

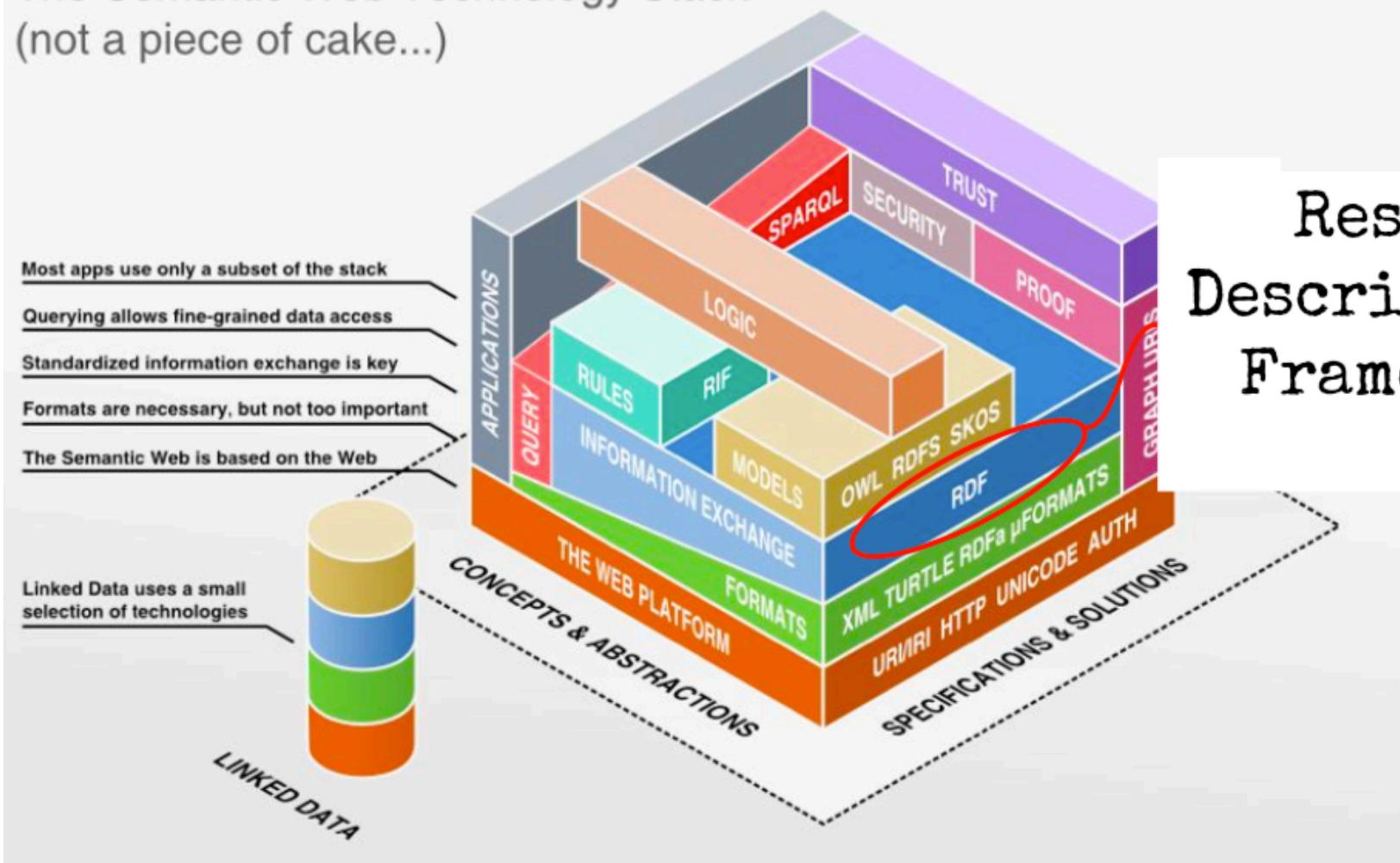
I K N E X

# Knowledge Engineering and Linked Data

**Spring 2023**  
**Dr. Amna Basharat**

# RDF

The Semantic Web Technology Stack  
(not a piece of cake...)





# **Resource Description Framework**

**Basic concepts of RDF**  
**Formats for serializing RDF data**  
**Advanced features of RDF**  
**What is Linked Data?**



# Resource Description Framework

- **Resource**

- can be everything
- must be uniquely identified and referencable via URI

- **Description**

- = description of resources
- ia representing properties and relationships among resources as graphs

- **Framework**

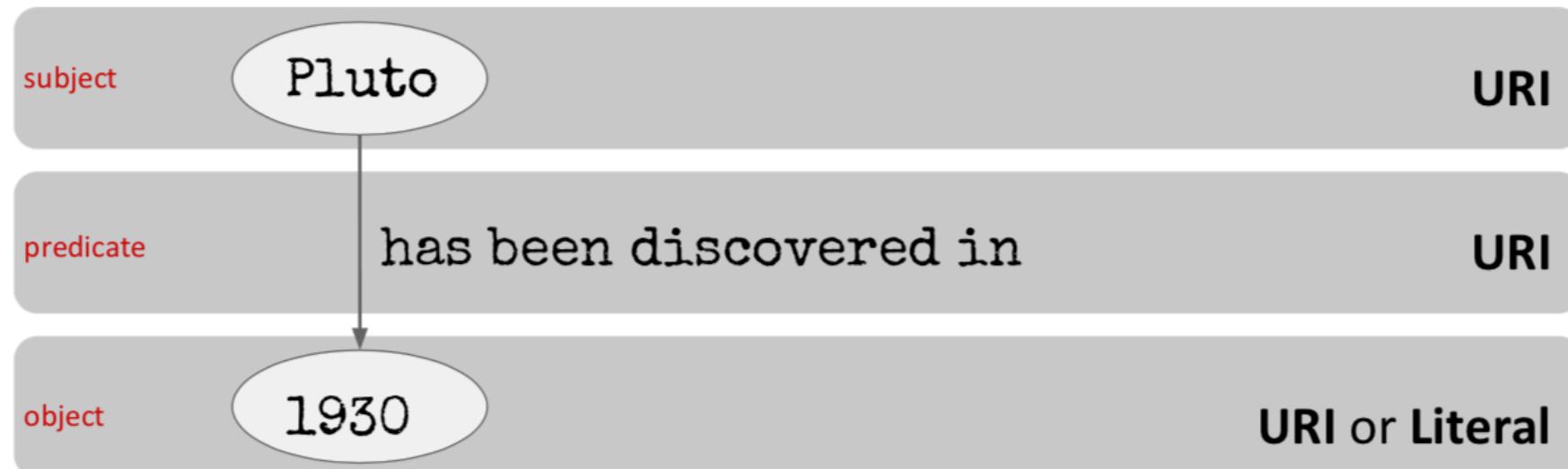
- = combination of web based protocols (URI, HTTP, XML, Turtle, JSON, ...)
- based on formal model (semantics)

- Knowledge in RDF is expressed as a list of **statements**

- All RDF statements follow the same simple schema (**= RDF Triple**)

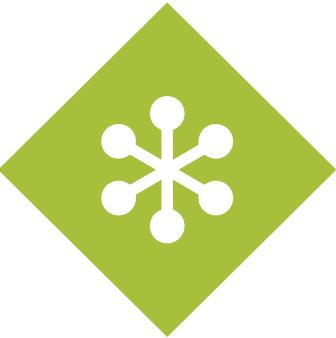


# Resource Description Framework





**URIs**



# Uniform Resource Identifier

- A Uniform Resource Identifier (URI) defines a simple and extensible schema for worldwide unique identification of abstract or physical resources (RFC 3986).
- A **Resource** can be every object with a clear identity (according to the context of the application)
  - as e.g., web pages, books, locations, persons, relations among objects, abstract concepts, etc.
- URI concept is already established in various domains, as e.g.,
  - the Web (URL),
  - Books and publications (ISBN, ISSN),
  - Digital Object Identifier (DOI)



# What if a URI Does Not Exist (Yet)?

- Define a URI by yourself:
  - avoid overlaps → use your own website
  - enable documentation at the same place → Content Negotiation
- Use separate URIs for the resource (Designatum) and its documentation (Designator) via
  - Content Negotiation and/or
  - URI references (e.g. via "#" fragment identifier)



# Resource Description Framework

- **RDF Statements (RDF-Triple):**

Subject	+	Property	+	Object / Value	
URI		URI		URI / Literal	RDF Building Blocks

N-Triples Serialization

```
<http://dbpedia.org/resource/Pluto> <http://dbpedia.org/ontology/discovered> "1930" .
```

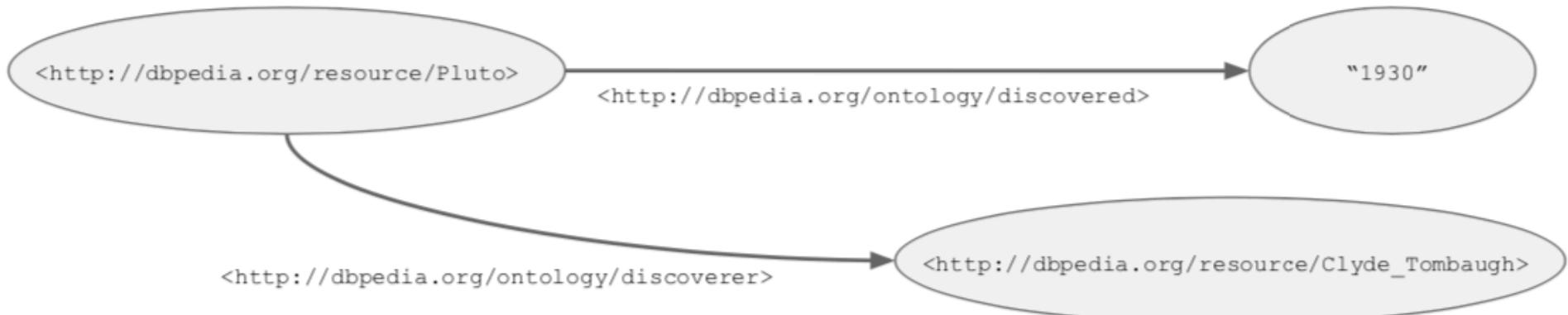




# Resource Description Framework

- **URIs and Literals**

- URIs reference resources uniquely
- Literals describe data values that don't have a separate existence.





