathcal{ALC}

owl:intersectionOf (\cap) [I]



```
:Carmela rdf:type :Mother .  
:Mother rdfs:subClassOf [ owl:intersectionOf ( :Female :Parent ) ]  
⇒ :Carmela rdf:type :Female , :Parent .
```

owl:intersectionOf (\cap) [II]



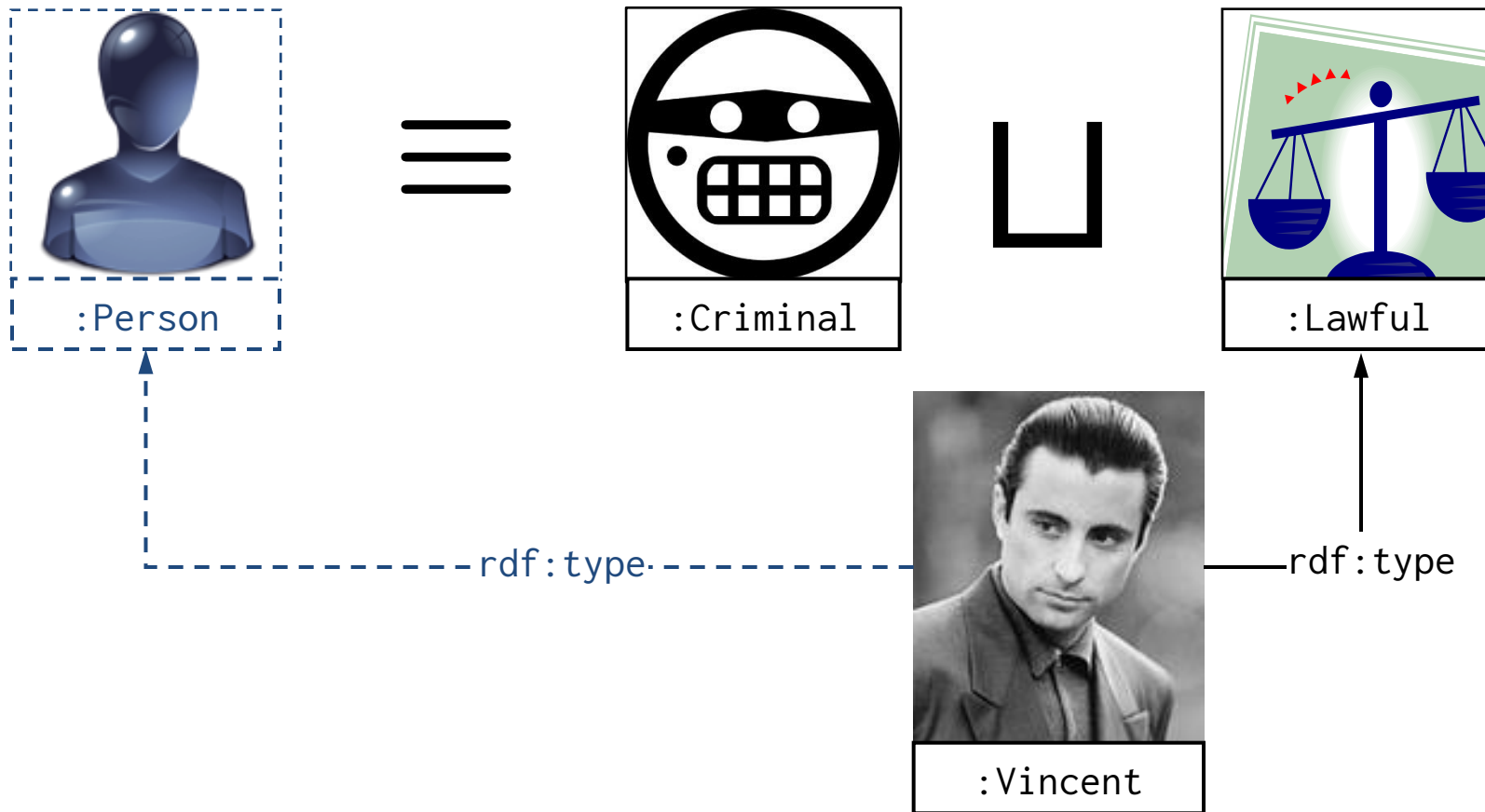
```
:Carmela rdf:type :Female , :Parent .
```

```
:Mother owl:equivalentClass [ owl:intersectionOf ( :Female :Parent ) ]
```

```
⇒ :Carmela rdf:type :Mother .
```

