

DR. MIGUEL CERIANI

PROF. ALEJANDRO VAISMAN

ONTOLOGÍAS EN LA WEB

6. RDFS Y INFERENCIA

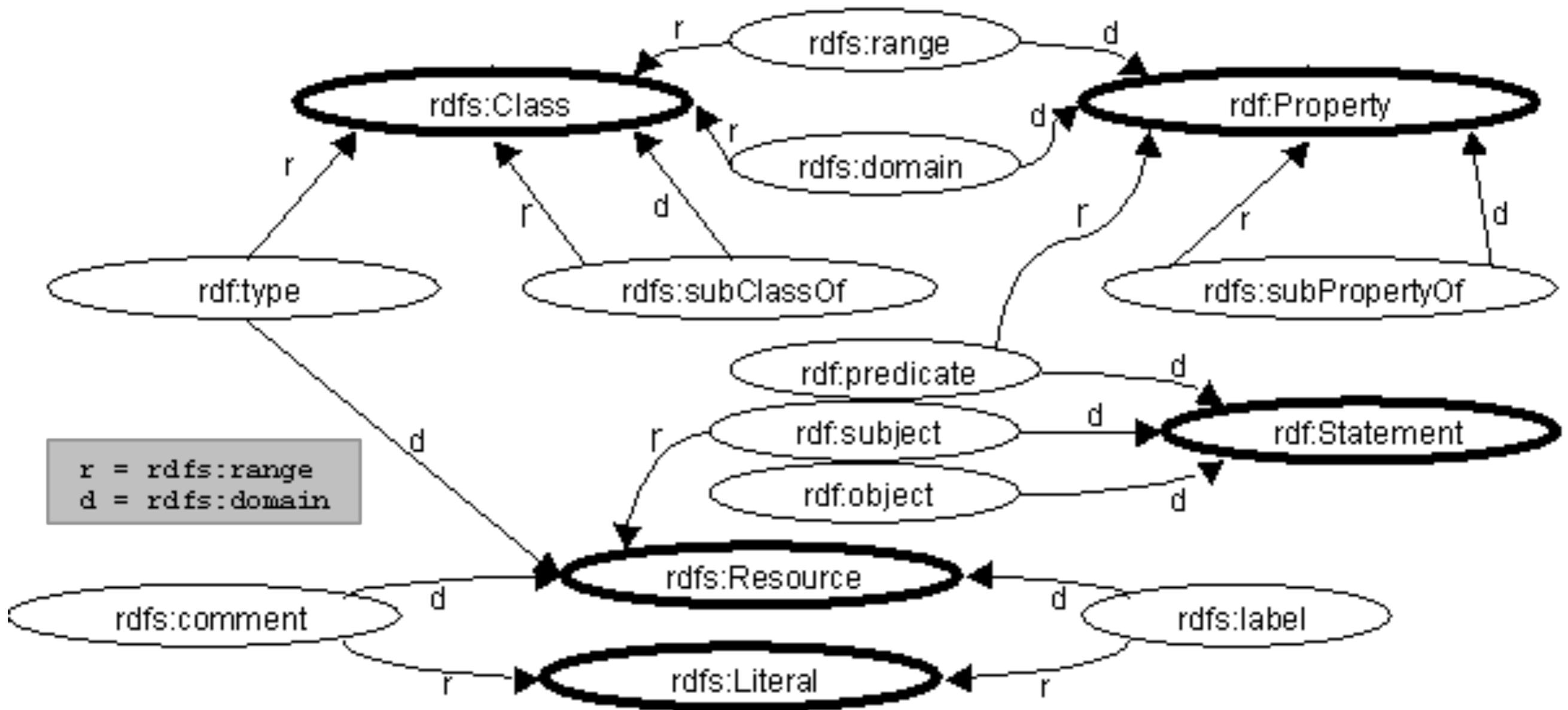
REFERENCIAS PARA ESTA CLASE (RDFS)

- ▶ Caps. 6 y 7 de "Semantic for the Working Ontologist"
- ▶ [W3C RDF 1.1 Schema](#)
- ▶ Semantic University: [RDFS Introduction](#)

REPASO RDFS

- ▶ Clases: `rdfs:Resource`, `rdfs:Class`, `rdfs:Literal`, `rdf:Property`
- ▶ Propiedades: `rdf:type`, `rdfs:subClassOf`, `rdfs:domain`, `rdfs:range`, `rdfs:subPropertyOf`
- ▶ Más Propiedades: `rdfs:label`, `rdfs:comment`