# Literals and Data Types

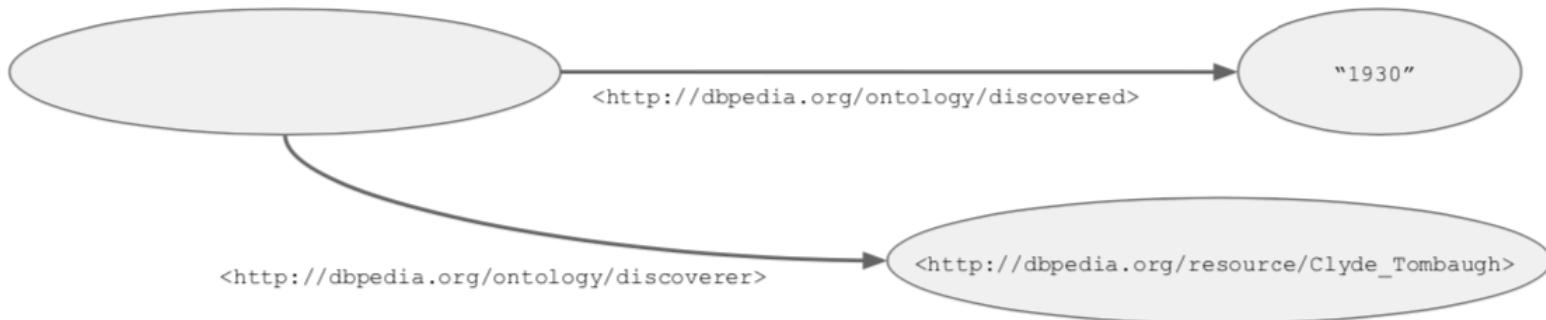
- typed literals can be expressed via **XML Schema datatypes**
- Namespace for typed literals:  
`http://www.w3.org/2001/XMLSchema#`
- Examples:  
`"Semantics"^^<http://www.w3.org/2001/XMLSchema#string>`  
`"1161.00"^^<http://www.w3.org/2001/XMLSchema#float>`  
`"2015-08-02"^^<http://www.w3.org/2001/XMLSchema#date>`
- **Language Tags** denote the (natural) language of the text:
  - Example:  
`"Semantik"@de , "Semantics"@en`

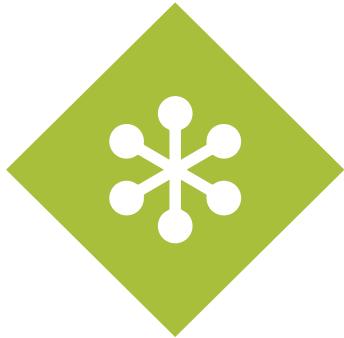


# Blank Nodes

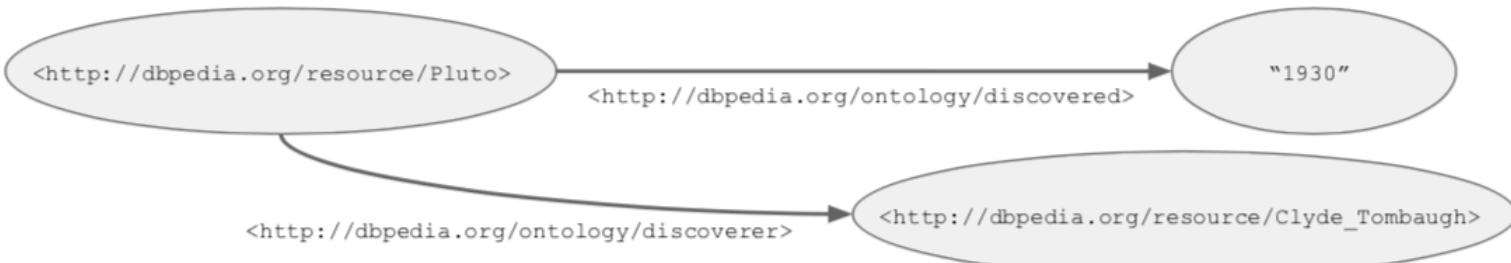
- **Blank Nodes**

- denote existence of an individual with specific attributes, but without
- providing an identification or reference





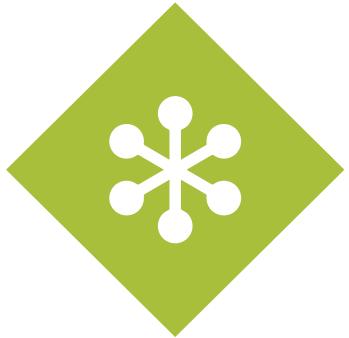
# RDF Serialization



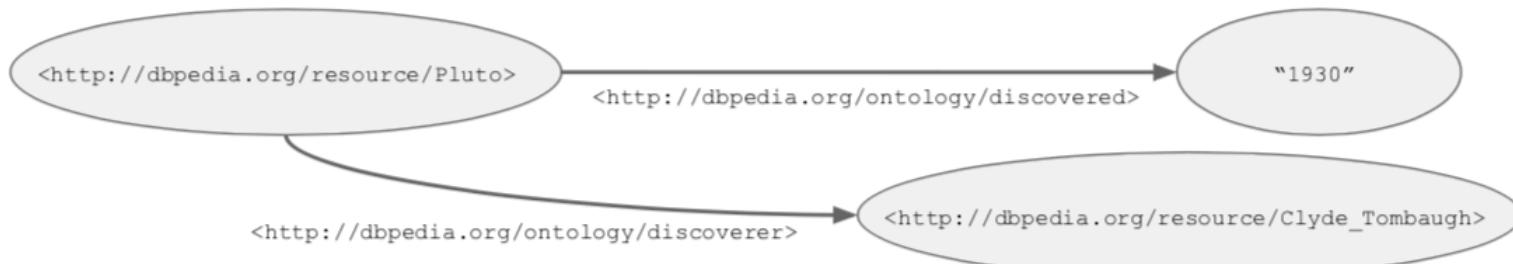
## ● N-Triples Notation

- URIs/IRIs in angle brackets
- Literals in quotation marks
- Triple ends with a period

```
<http://dbpedia.org/resource/Pluto> <http://dbpedia.org/ontology/discovered> "1930" .  
<http://dbpedia.org/resource/Pluto> <http://dbpedia.org/ontology/discoverer>  
<http://dbpedia.org/resource/Clyde_Tombaugh> .
```



# RDF Serialization



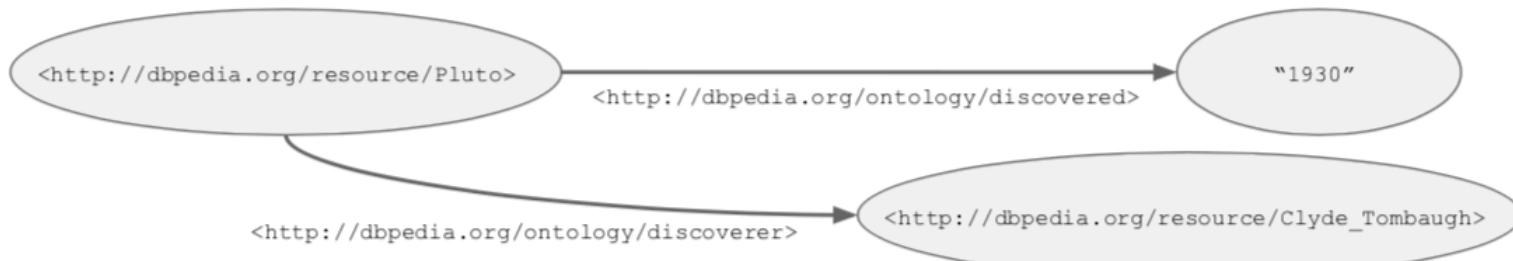
- **RDF/XML Notation**

S      P      O

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:ns0="http://dbpedia.org/ontology/">
  <rdf:Description rdf:about="http://dbpedia.org/resource/Pluto">
    <ns0:discovered>1930</ns0:discovered>
    <ns0:discoverer rdf:resource="http://dbpedia.org/resource/Clyde_Tombaugh"/>
  </rdf:Description>
</rdf:RDF>
```



# RDF Serialization



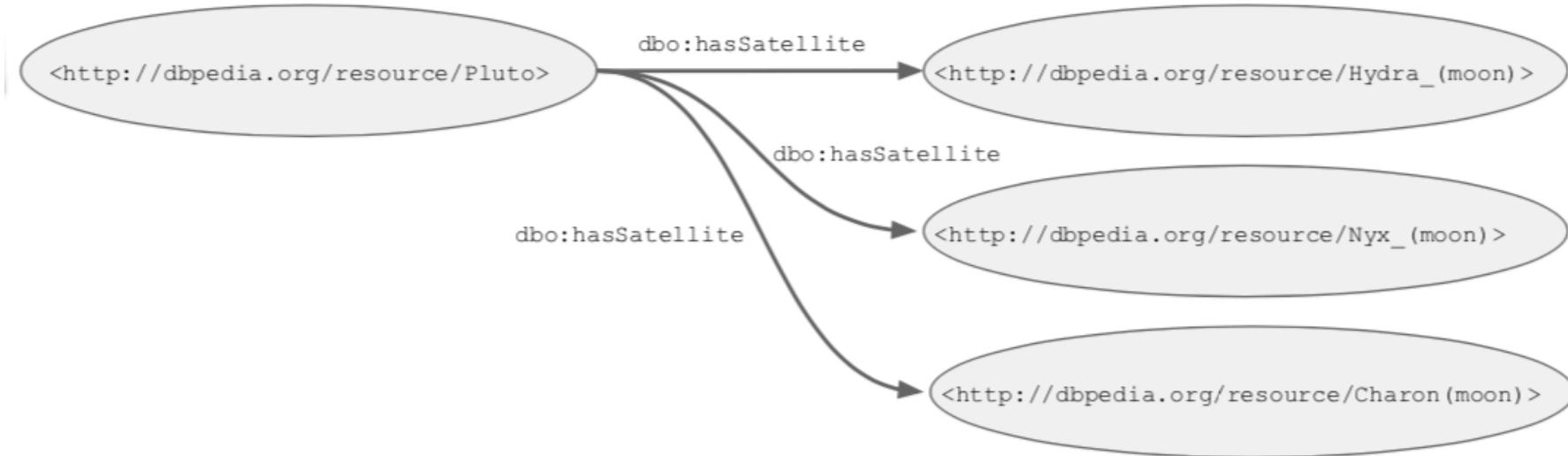
- **Turtle (Terse RDF Triple Language) Notation**

- extension of N\_Triples

```
@prefix dbo: <http://dbpedia.org/ontology/> .  
@base  <http://dbpedia.org/resource/> .  
  
<Pluto> dbo:discovered "1930" .  
  
<Pluto> dbo:discoverer <Clyde_Tombaugh> .
```



# RDF Serialization



- Abbreviating in Turtle

```
@prefix dbo: <http://dbpedia.org/ontology/> .  
@base <http://dbpedia.org/resource/> .  
  