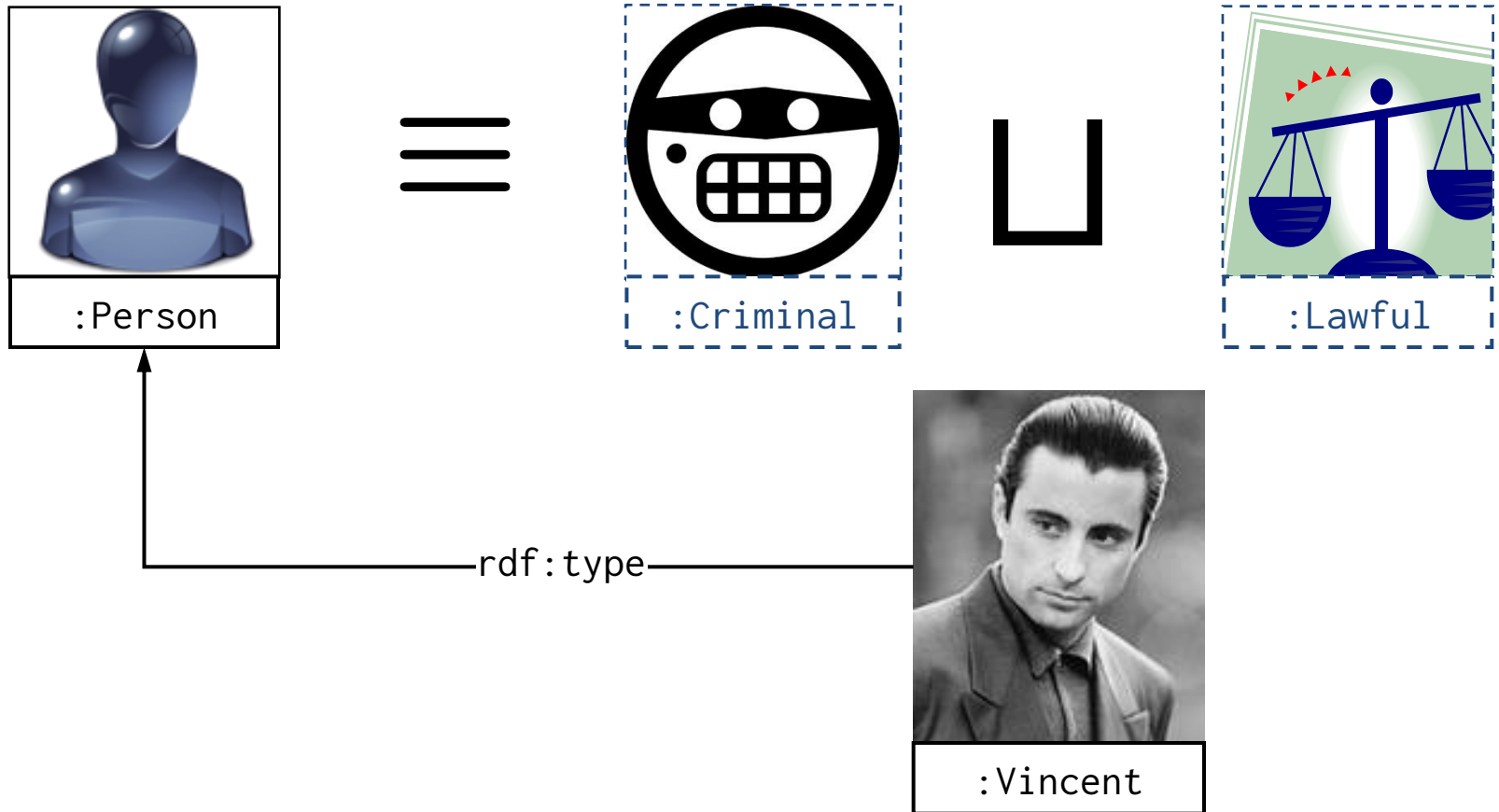


owl:unionOf (\sqcup) [I]



```
:Vincent rdf:type :Lawful .  
:Person owl:equivalentClass [ owl:unionOf ( :Criminal :Lawful ) ]  
⇒ :Vincent rdf:type :Person .
```

owl:unionOf (\sqcup) [II]



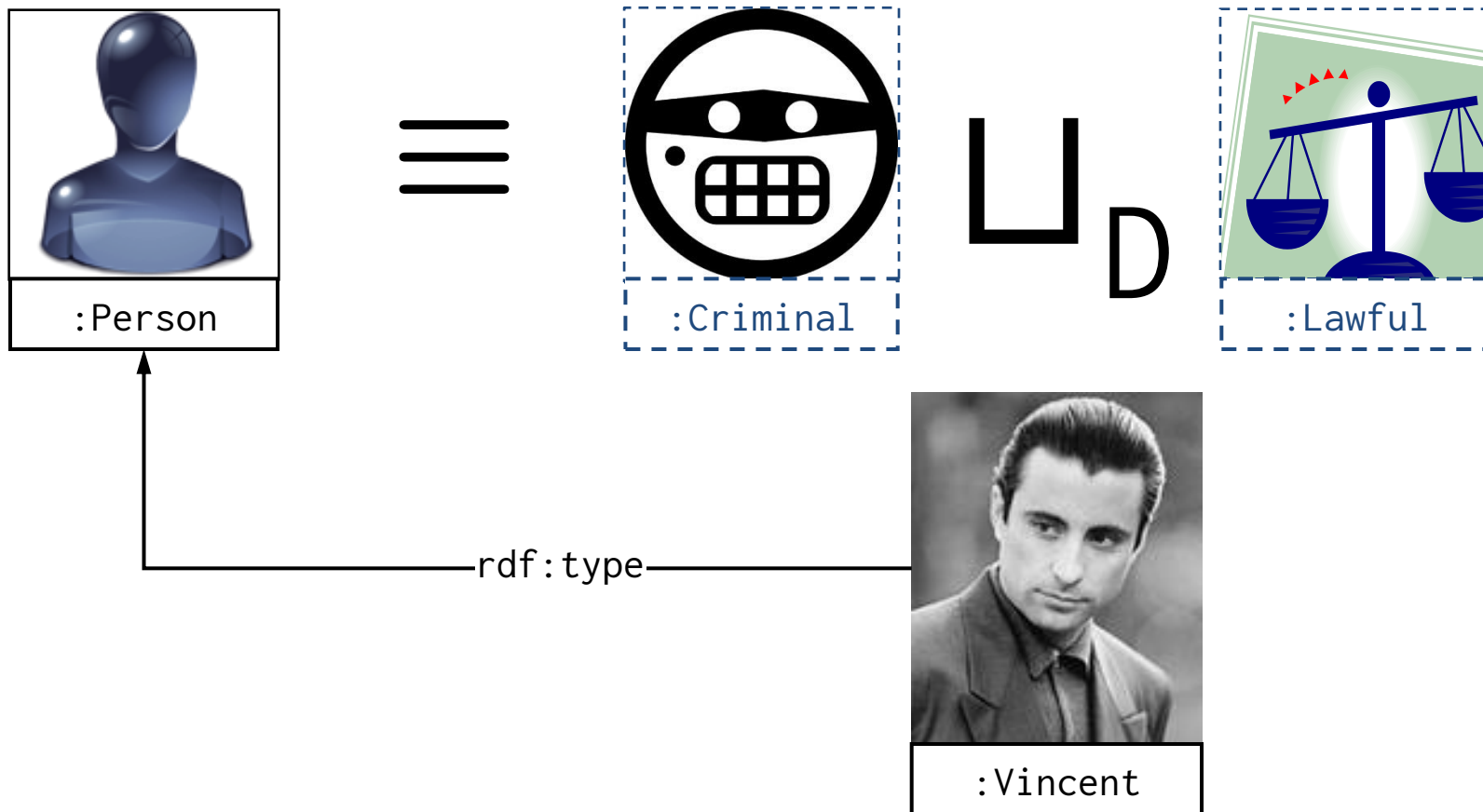
```
:Vincent rdf:type :Person .
```

```
:Person owl:equivalentClass [ owl:unionOf ( :Criminal :Lawful ) ]
```

```
⇒ # :Vincent must be either :Lawful or :Criminal (or both)
```