RELACIONES ENTRE CLASES Y PROPIEDADES RDFS



PROTÉGÉ: EDITOR DE ONTOLOGÍAS

The screenshot displays the Protégé ontology editor interface for a pizza ontology. The browser address bar shows the URL `http://www.co-ode.org/ontologies/pizza/pizza.owl`. The menu bar includes File, Edit, Reasoner, Tools, Refactor, Tabs, View, Window, and Help. The toolbar contains navigation and search icons. The main interface is divided into two panes:

- Asserted Class Hierarchy: Pizza:** A tree view showing the hierarchy of classes. The root is 'Thing', followed by 'DomainConcept', 'Country', 'Food', and 'Pizza'. Under 'Pizza', there are several subclasses including 'CheeseyPizza', 'InterestingPizza', 'MeatyPizza', 'NamedPizza', 'NonVegetarianPizza', 'RealItalianPizza', 'SpicyPizza', 'SpicyPizzaEquivalent', 'ThinAndCrispyPizza', 'VegetarianPizza', 'VegetarianPizzaEquivalent', 'PizzaBase', and 'PizzaTopping'. 'ValuePartition' is also listed at the bottom.
- OWL Viz: Pizza:** A graph view showing the relationships between classes. The 'Pizza' class is highlighted with a blue box. It is connected to various subclasses via 'is-a' relationships. The graph shows a complex network of relationships, including 'Hot', 'Medium', 'Mild', 'IceCream', 'PizzaBase', 'PizzaTopping', 'CheeseyPizza', 'ThinAndCrispyPizza', 'NamedPizza', 'SpicyPizzaEquivalent', 'VegetarianPizzaEquivalent1', 'VegetarianPizza', 'NonVegetarianPizza', 'InterestingPizza', 'QuattroFormaggi', 'Napoletana', 'Rosa', 'PolloAdAstra', 'Soho', 'Margherita', 'Fiorentina', and 'LaReine'.

WEB PROTÉGÉ: EDITOR DE ONTOLOGÍAS COLABORATIVO

The screenshot displays the WebProtégé web interface for editing ontologies. The top navigation bar includes the Protégé logo, a user profile 'ttania', and a 'Help' button. Below this, a series of tabs shows the current project 'Gene Ontology' and various views: 'Classes', 'Properties', 'Individuals', 'Notes and Discussions', 'Changes By Entity', and 'Project Dashboard'. The main workspace is divided into three panels:

- Classes Panel:** A tree view of the ontology classes. The 'cell part' class is expanded, showing a list of subclasses including 'annulate lamellae', 'apical complex', 'apical part of cell', 'ascus epiplasm', 'aspartate carbamoyltransferase cor', 'basal labyrinth', 'basal part of cell', 'basal ring of apical complex', 'CD20-Lck-Fyn complex', 'CD20-Lck-Lyn-Fyn complex', 'cell body', 'cell division site', 'cell division site part', 'cell envelope Sec protein transport c', 'cell leading edge', 'cell periphery', 'cell pole', 'cell projection', 'cell projection part', 'cell septum', and 'cell septum part'.
- Class description for annulate lamellae:** A detailed view of the selected class. It includes:
 - Display name:** annulate lamellae
 - IRI:** http://purl.obolibrary.org/obo/GO_0005642
 - Annotations:**
 - rdfs:label:** annulate lamellae
 - id:** GO:0005642
 - definition:** Stacks of endoplasmic reticulum (ER) membranes containing a high density of nuclear pores, thought to form from excess nuclear membrane components, that have been described in a number of different cells. Annulate lamellar membranes are continuous with and embedded within the ER.
 - has_obo_namespace:** cellular_component
 - Properties:**
 - part_of:** nuclear outer membrane-endoplasmic reticulum membrane network
- Discussions for annulate lamellae:** A panel for collaborative discussion, featuring a 'Post new topic' button.
- Project feed:** A panel at the bottom right for project updates.