<Pluto> dbo:hasSatellite <Hydra_(moon)> ,  
    <Nyx_(moon)> ,  
    <Charon_(moon)> .
```

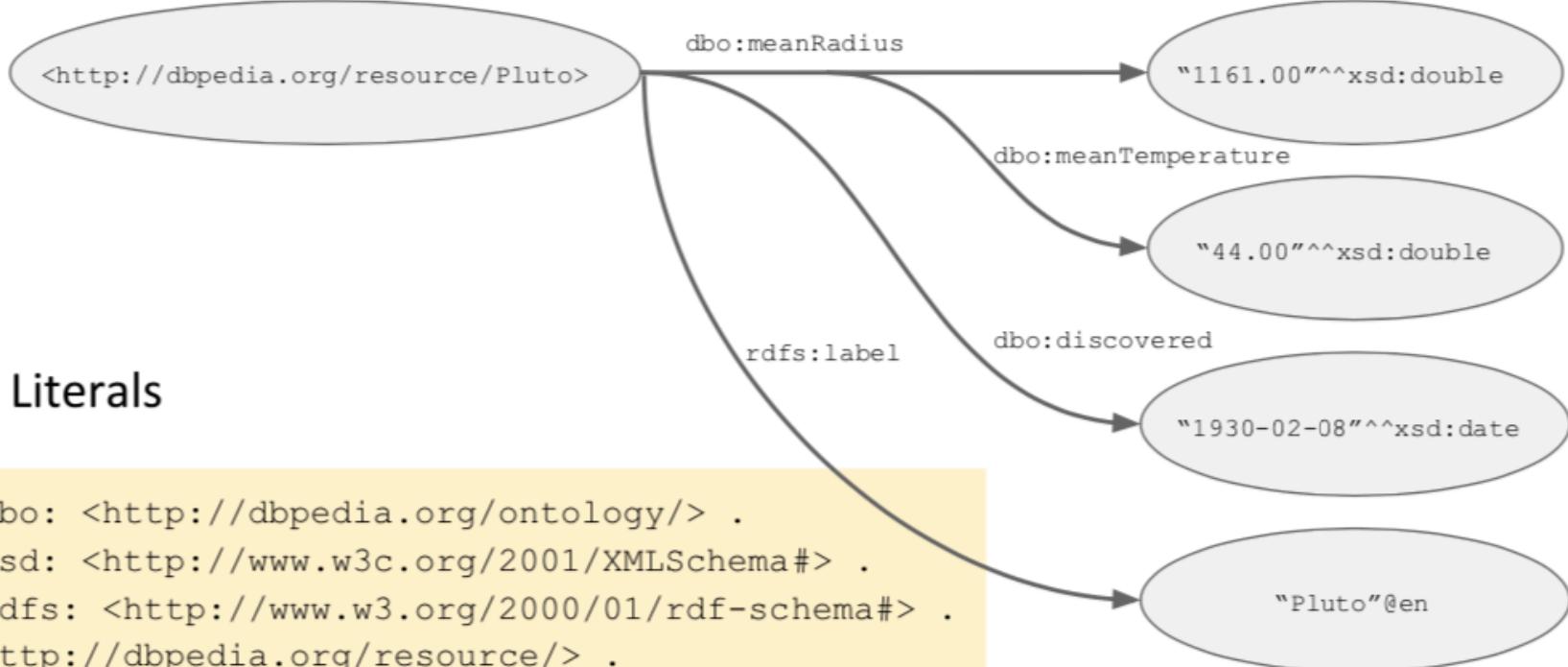
comma indicates that subsequent triples have same subject and property (**object list**)



# Turtle - Terse RDF

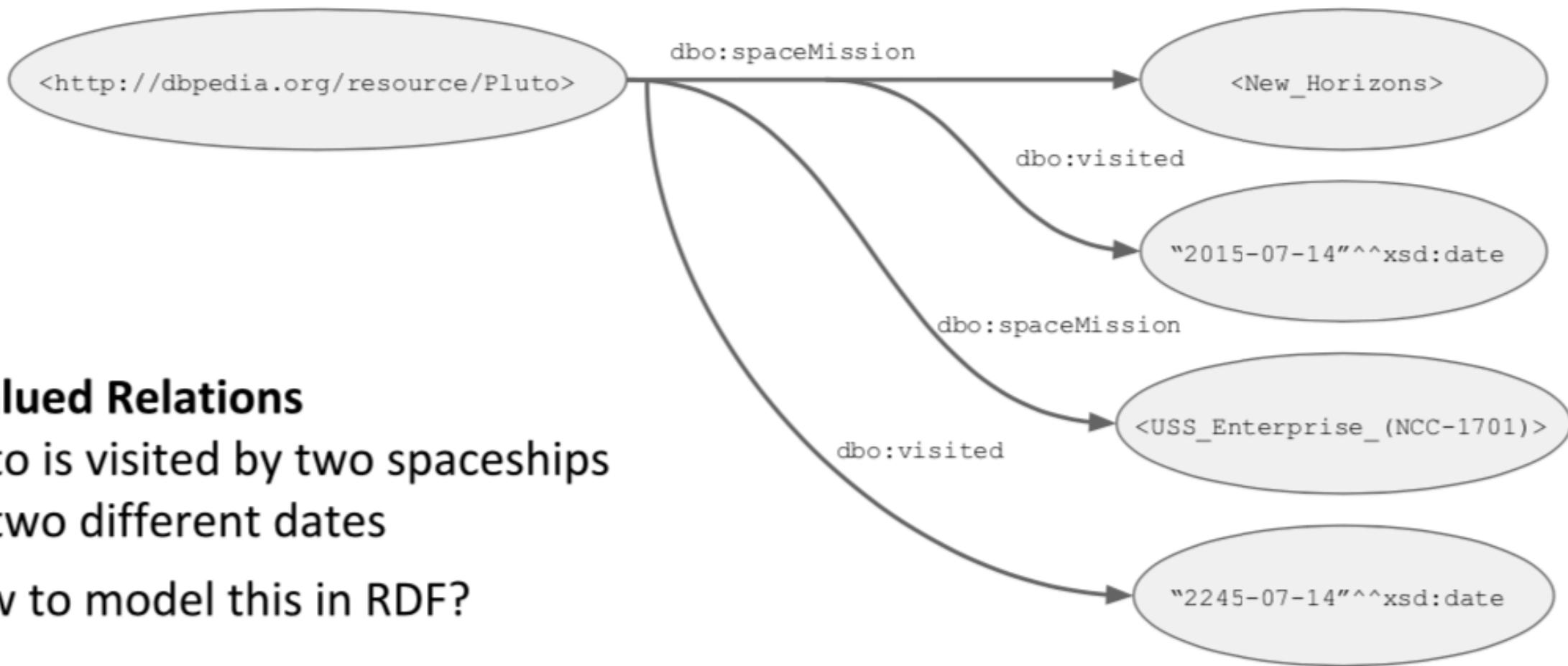
- Typed Literals

```
@prefix dbo: <http://dbpedia.org/ontology/> .  
@prefix xsd: <http://www.w3c.org/2001/XMLSchema#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
@base <http://dbpedia.org/resource/> .  
  