owl:disjointUnionOf (\sqcup_D)



```
:Vincent rdf:type :Person .
```

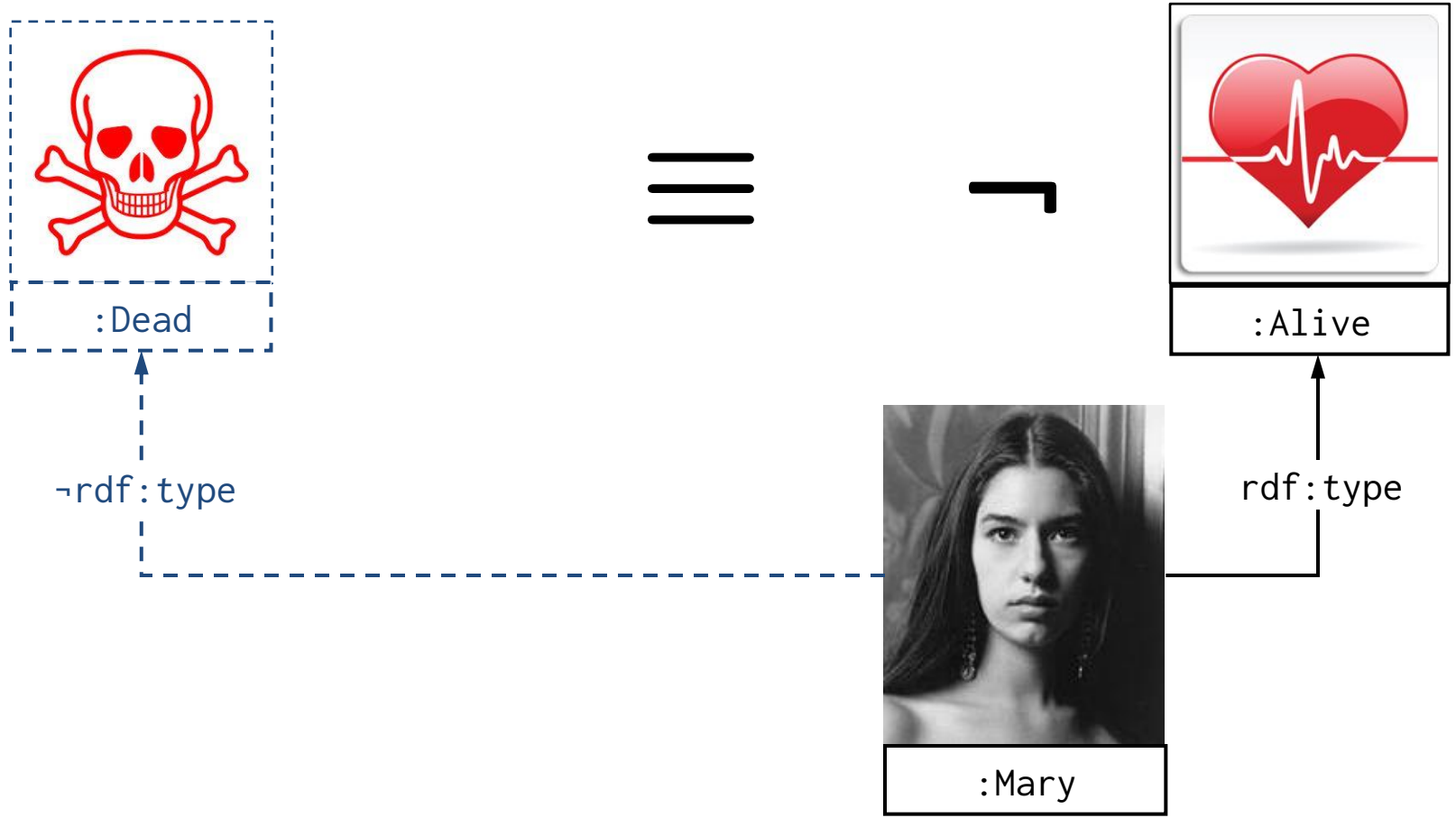
```
:Person owl:equivalentClass
```

```
  [ owl:disjointUnionOf ( :Criminal :Lawful ) ]
```

```
⇒ # :Vincent must be either :Lawful or :Criminal (not both)
```



owl:complementOf (\neg) [I]



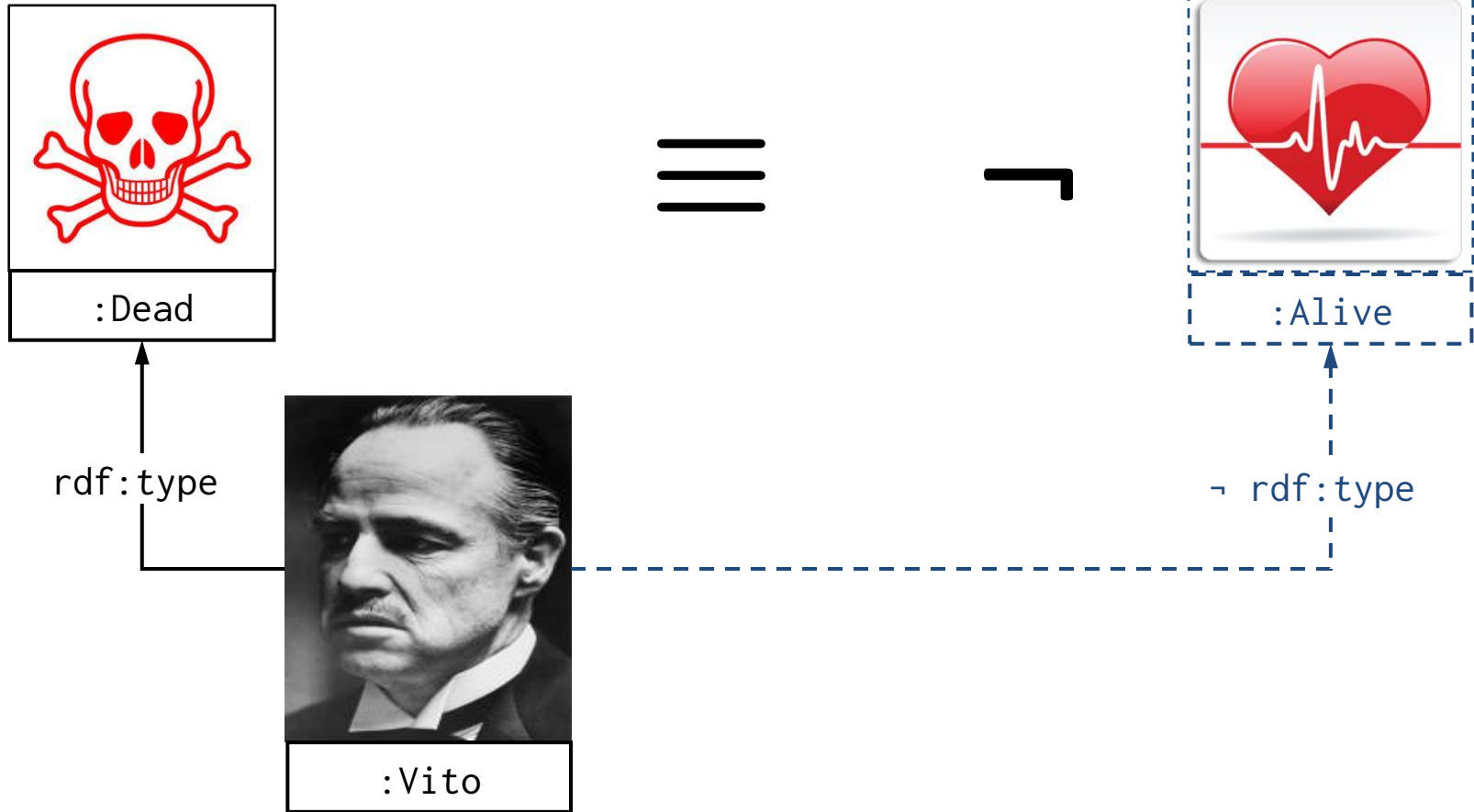
```

:Mary rdf:type :Alive .
:Dead owl:equivalentClass [ owl:complementOf :Alive ]
⇒ [] owl:sourceIndividual :Mary ; owl:targetProperty rdf:type ;
    owl:targetIndividual :Dead .