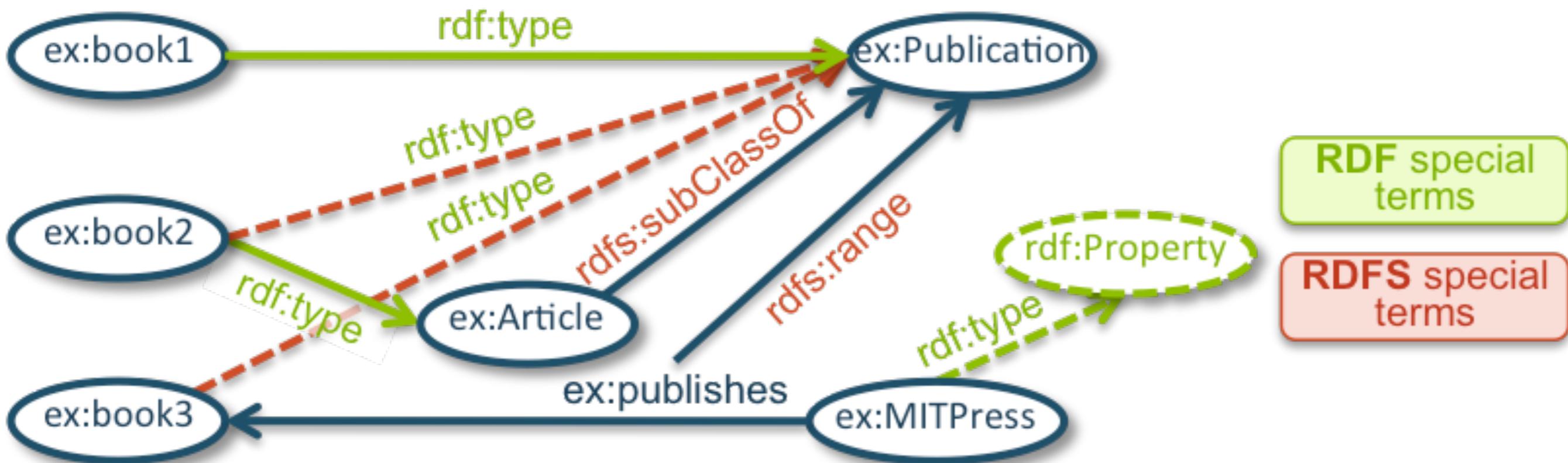
ACTIVIDAD: RDFS

- ▶ Instalar Protégé Desktop
- ▶ Crear una nueva ontología
- ▶ Definir una pequeña jerarquía de clases
- ▶ Definir algunas propiedades y caracterizarlas con domain y range
- ▶ Definir algunas entidades
- ▶ Salvar/Exportar en RDF/XML y Turtle

¿QUÉ PODEMOS HACER CON RDFS/OWL?

- ▶ evaluar la consistencia de un modelo (satisfacibilidad lógica)
- ▶ deducir nuevos hechos (implicación lógica)