<Pluto> dbo:meanRadius "1161.00"^^xsd:double ;  
    dbo:meanTemperature "44.00"^^xsd:double ;  
    dbo:discovered "1930-02-18"^^xsd:date ;  
    rdfs:label "Pluto"@en .
```





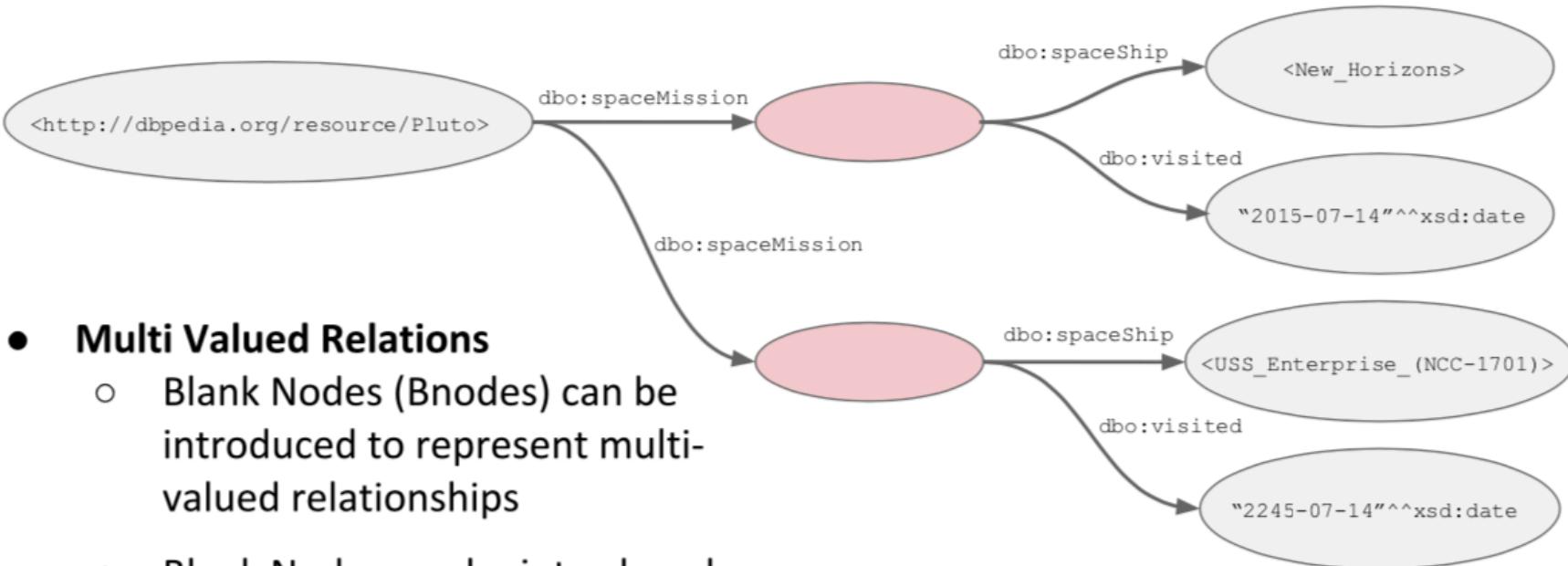
# Multi-Valued Relations



- **Multi Valued Relations**
  - Pluto is visited by two spaceships on two different dates
  - How to model this in RDF?
  - **Problem:** unique association



# Blank Nodes

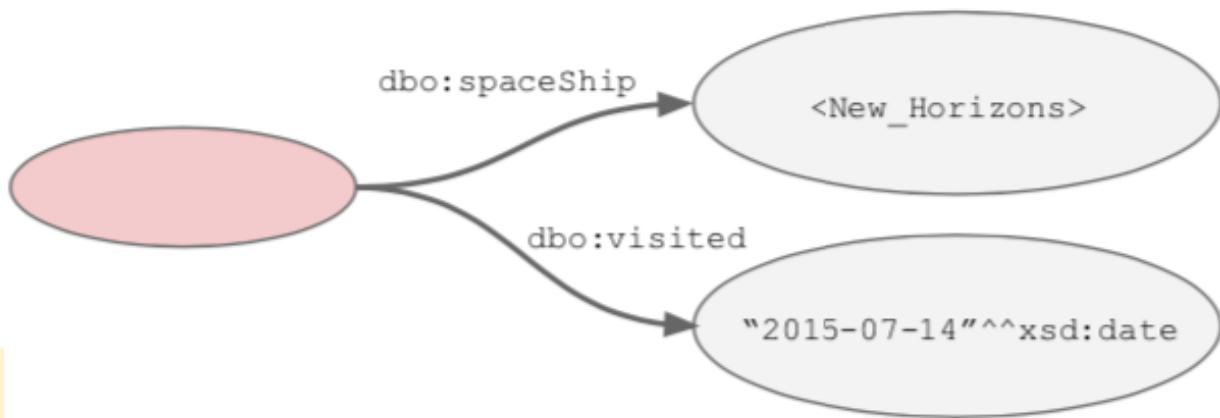


- **Multi Valued Relations**

- Blank Nodes (Bnodes) can be introduced to represent multi-valued relationships
- Blank Nodes can be introduced for resources that don't need a name (auxiliary nodes)



# Turtle - Terse RDF

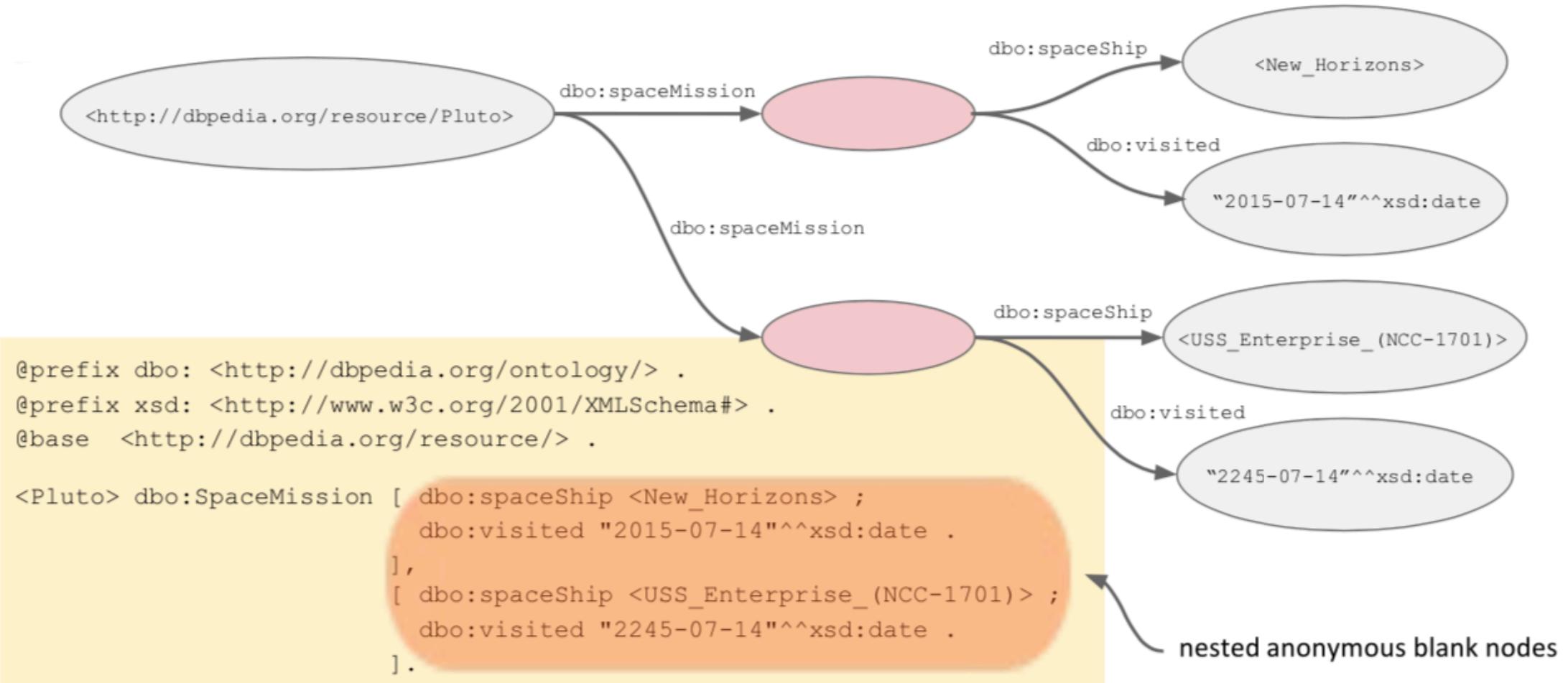


```
@prefix dbo: <http://dbpedia.org/ontology/> .  
@prefix xsd: <http://www.w3c.org/2001/XMLSchema#> .  
@base <http://dbpedia.org/resource/> .  
  