```




owl:complementOf (\neg) [II]



```
:Vito rdf:type :Dead .
```

```
:Dead owl:equivalentClass [ owl:complementOf :Alive ]
```

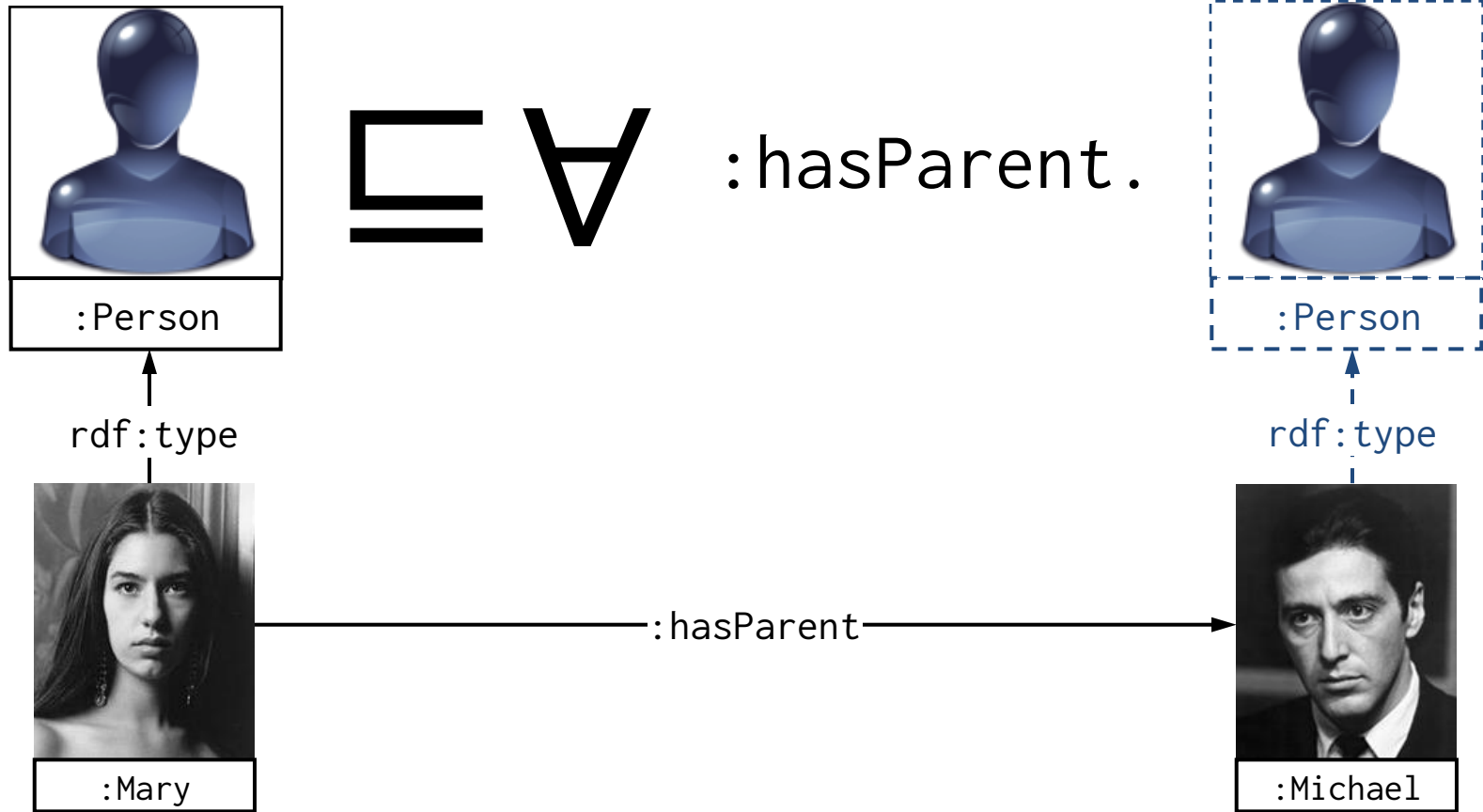
```
⇒ [] owl:sourceIndividual :Vito ; owl:targetProperty rdf:type ;  
    owl:targetIndividual :Alive .
```

owl:oneOf ({})



```
:Godfather owl:equivalentClass
  [ owl:oneOf (:Vito :Michael :Vincent) ]
⇒ :Vito rdf:type :Godfather .
⇒ :Michael rdf:type :Godfather .
⇒ :Vincent rdf:type :Godfather .
```

owl:allValuesFrom (\forall) [I]