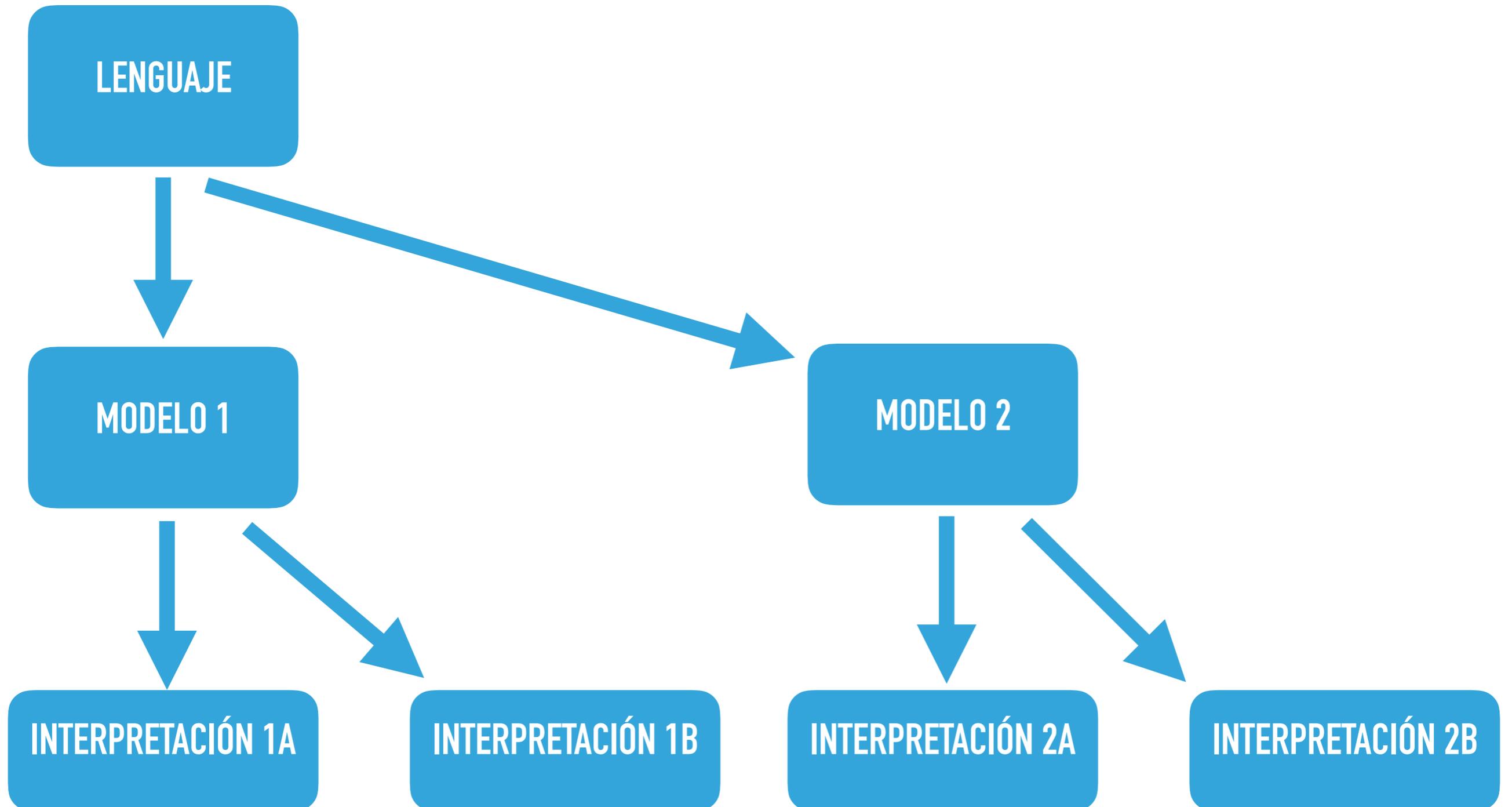
INFERENCIA



WEB SEMÁNTICA Y LÓGICA

- ▶ RDF, RDFS, OWL son lenguajes formales basados sobre IRIs, triplas y vocabularios estándar (rdf:, rdfs:, owl:)
- ▶ con estos lenguajes formales podemos definir modelos (RDF, RDFS, OWL)
- ▶ un posible significado de un modelo es una interpretación

MODELOS Y INTERPRETACIONES



EJEMPLO 1 (ÁLGEBRA)