[] dbo:spaceShip <New_Horizons> ;  
    dbo:visited "2015-07-14"^^xsd:date .
```

anonymous blank node as subject

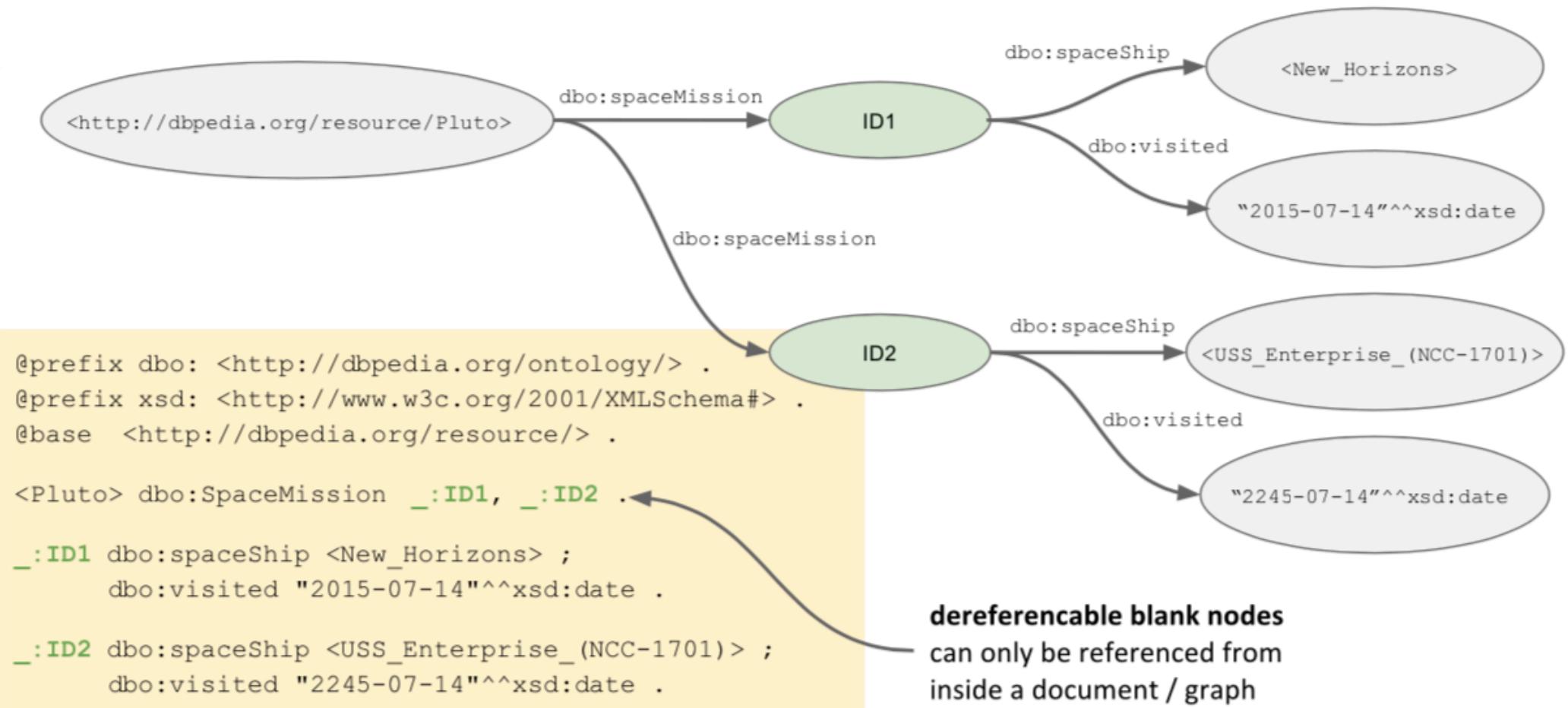


# Turtle - Terse RDF





# Turtle - Terse RDF



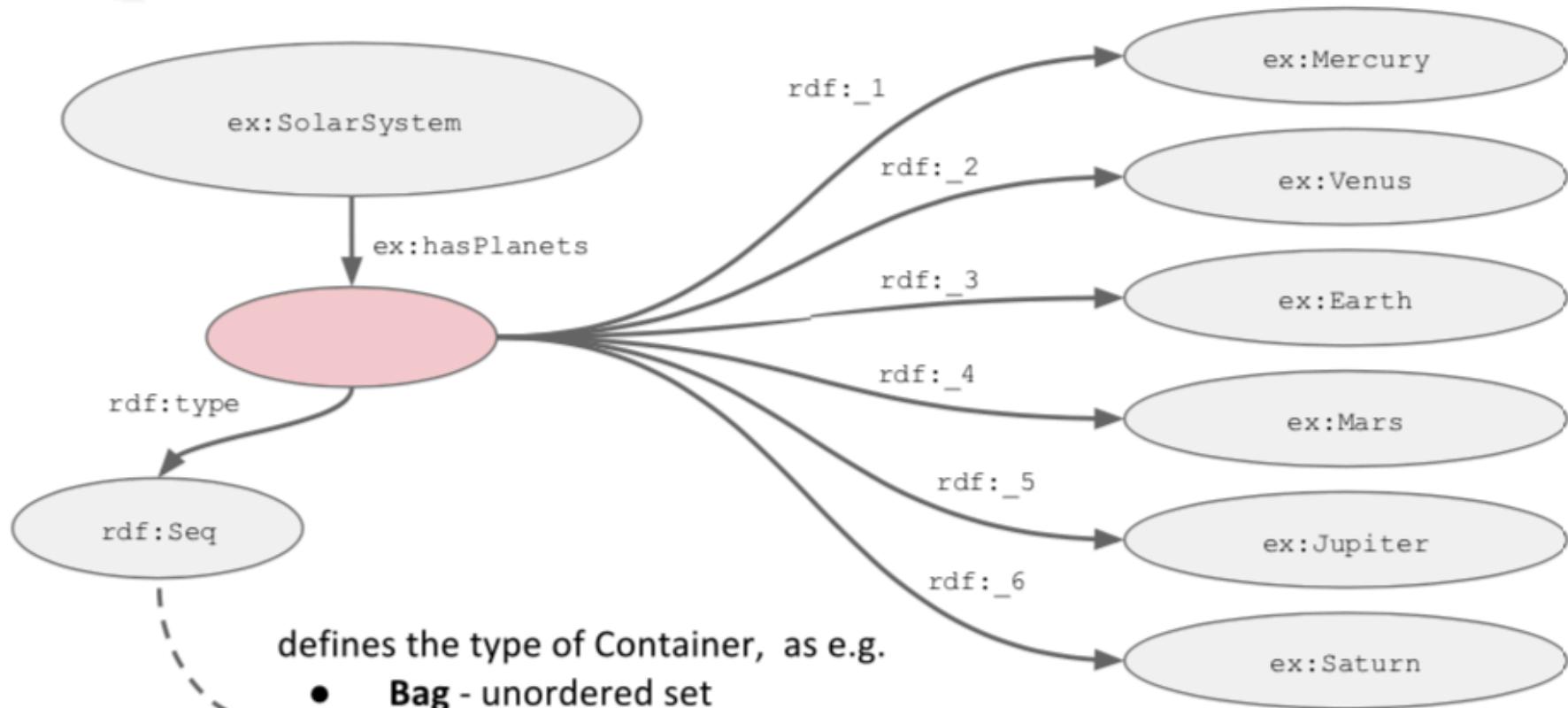


# RDF Lists

- General Data structure to enumerate any resources or literals
- Only shortcuts without additional semantic expressivity
- Distinguish between
  - **Container**: open list, i.e. extension (new entries) possible
  - **Collection**: closed list, i.e. no extension possible



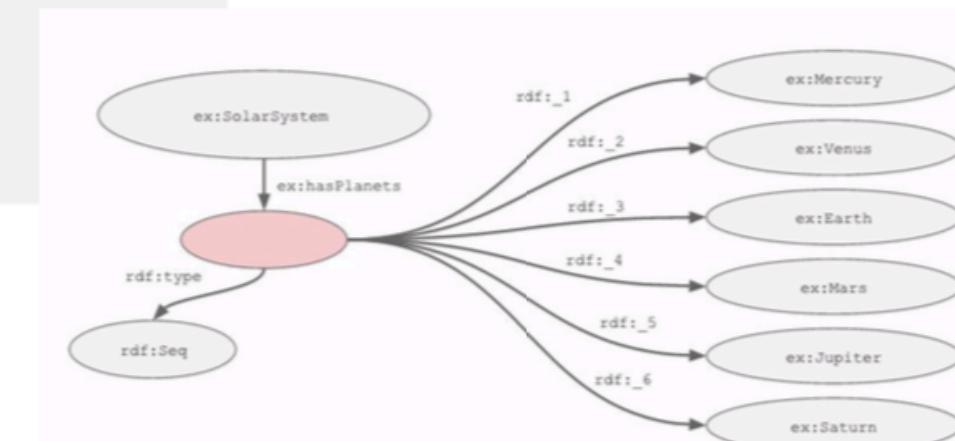
# RDF Container





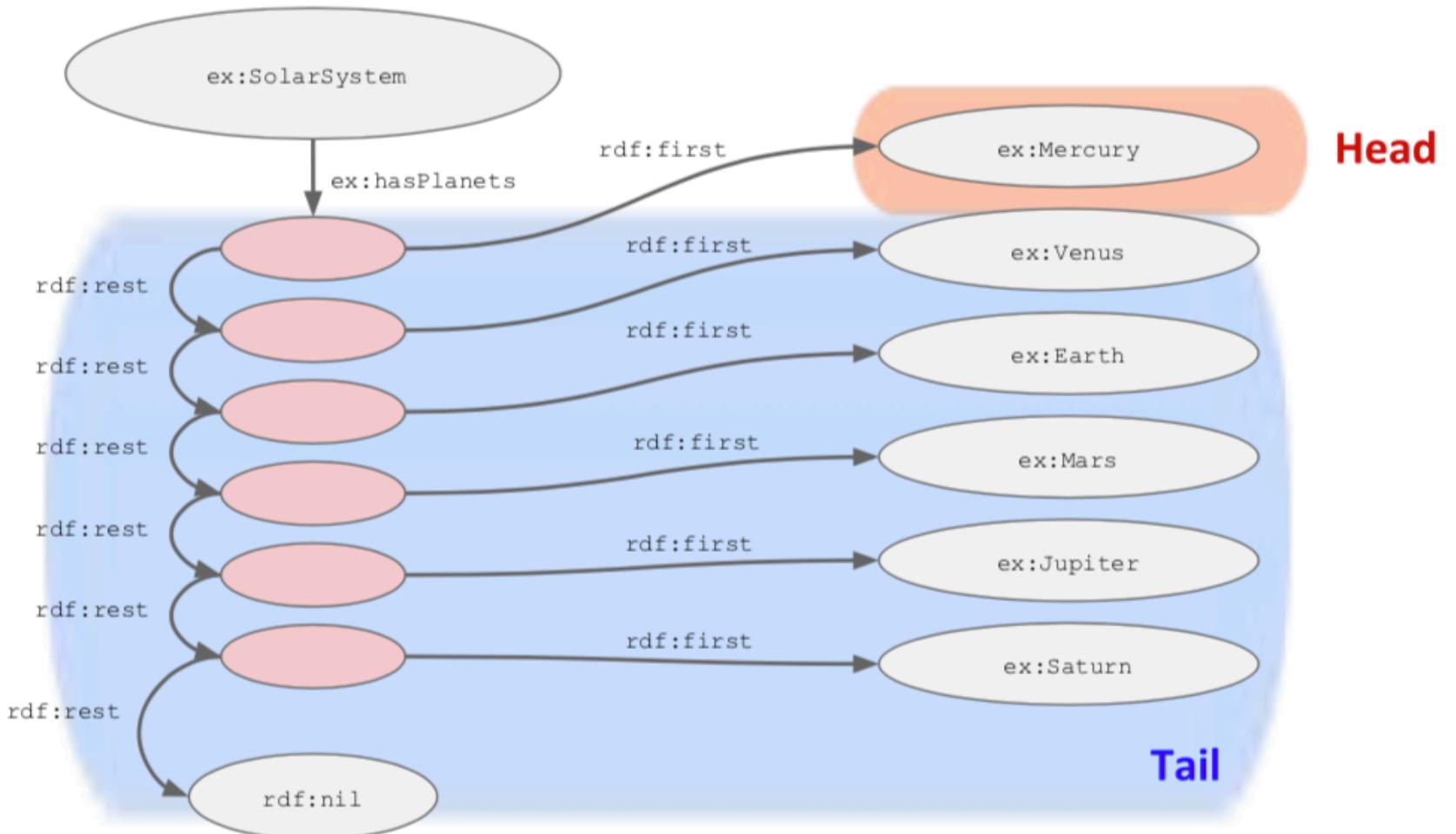
# RDF Container

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix ex: <http://example.org/test#> .  
  
ex:SolarSystem ex:hasPlanets [  
    a rdf:Seq ;  
    rdf:_1 ex:Mercury ;  
    rdf:_2 ex:Venus ;  
    rdf:_3 ex:Earth ;  
    rdf:_4 ex:Mars ;  
    rdf:_5 ex:Jupiter ;  
    rdf:_6 ex:Saturn  
] .
```





# RDF Collection

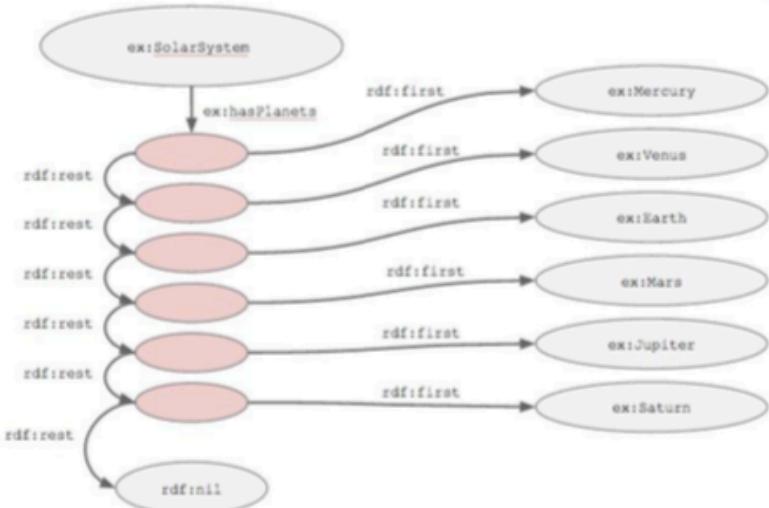




# RDF Collection

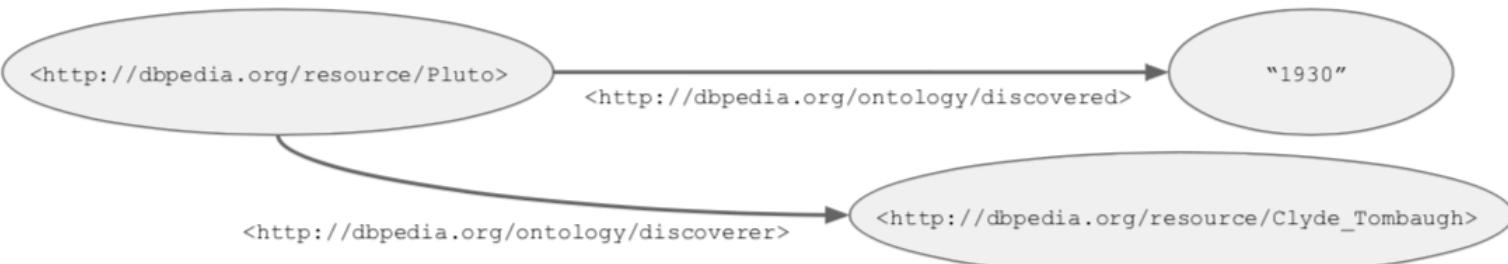
```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix ex: <http://example.org/test#> .  
  
ex:SolarSystem ex:hasPlanets [  
    rdf:first ex:Mercury ; rdf:rest [  
        rdf:first ex:Venus ; rdf:rest [  
            rdf:first ex:Earth ; rdf:rest [  
                rdf:first ex:Mars ; rdf:rest [  
                    rdf:first ex:Jupiter ; rdf:rest [  
                        rdf:first ex:Saturn ;  
                        rdf:rest rdf:nil  
                    ]  
                ]  
            ]  
        ]  
    ]  
] .
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix ex: <http://example.org/test#> .  
  
ex:SolarSystem ex:hasPlanets (  
    ex:Mercury ex:Venus ex:Earth ex:Mars ex:Jupiter ex:Saturn  
) .
```





# RDF Serialization



## • JSON-LD Notation (RDF 1.1)



# **How Would You Say?**

**“My dear Watson, ...I suppose that  
it was the gardener who has killed  
the butler!”**



# RDF Reification

Making Statements about  
Statements



# RDF Reification

- RDF also permits interleaving of statements, i.e. to make statements about statements
- Example:
  - Sherlock Holmes supposes that the gardener has killed the butler
- ■ **Part 1: the gardener has killed the butler**  
ex:Gardener ex:hasKilled ex:butler .
- ■ **Part 2: Sherlock Holmes supposes...**  
dbpedia:Sherlock\_Holmes ex:supposes ????