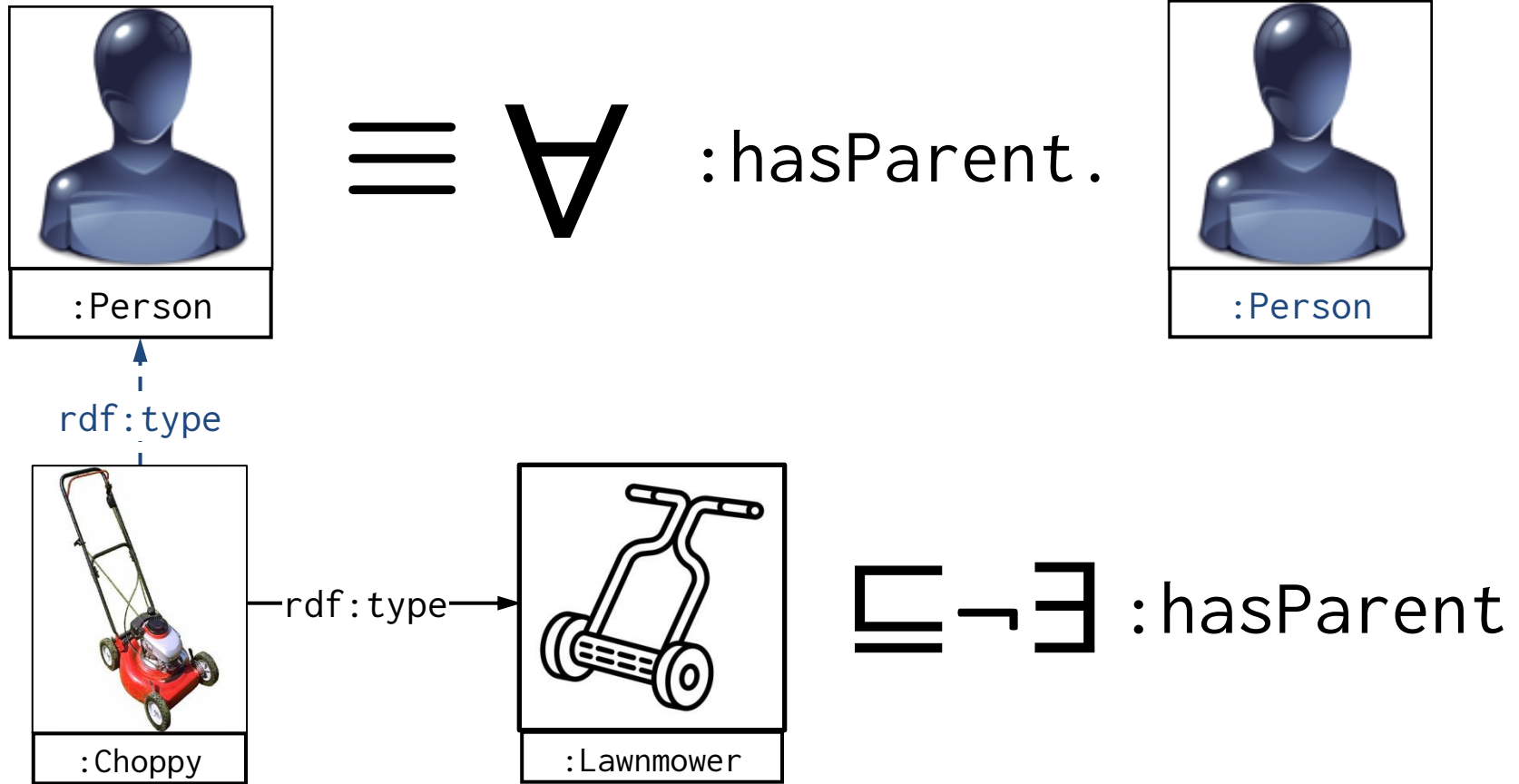
```
:Mary rdf:type :Person ; :hasParent :Michael .
```

```
:Person rdfs:subClassOf
```

```
[ owl:allValuesFrom :Person ; owl:onProperty :hasParent ]
```

```
⇒ :Michael rdf:type :Person .
```

owl:allValuesFrom (\forall) [II]



Need to be careful using \equiv with owl:allValuesFrom!

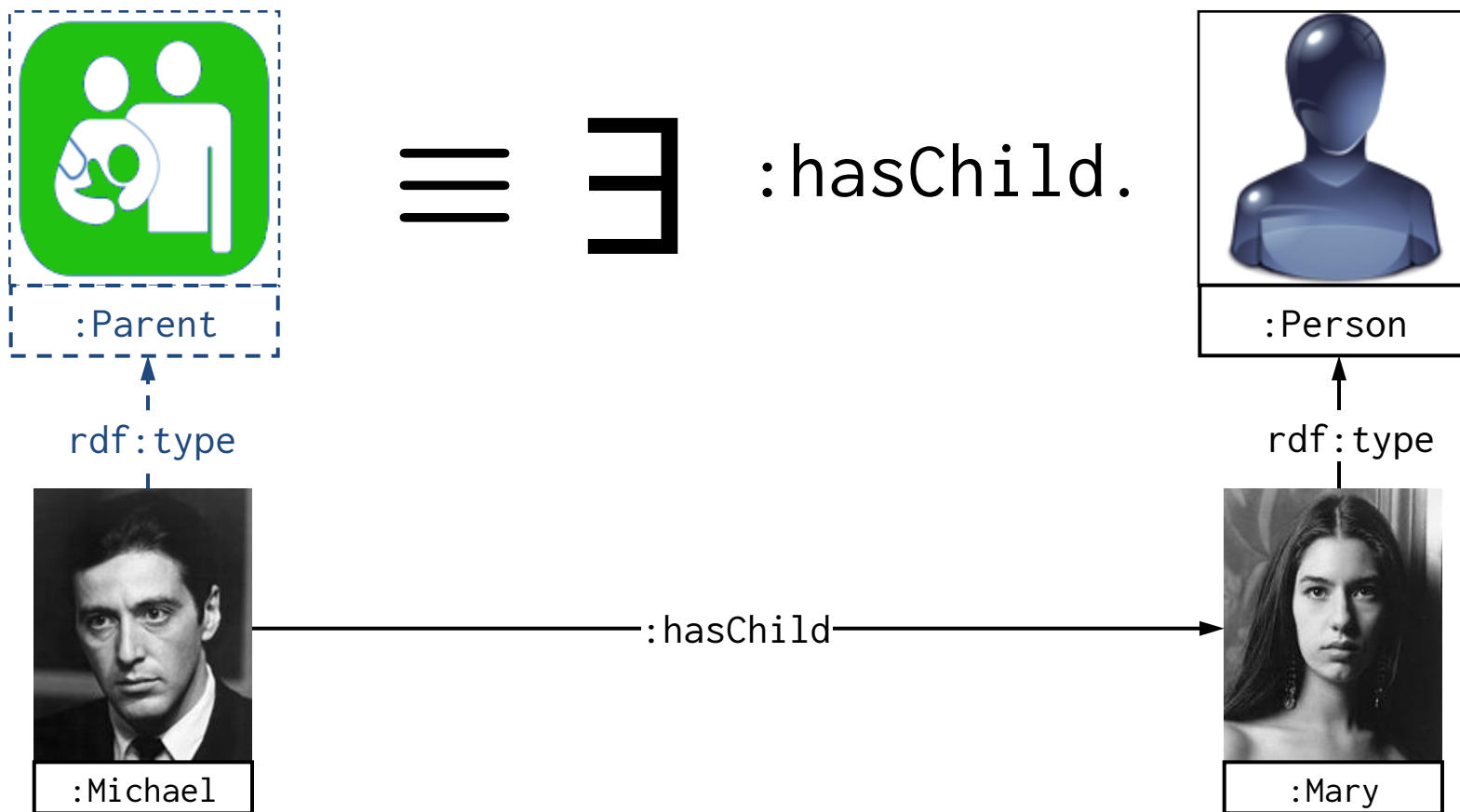
If all of X's parents are :Persons, X is a :Person.