Modelo

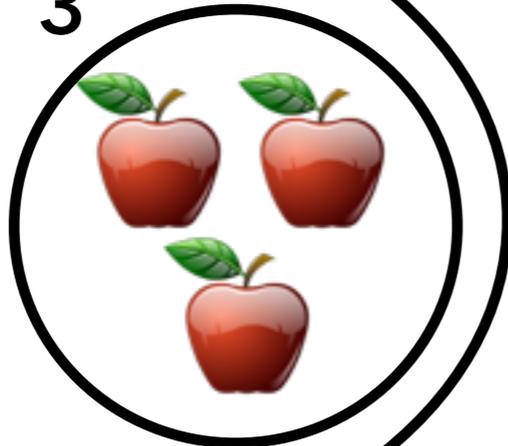
$$X = 2 + 3$$

Interpretaciones

X

2

3



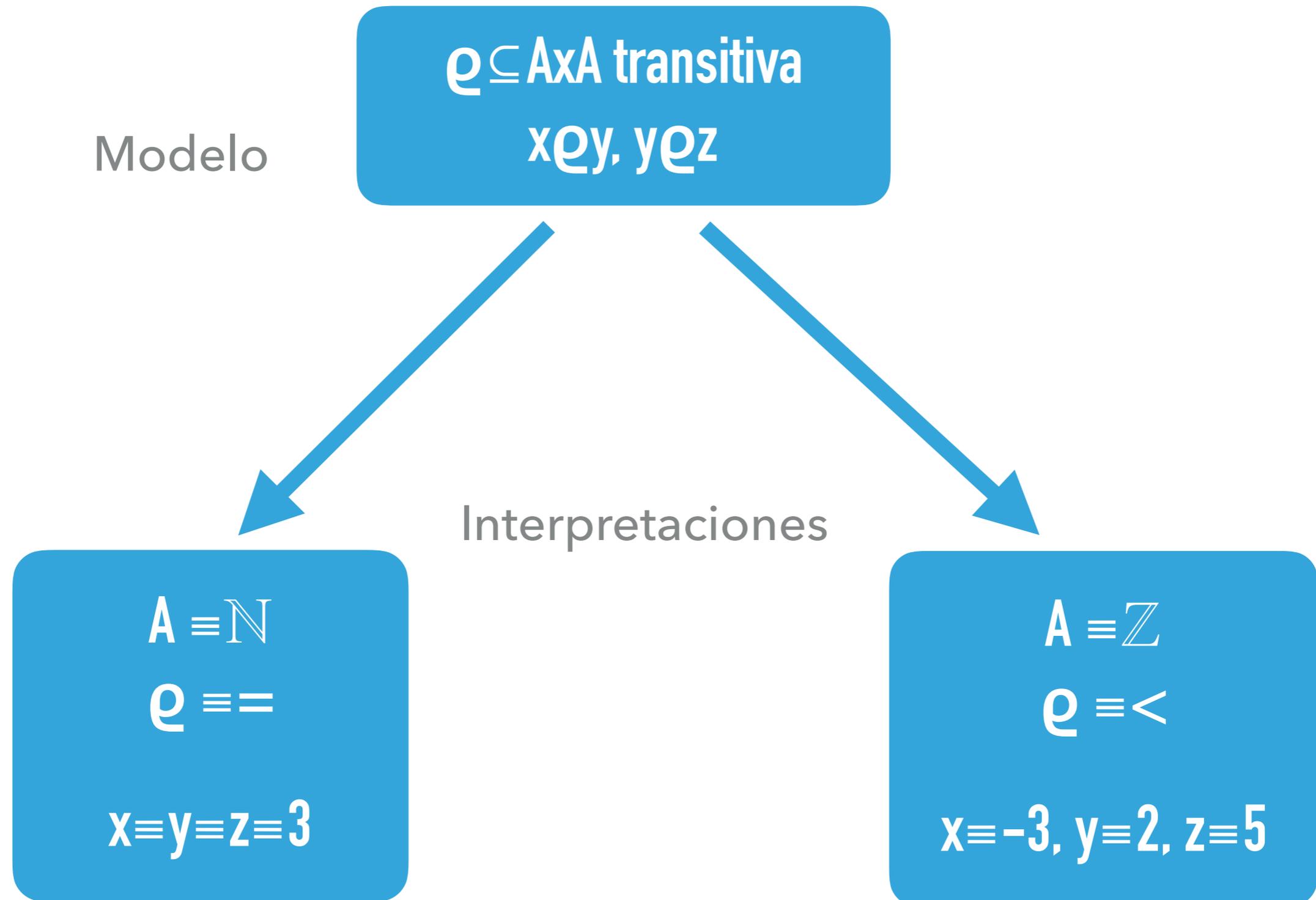
X

2

3



EJEMPLO 2 (CONJUNTOS Y RELACIONES)



INTERPRETACIONES, IMPLICACIÓN, CONSISTENCIA

- ▶ las interpretaciones tienen que ser coherentes con los vínculos que elijamos: “simples”, RDF, RDFS, conjunto de tipos de dato reconocidos...
- ▶ interpretaciones con vínculos X , se llaman “interpretaciones- X ”: interpretaciones-simples, interpretaciones-RDF, interpretaciones-RDFS,...
- ▶ un grafo RDF A “ X -implica” otro grafo RDF B si cada interpretación- X de A es una interpretación- X para B
- ▶ un grafo RDF A es “ X -satisfacible” si tiene al menos una interpretación- X .

IMPLICACIÓN: EJEMPLO 1 (ÁLGEBRA)