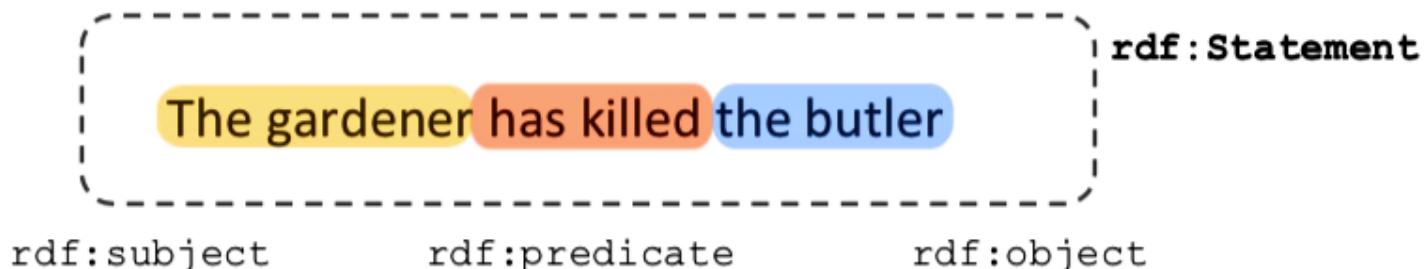


# RDF Reification

- **rdf:Statement**

defines an RDF statement consisting of subject, predicate, object

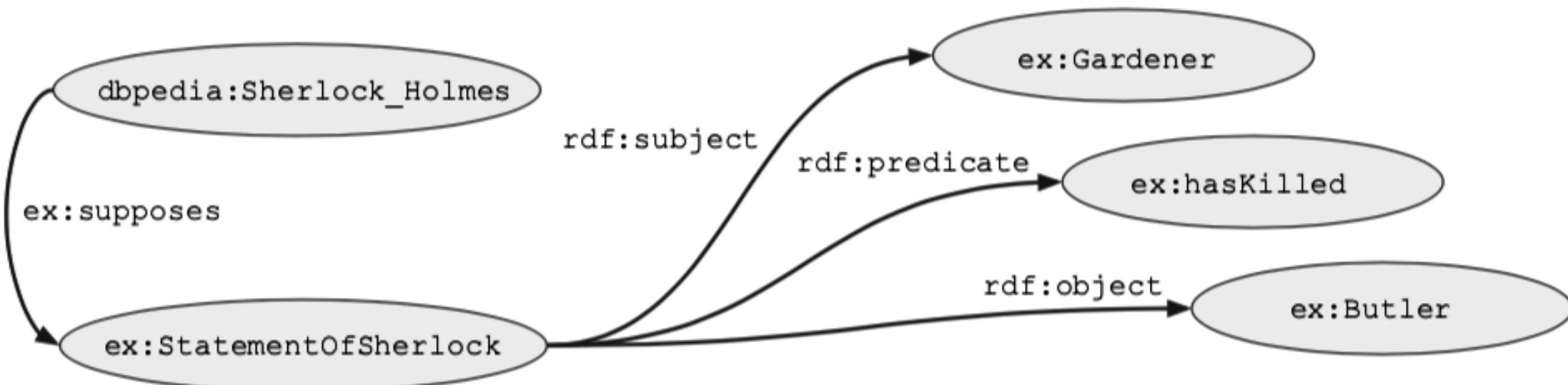
- - **rdf:subject** - the described resource
  - **rdf:predicate** - the original property
  - **rdf:object** - the value of the property





# RDF Reification

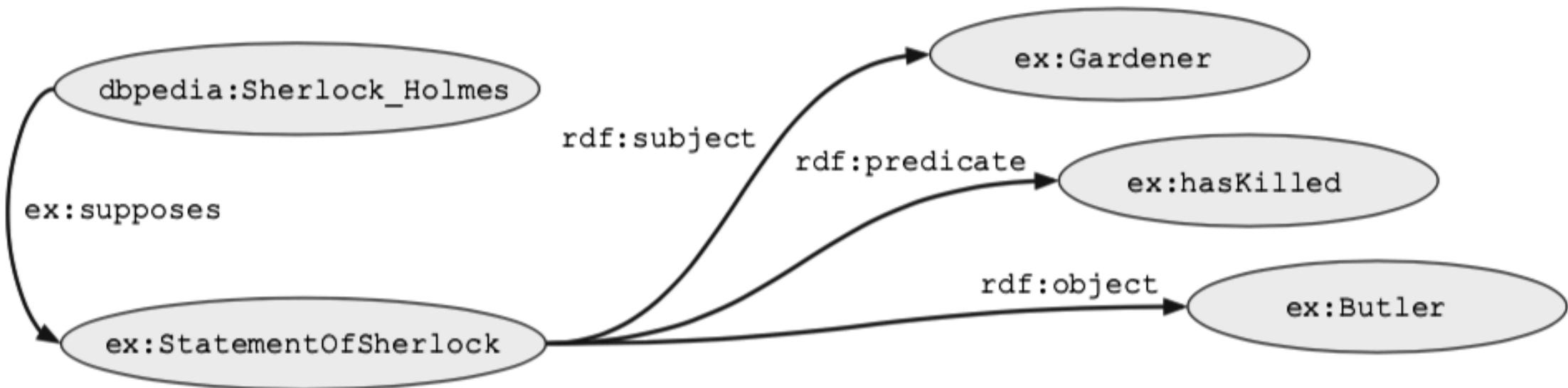
- Sherlock Holmes supposes that the gardener has killed the butler



@prefix	<u>rdf: &lt;<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>&gt;</u> .
@prefix	<u>dbpedia &lt;<a href="http://dbpedia.org/resource/">http://dbpedia.org/resource/</a>&gt;</u> .
@prefix	<u>ex: &lt;<a href="http://example.org/Crimestories#">http://example.org/Crimestories#</a>&gt;</u> .



# RDF Reification



```
@prefix dbpedia: <http://dbpedia.org/resource/> .  
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix ex:      <http://example.org/Crimestories#> .
```

```
dbpedia:SherlockHolmes ex:supposes ex:StatementOfSherlock .  
ex:StatementOfSherlock a rdf:Statement ;  
    rdf:subject ex:Gardener ;  
    rdf:object ex:hasKilled ;  
    rdf:object ex:Butler .
```



# RDF Reification

- What is the use of reification?
  - modeling data provenance
  - formalizing statements about reliability and trust
  - define metadata about statements
- But... you should be careful....
  - with reification relations can be transformed into classes/instances (type conflicts)
  - definition of infinite recursions and cycles



# RDFS

Enhancing the expressivity of RDF

# What does it really mean?

<http://dbpedia.org/resource/Pluto>

<http://dbpedia.org/property/satelliteOf>

[http://dbpedia.org/resource/Charon\\_\(moon\)](http://dbpedia.org/resource/Charon_(moon))

<http://dbpedia.org/ontology/discoverer>

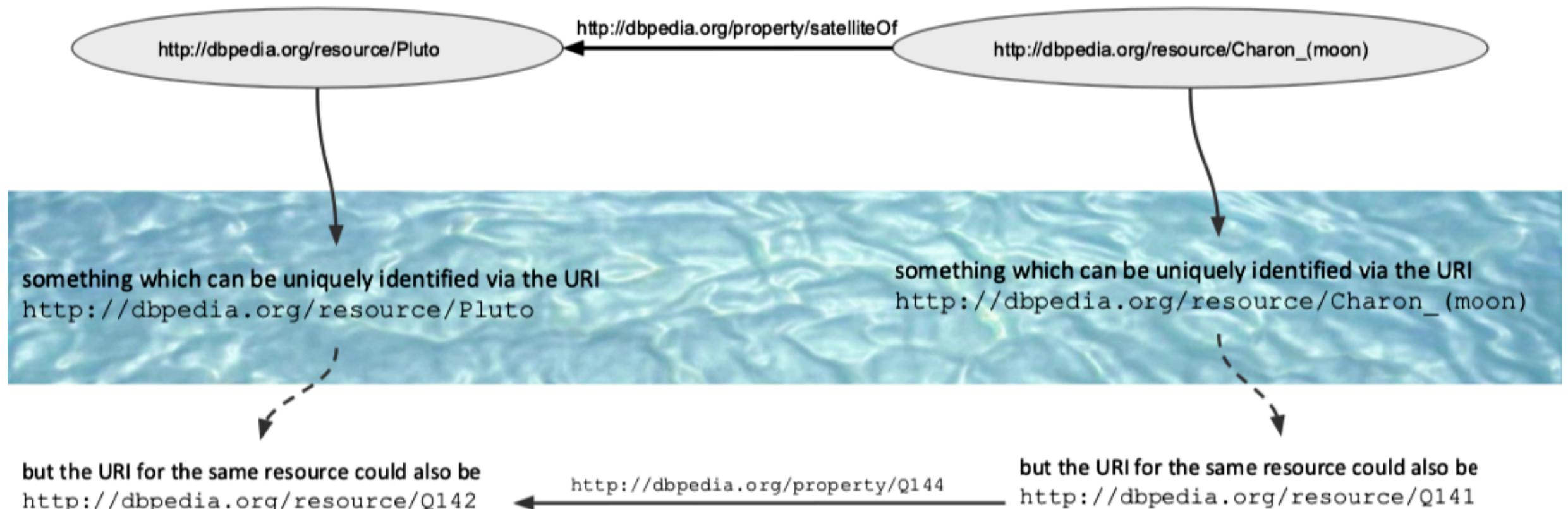
[http://dbpedia.org/resource/Clyde\\_Tombaugh](http://dbpedia.org/resource/Clyde_Tombaugh)

Where does the intended meaning really come from?