But :Choppy does not have parents, so all its parents are trivially anything!

Thus :Choppy must be a :Person!



owl:someValuesFrom (∃) [1]



```
:Michael :hasChild :Mary . :Mary rdf:type :Person .
```

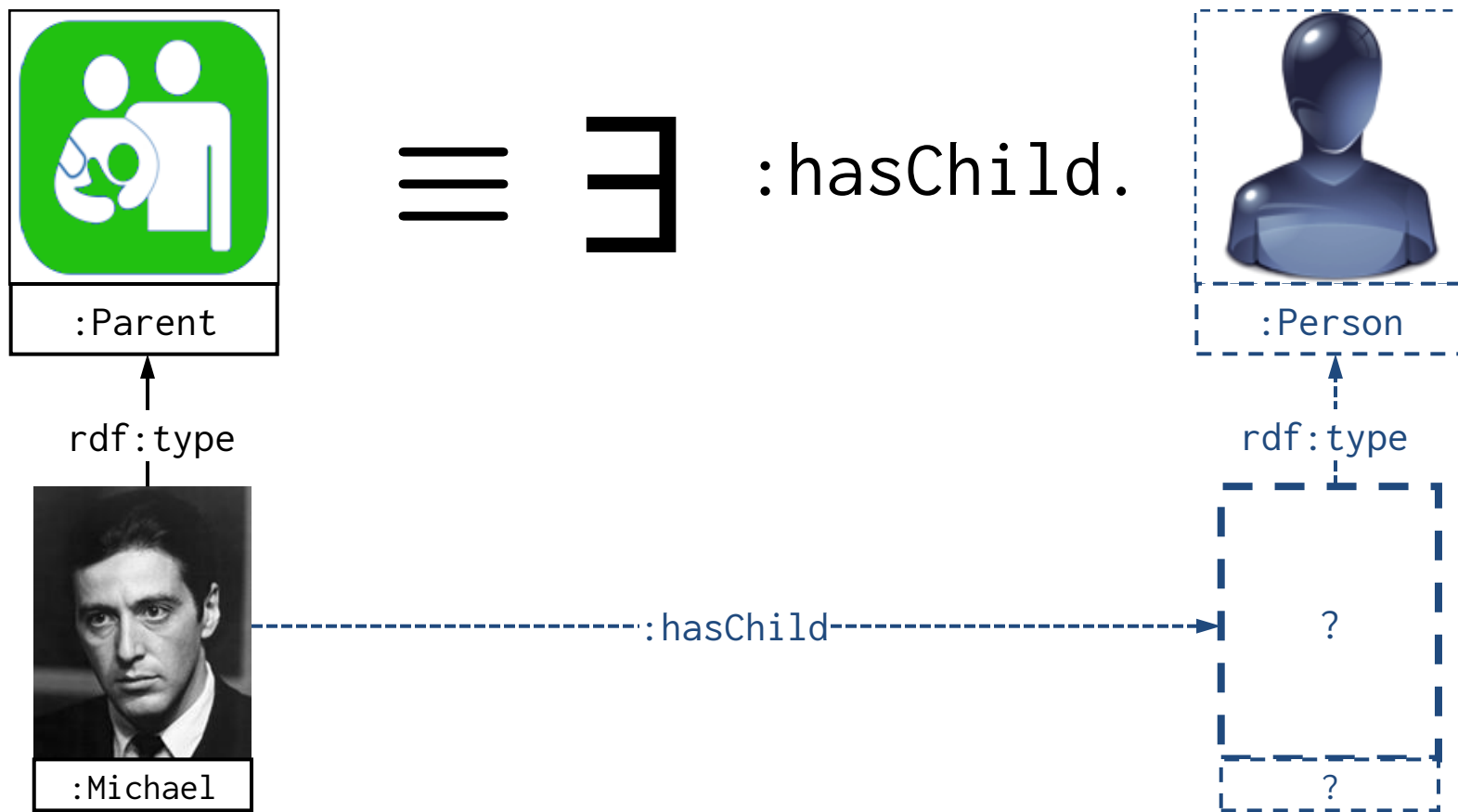
```
:Parent owl:equivalentClass
```

```
[ owl:someValuesFrom :Person ; owl:onProperty :hasChild ]
```

```
⇒ :Michael rdf:type :Parent .
```