Modelo 1

$$X = 2 + 3$$

 \models

$$X = 5$$

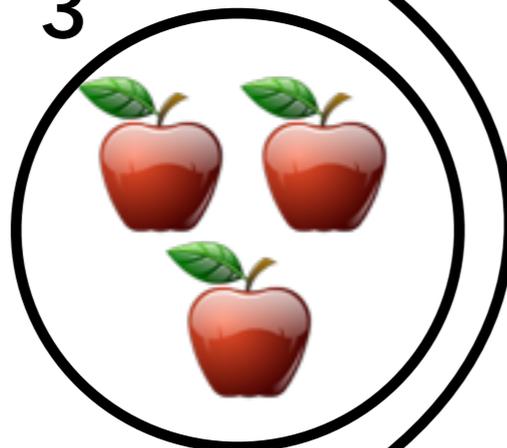
Modelo 2

Interpretaciones

 $X = 5$

2

3

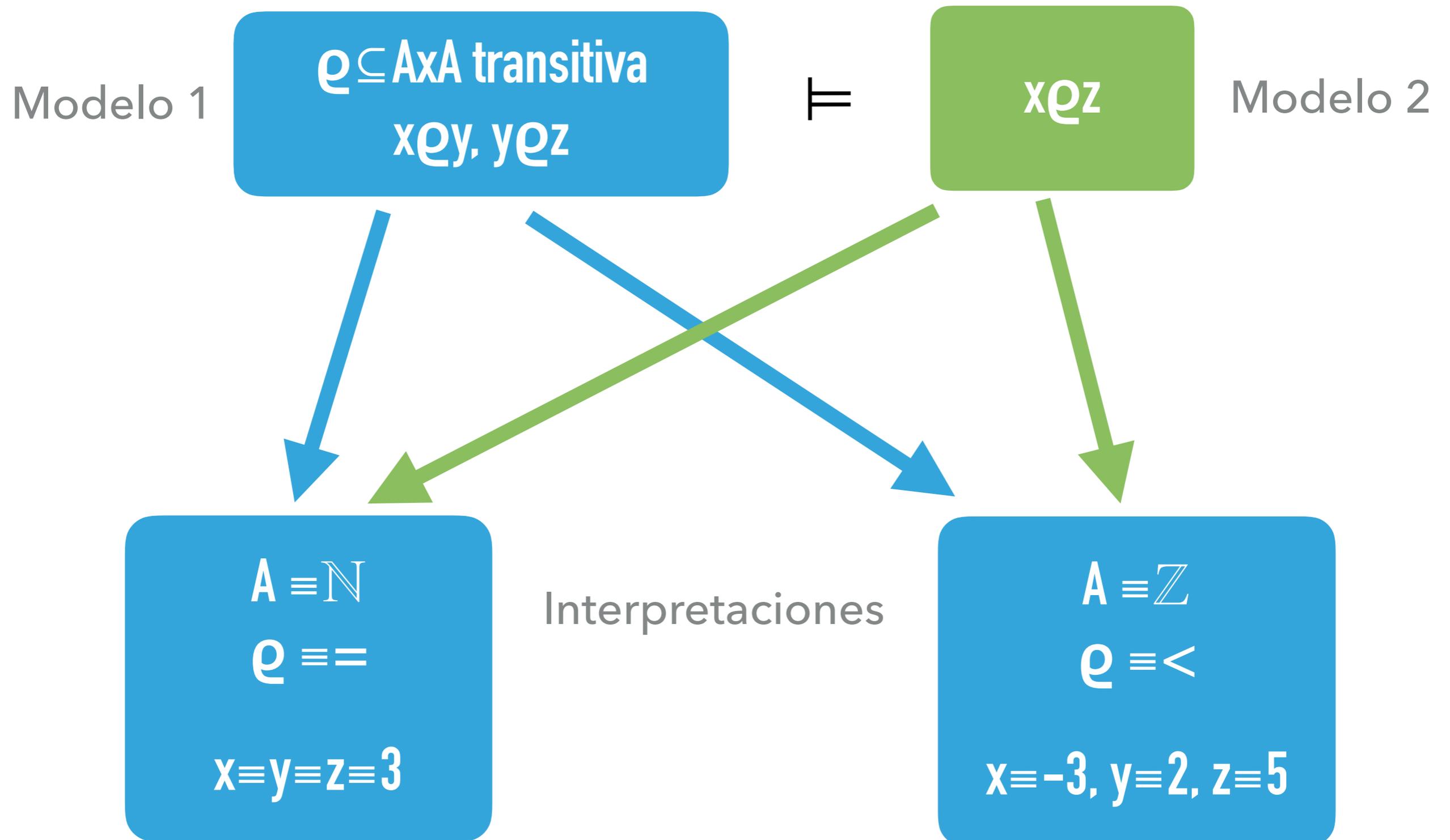
 $X = 5$

2

3



IMPLICACIÓN: EJEMPLO 2 (CONJUNTOS Y RELACIONES)



INTERPRETACIONES

- ▶ Interpretación Simple
- ▶ Interpretación con Conjunto de Datatypes D
- ▶ Interpretación RDF (con D)
- ▶ Interpretación RDFS (con D)

INTERPRETACIÓN SIMPLE

- ▶ cada IRI es asociada a un elemento en el conjunto de los recursos IR
- ▶ cada propiedad es asociada además a un subconjunto de $IR \times IR$
- ▶ los nodos en blanco no son asociados a elementos de IR (interpretación existencial)

INTERPRETACIÓN SIMPLE: EFECTOS

- ▶ todos los grafos RDF son satisfacibles
- ▶ un grafo RDF A implica un grafo B si y solo si un sub-grafo de A es igual a B o una instancia de B (substitución de nodos en blanco)
- ▶ prácticamente no muy util de por sí solo

INTERPRETACIÓN CON CONJUNTO DE DATATYPES D

- ▶ valen los vínculos de una interpretación simple
- ▶ D es el conjunto de datatypes (xsd:integer, xsd:string, etc.) reconocidos
- ▶ cada datatype tiene un función de interpretación universal
- ▶ según el datatype, distintos literales pueden confluír en la misma interpretación
("25.0"^{^^}xsd:decimal y "25"^{^^}xsd:decimal)

INTERPRETACIÓN CON CONJUNTO DE DATATYPES D: EFECTOS

- ▶ los grafos RDF pueden ser no satisfacibles solo en el caso de literales que no respetan los vínculos del datatype

`ex:a ex:p "25.0"^^xsd:boolean .`

- ▶ un grafo RDF A implica un grafo B en los casos previos o dependiendo de los datatype específicos

`ex:a ex:p "25.0"^^xsd:decimal .`

implica `ex:a ex:p "25"^^xsd:decimal .`

INTERPRETACIÓN RDF (CON D)