# What Does It Really Mean?



# The Semantic Web Technology Stack (not a piece of cake...)

Most apps use only a subset of the stack

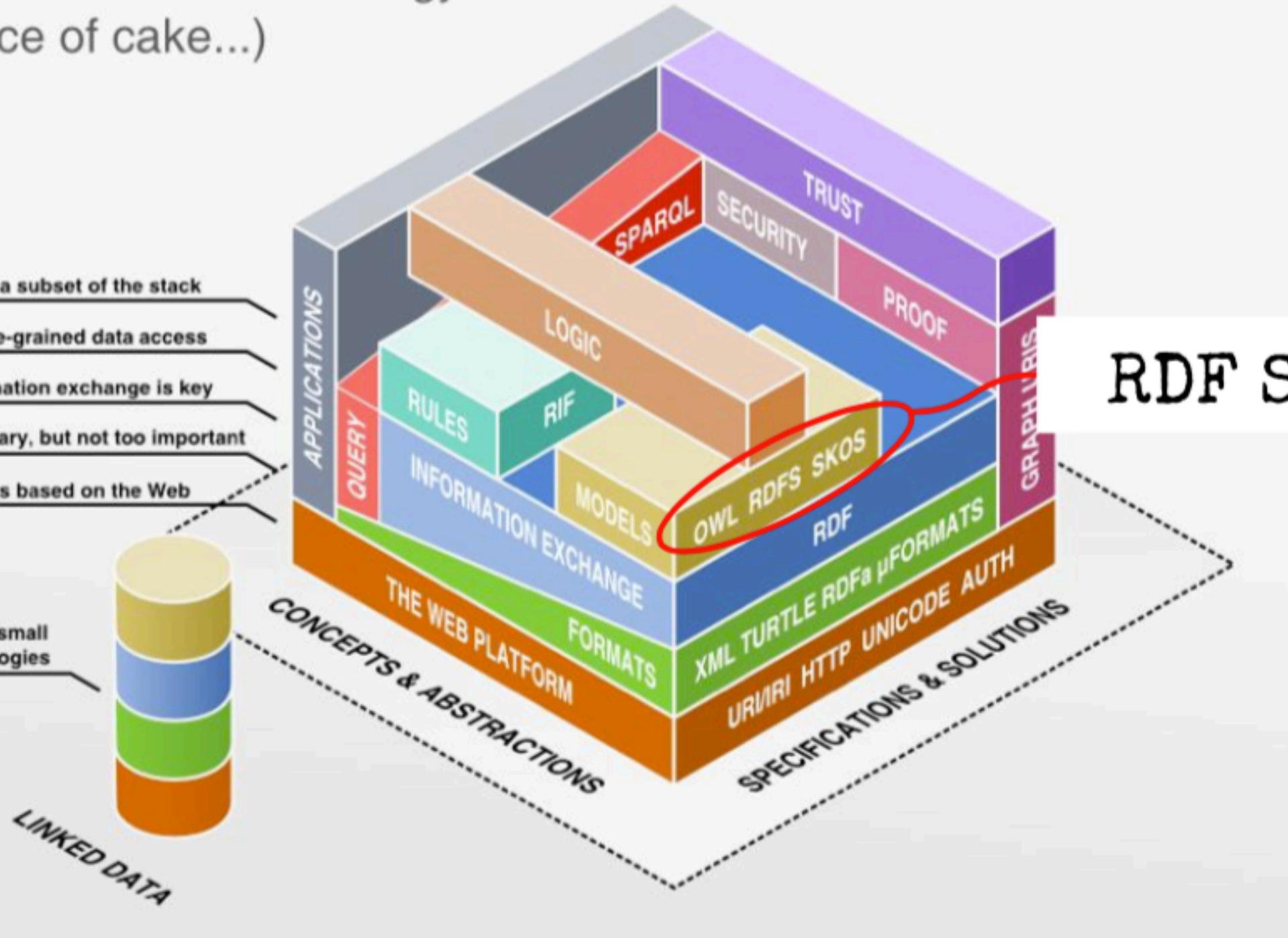
Querying allows fine-grained data access

Standardized information exchange is key

Formats are necessary, but not too important

The Semantic Web is based on the Web

Linked Data uses a small selection of technologies



## RDF Schema



# RDF Schema

- RDF Schema, officially called “RDF Vocabulary Description Language”
- RDF Schema allows:
  - Definition of classes via rdfs:Class
  - Class instantiation in RDF via rdf:type
- Example:
  - :Planet rdf:type rdfs:Class .
  - :Earth rdf:type :Planet .



# RDF Schema

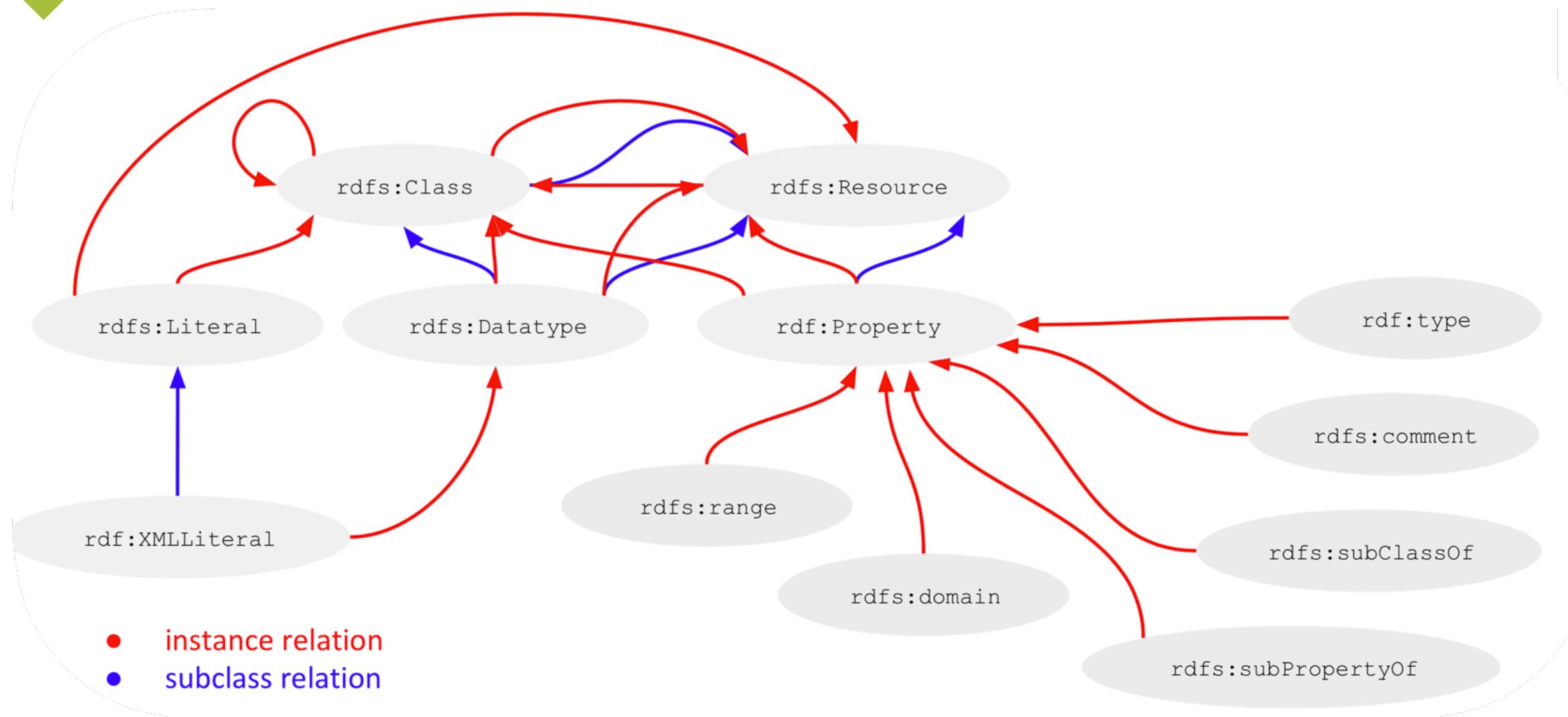
- Definition of properties via rdf:Property
- Definition of property restrictions on domain and range via rdfs:domain and rdfs:range
- Example
  - :CelestialBody rdf:type rdfs:Class .
  - :satelliteOf rdf:type rdf:Property .
  - :satelliteOf rdfs:domain :CelestialBody .
  - :satelliteOf rdfs:range :CelestialBody .

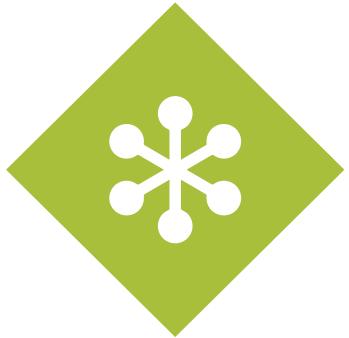


# RDF Schema

- Everything in the RDF model is a resource
  - rdfs:Class rdf:type rdfs:Resource .
  - rdf:Property rdf:type rdfs:resource .
  - rdfs:Literal rdf:type rdfs:Resource .
  - rdfs:XMLLiteral rdf:type rdfs:Resource .
  - rdfs:Datatype rdf:type rdfs:Resource .
  - rdfs:Container rdf:type rdfs:Resource .
  - rdfs:ContainerMembershipProperty rdf:type rdfs:Resource .