owl:someValuesFrom (\exists) [II]



```
:Michael rdf:type :Parent .
```

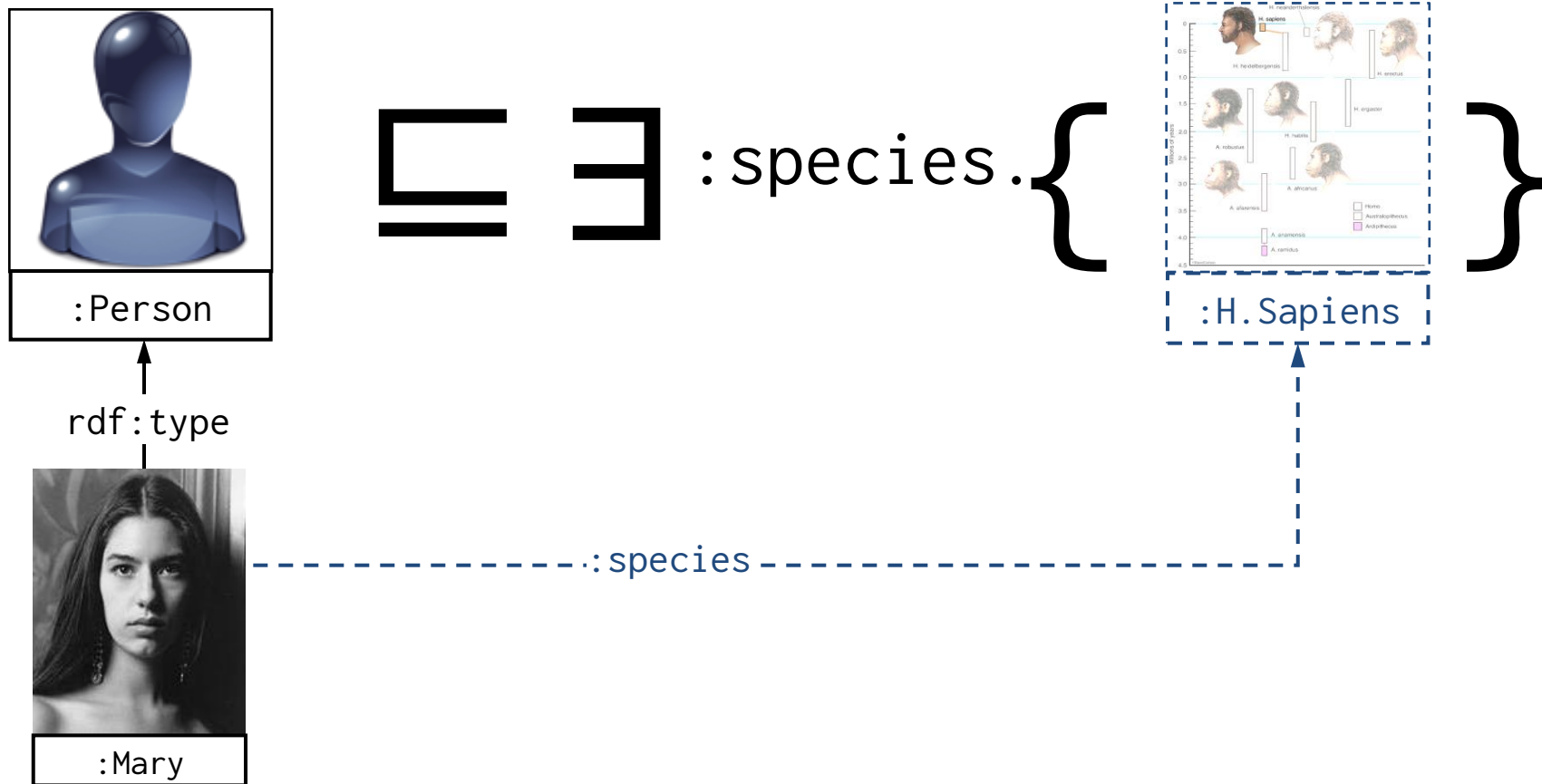
```
:Parent owl:equivalentClass
```

```
[ owl:someValuesFrom :Person ; owl:onProperty :hasChild ]
```

```
⇒ :Michael :hasChild _:someone . _:someone rdf:type :Person .
```



owl:hasValue ($\exists P.\{x\}$) [I]



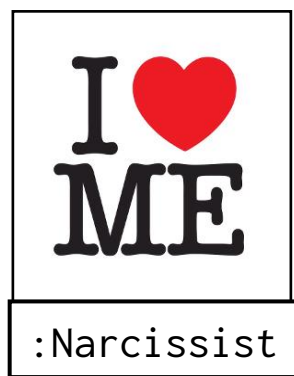
`:Mary rdf:type :Person .`

`:Person rdfs:subClassOf`

`[owl:hasValue :H.Sapiens ; owl:onProperty :species]`

\Rightarrow `:Mary :species :H.Sapiens .`

owl:hasSelf (Self) [I]



Self(*loves*)

rdf:type



:loves

```

:Michael rdf:type :Narcissist .
:Narcissist rdfs:subClassOf
  [ owl:hasSelf true ; owl:onProperty :loves ]
⇒ :Michael :loves :Michael .
  
```

owl:hasSelf (Self) [III]



≡

Self(:loves)

rdf:type



:loves

:Michael

```
:Michael :loves :Michael .
```

```
:Narcissist owl:equivalentClass
```

```
  [ owl:hasSelf true ; owl:onProperty :loves ]
```

```
⇒ :Michael rdf:type Narcissist .
```




QUALIFIED CARDINALITY RESTRICTIONS (\geq , \leq , $=$)

- Define a class with a given number of values from a given class for a property:

- **Exact:** `:Biped \equiv =2 (:hasLimb.Leg)`

```
:Biped owl:equivalentClass [ owl:qualifiedCardinality 2 ;  
                               owl:onProperty :hasLimb ;  
                               owl:onClass :Leg ] .
```