- ▶ valen los vínculos de una interpretación con un conjunto D de datatypes
- ▶ por cada tripla $\langle s \ p \ o \rangle$, tiene que ser $\langle p \ \text{rdf:type} \ \text{rdf:Property} \rangle$
- ▶ por cada datatype d en D y cada literal $".."$ ^{d} , el literal es interpretado con tipo d
- ▶ por cada propiedad p en el namespace `rdf:` (`rdf:type`, `rdf:nil`, ...), tiene que ser $\langle p \ \text{rdf:type} \ \text{rdf:Property} \rangle$

INTERPRETACIÓN RDF (CON D): EFECTOS

▶ triplas axiomáticas

```
rdf:type rdf:type rdf:Property .
rdf:subject rdf:type rdf:Property .
rdf:predicate rdf:type rdf:Property .
rdf:object rdf:type rdf:Property .
rdf:first rdf:type rdf:Property .
rdf:rest rdf:type rdf:Property .
rdf:value rdf:type rdf:Property .
rdf:nil rdf:type rdf:List .
rdf:_1 rdf:type rdf:Property .
rdf:_2 rdf:type rdf:Property .
...
```

INTERPRETACIÓN RDF (CON D): EFECTOS

- ▶ reglas deductivas para la implicación

	si	entonces
rdfD1	<code>xxx aaa "sss"^^ddd .</code> <i>con ddd en D</i>	<code>xxx aaa _:nnn .</code> <code>_:nnn rdf:type ddd .</code>
rdfD2	<code>xxx aaa yyy .</code>	<code>aaa rdf:type rdf:Property .</code>

INTERPRETACIÓN RDFS (CON D)

- ▶ todos los nodos son de tipo `rdfs:Resource`
- ▶ todos los literales de texto con idioma son de tipo `rdf:langString`
- ▶ cada tipo datatype `d` en `D` es de tipo `rdfs:Datatype`
- ▶ si $\langle x \text{ rdfs:domain } y \rangle$, $\langle u \ x \ v \rangle \quad \text{---} \rightarrow \quad u$ es de tipo `y`
- ▶ si $\langle x \text{ rdfs:range } y \rangle$, $\langle u \ x \ v \rangle \quad \text{---} \rightarrow \quad v$ es de tipo `y`
- ▶ `rdfs:subPropertyOf` y `rdfs:subClassOf` son transitivas, reflexivas y se respeta el significado como "subconjunto de"
- ▶ cada clase es `rdfs:subClassOf` de `rdfs:Resource`
- ▶ cada `rdfs:ContainerMembershipProperty` es `rdf:subPropertyOf` de `rdfs:member`
- ▶ cada `rdfs:Datatype` es `rdf:subClassOf` de `rdfs:Literal`

INTERPRETACIÓN RDFS (CON D): EFECTOS

▶ triplas axiomáticas 1/3

```
rdf:type rdfs:domain rdfs:Resource .
rdfs:domain rdfs:domain rdf:Property .
rdfs:range rdfs:domain rdf:Property .
rdfs:subPropertyOf rdfs:domain rdf:Property .
rdfs:subClassOf rdfs:domain rdfs:Class .
rdf:subject rdfs:domain rdf:Statement .
rdf:predicate rdfs:domain rdf:Statement .
rdf:object rdfs:domain rdf:Statement .
rdfs:member rdfs:domain rdfs:Resource .
rdf:first rdfs:domain rdf:List .
rdf:rest rdfs:domain rdf:List .
rdfs:seeAlso rdfs:domain rdfs:Resource .
rdfs:isDefinedBy rdfs:domain rdfs:Resource .
rdfs:comment rdfs:domain rdfs:Resource .
rdfs:label rdfs:domain rdfs:Resource .
rdf:value rdfs:domain rdfs:Resource .
```

INTERPRETACIÓN RDFS (CON D): EFECTOS

▶ triplas axiomáticas 2/3

```
rdf:type rdfs:range rdfs:Class .
rdfs:domain rdfs:range rdfs:Class .
rdfs:range rdfs:range rdfs:Class .
rdfs:subPropertyOf rdfs:range rdf:Property .
rdfs:subClassOf rdfs:range rdfs:Class .
rdf:subject rdfs:range rdfs:Resource .
rdf:predicate rdfs:range rdfs:Resource .
rdf:object rdfs:range rdfs:Resource .
rdfs:member rdfs:range rdfs:Resource .
rdf:first rdfs:range rdfs:Resource .
rdf:rest rdfs:range rdf:List .
rdfs:seeAlso rdfs:range rdfs:Resource .
rdfs:isDefinedBy rdfs:range rdfs:Resource .
rdfs:comment rdfs:range rdfs:Literal .
rdfs:label rdfs:range rdfs:Literal .
rdf:value rdfs:range rdfs:Resource .
```

INTERPRETACIÓN RDFS (CON D): EFECTOS

▶ triplas axiomáticas 3/3

```
rdf:Alt rdfs:subClassOf rdfs:Container .
rdf:Bag rdfs:subClassOf rdfs:Container .
rdf:Seq rdfs:subClassOf rdfs:Container .
rdfs:ContainerMembershipProperty rdfs:subClassOf rdf:Property .

rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso .

rdfs:Datatype rdfs:subClassOf rdfs:Class .

rdf:_1 rdf:type rdfs:ContainerMembershipProperty .
rdf:_1 rdfs:domain rdfs:Resource .
rdf:_1 rdfs:range rdfs:Resource .
rdf:_2 rdf:type rdfs:ContainerMembershipProperty .
rdf:_2 rdfs:domain rdfs:Resource .
rdf:_2 rdfs:range rdfs:Resource .
...
```

INTERPRETACIÓN RDFS (CON D): EFECTOS

- ▶ reglas deductivas para la implicación 1/4

	si	entonces
rdfs1	<i>cada</i> IRI <i>aaa</i> en <i>D</i>	<code>aaa rdf:type rdfs:Datatype .</code>
rdfs2	<code>aaa rdfs:domain xxx .</code> <code>yyy aaa zzz .</code>	<code>yyy rdf:type xxx .</code>
rdfs3	<code>aaa rdfs:range xxx .</code> <code>yyy aaa zzz .</code>	<code>zzz rdf:type xxx .</code>
rdfs4a	<code>xxx aaa yyy .</code>	<code>xxx rdf:type rdfs:Resource .</code>
rdfs4b	<code>xxx aaa yyy.</code>	<code>yyy rdf:type rdfs:Resource .</code>

INTERPRETACIÓN RDFS (CON D): EFECTOS

- ▶ reglas deductivas para la implicación 2/4

	si	entonces
rdfs 5	<code>xxx rdfs:subPropertyOf yyy</code> <code>yyy rdfs:subPropertyOf zzz</code>	<code>xxx rdfs:subPropertyOf zzz</code>
rdfs 6	<code>xxx rdf:type rdf:Property</code>	<code>xxx rdfs:subPropertyOf xxx</code>
rdfs 7	<code>aaa rdfs:subPropertyOf</code> <code>bbb</code> <code>xxx aaa yyy .</code>	<code>xxx bbb yyy</code>

INTERPRETACIÓN RDFS (CON D): EFECTOS

- ▶ reglas deductivas para la implicación 3/4

	si	entonces
rdfs8	<code>xxx rdf:type rdfs:Class .</code>	<code>xxx rdfs:subClassOf rdfs:Resource .</code>
rdfs9	<code>xxx rdfs:subClassOf yyy .</code> <code>zzz rdf:type xxx .</code>	<code>zzz rdf:type yyy .</code>
rdfs10	<code>xxx rdf:type rdfs:Class .</code>	<code>xxx rdfs:subClassOf xxx .</code>
rdfs11	<code>xxx rdfs:subClassOf yyy .</code> <code>yyy rdfs:subClassOf zzz .</code>	<code>xxx rdfs:subClassOf zzz .</code>

INTERPRETACIÓN RDFS (CON D): EFECTOS

- ▶ reglas deductivas para la implicación 4/4

	si	entonces
rdfs 12	<code>xxx rdf:type rdfs:ContainerMembership Property</code>	<code>xxx rdfs:subPropertyOf rdfs:member</code>
rdfs 13	<code>xxx rdf:type rdfs:Datatype</code>	<code>xxx rdfs:subClassOf rdfs:Literal</code>

RAZONADOR (REASONER)

- De un dataset genera nuevas triplas utilizando un conjunto de reglas (RDF, RDFS, OWL, datalog, ...)
- Las nuevas triplas pueden ser:
 - ▶ agregadas cada vez que el dataset es modificado (**forward chaining**); o
 - ▶ generadas para responder a una query solo cuando necesario (**backward chaining**)
- Sea Protégé que los principales Graph Store permiten elegir entre varios razonadores incorporados o externos

ACTIVIDAD: INFERENCIA RDFS

- ▶ Utilizar en Protégé un reasoner
- ▶ Crear en un Graph Store (Sesame, OpenLink Virtuoso, Apache Fuseki) un repository con inferencia RDFS
- ▶ Exportar Ontología realizada con Protégé
- ▶ Importarla en el Graph Store
- ▶ Realizar queries para probar las inferencias

EMAIL

mceriani@itba.edu.ar