- **Max:** `:Mafia \sqsubseteq \leq 1 (:hasCurrentMember.Godfather)`

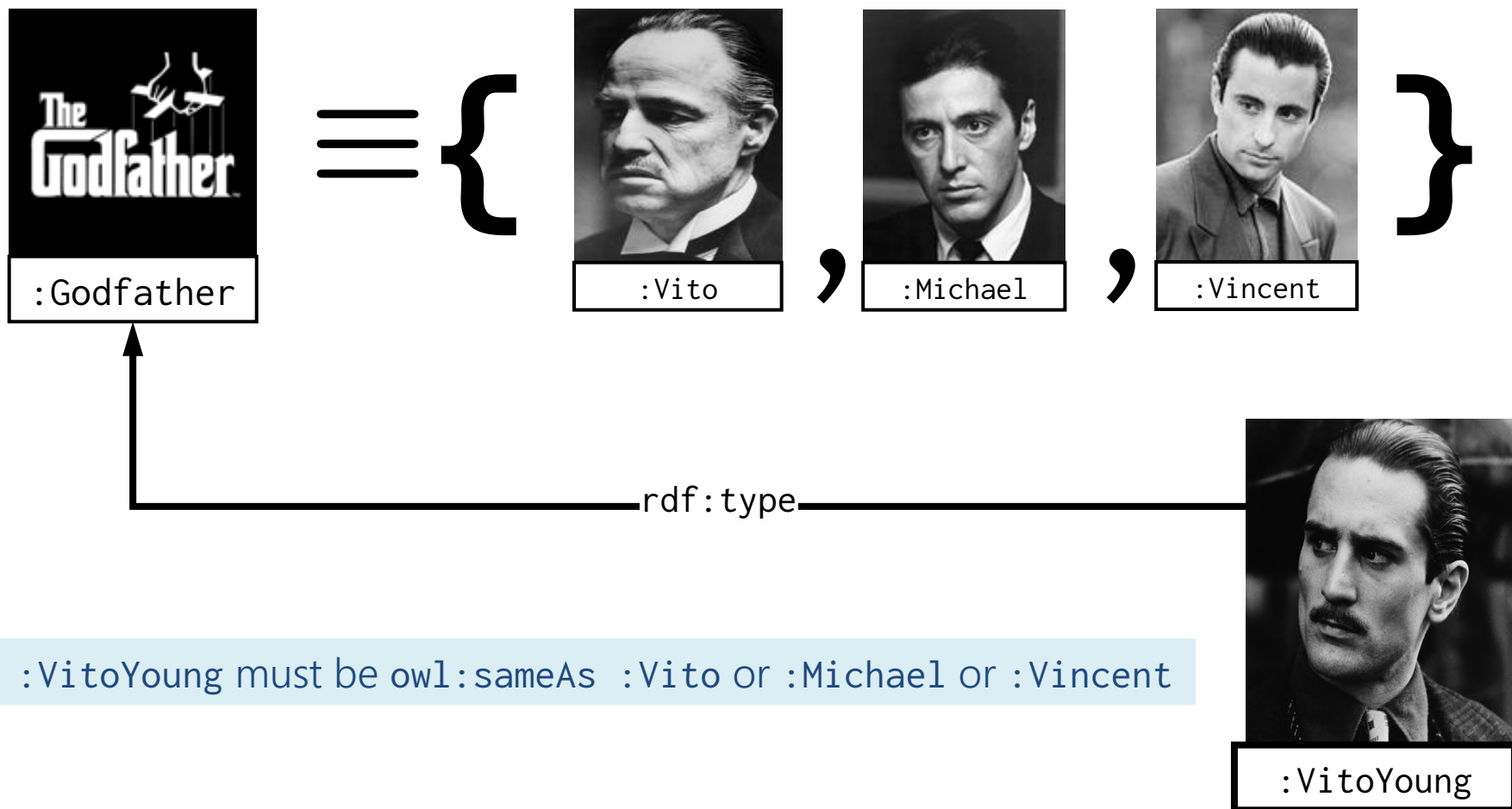
```
:Mafia rdfs:subClassOf [ owl:maxQualifiedCardinality 1 ;  
                          owl:onProperty :hasCurrentMember ;  
                          owl:onClass :Godfather ] .
```

- **Min:** `:Twin \sqsubseteq \geq 1 (:sibling.Twin)`

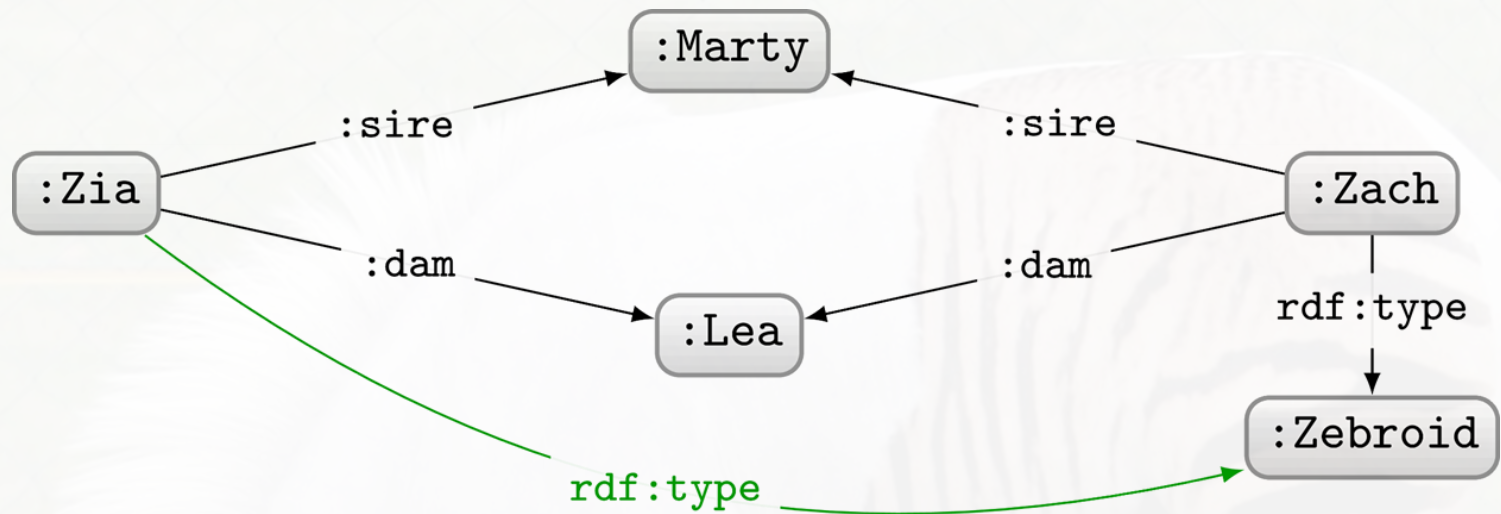
```
:Twin rdfs:subClassOf [ owl:minQualifiedCardinality 1 ;  
                          owl:onProperty :sibling ;  
                          owl:onClass :Twin ] .
```



SLIDES ARE EXAMPLES, NOT DEFINITIONS



⇒ :VitoYoung must be owl:sameAs :Vito or :Michael or :Vincent



- sire is a sub-property of parent
- dam is a sub-property of parent
- A Zebroid has exactly one parent a Zebra
- A Zebroid has exactly one parent a (-Zebra and a Equine)
- A Zebroid is a sub-class of Equine
- An Equine has exactly two parents
- Two things cannot be related by sire and dam at the same time

An iceberg floating in the ocean. The tip of the iceberg is visible above the water surface, while the much larger, submerged part is visible below. The sky is blue with light clouds, and the water is a deep blue. The text '← RDFS' is positioned to the right of the visible tip of the iceberg.

← RDFS

A cartoon illustration of a brown owl wearing a blue snorkel mask and a blue snorkel tube. The owl is looking forward with large, yellow eyes. The background is dark blue.

← OWL

QUESTIONS?