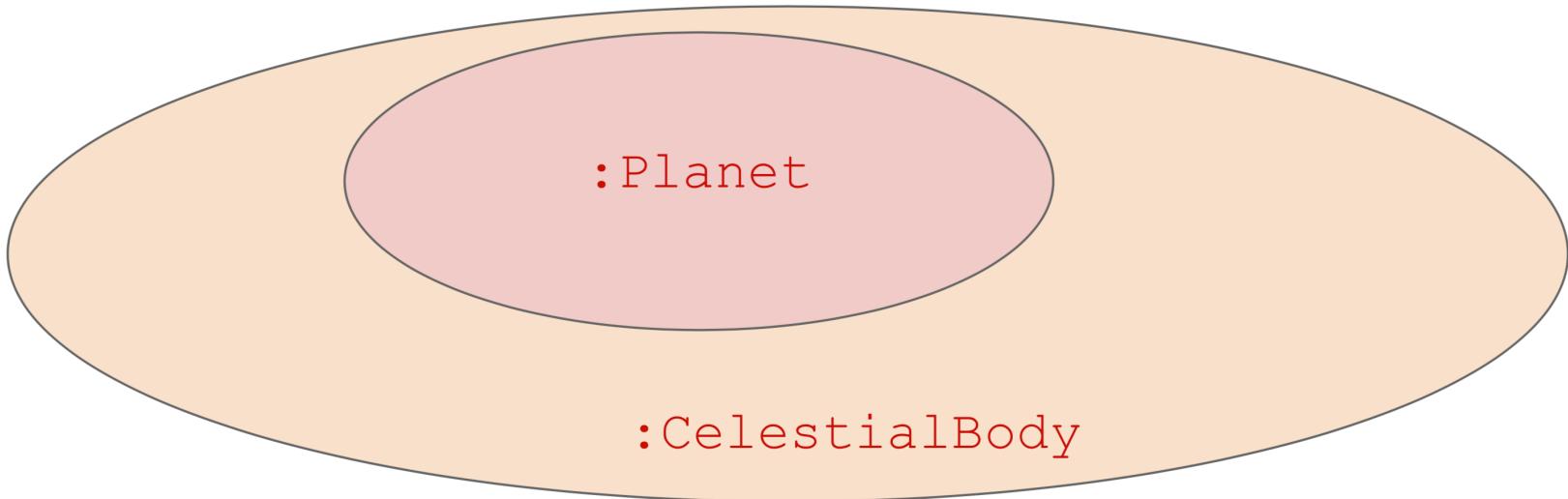
# RDF Schema

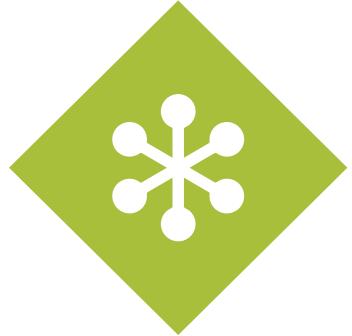




# Definition of Hierarchical Relationships

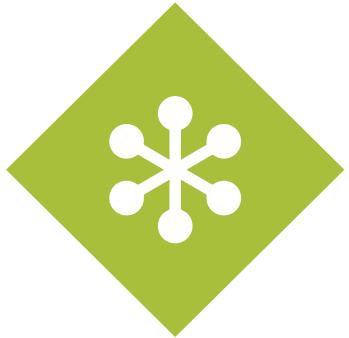
- Subclasses and superclasses via rdfs:subClassOf
  - Example:
    - *:Planet rdfs:subClassOf :CelestialBody .*





# Definition of Hierarchical Relationships

- Definition of hierarchical relationships:
  - Subclasses and superclasses via rdfs:subClassOf
  - Example:
    - *:Planet rdfs:subClassOf :CelestialBody .*
  - Subproperties and superproperties via subPropertyOf
  - Example
    - *:artificialSatelliteOf rdfs:subPropertyOf :satelliteOf .*



# RDF Schema

- Some more properties:
  - **rdfs:seeAlso**  
defines a relation of a resource to another, which explains it
  - **rdfs:isDefinedBy**  
subproperty of rdfs:seeAlso, defines the relation of a resource to its definition
  - **rdfs:comment** comment, usually as text
  - **rdfs:label**  
„readable“ name of a resource (contrary to ID)
  - **rdfs:member**  
super-property of all the container membership properties (e.g. rdf:\_1, ...)



# RDFS Example

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix : <http://example.org/Space#> .
```

```
:Planet rdf:type rdfs:Class ;  
        rdfs:subClassOf :CelestialBody .  
:Satellite rdf:type rdfs:Class ;  
        rdfs:subClassOf :CelestialBody .  
:ArtificialSatellite rdf:type rdfs:Class ;  
        rdfs:subClassOf :Satellite .
```

Class Definitions

```
:satelliteOf rdf:type rdf:Property ;  
        rdfs:domain :CelestialBody .  
        rdfs:range :CelestialBody .
```

Property Definitions

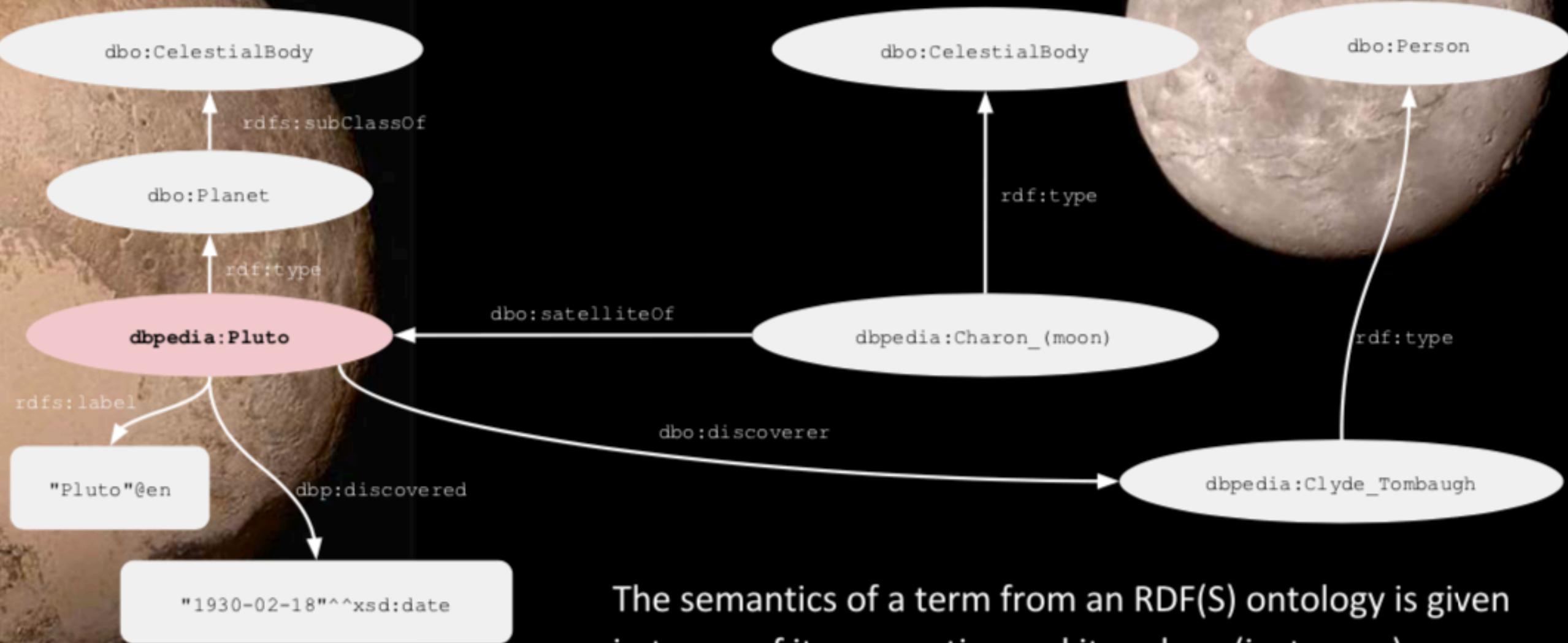
```
:Earth rdf:type :Planet .  
:Moon rdf:type :Satellite ;  
        :satelliteOf :Earth .  
:Sputnik1 rdf:type :ArtificialSatellite ;  
        :satelliteOf :Earth ;  
        rdfs:label "Sputnik 1"@en ;  
        rdfs:comment "the first artificial Earth satellite in 1957" .
```

Instance Definitions



# **Logical Inference With RDFS**

# How much knowledge (semantics) is there?



The semantics of a term from an RDF(S) ontology is given in terms of its properties and its values (instances)

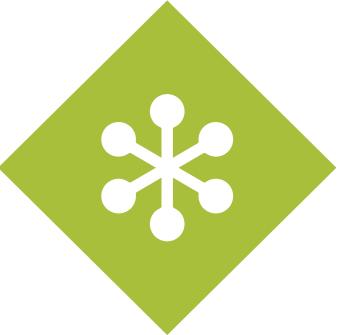


# RDF(S) Semantics

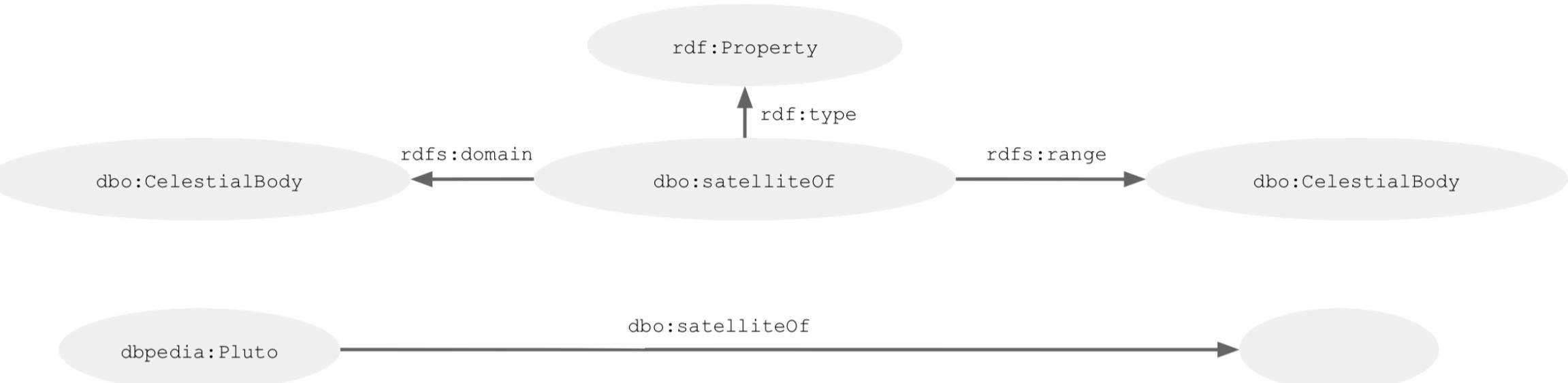
- In difference to other data definition languages, RDF(S) is based on **formal semantics**
- Formal semantics enables RDF(S) to draw **valid** and **sound logical inferences**
- Examples:
  - dbpedia:Pluto `rdf:type` dbo:Planet
  - dbo:Planet `rdfs:subClassOf` dbo:CelestialBody
  - dbo:artificialSatelliteOf `rdfs:subPropertyOf` dbo:satelliteOf

dbpedia:Pluto  $\in$  dbo:Planet  
dbo:Planet  $\subseteq$  dbo:CelestialBody  
dbo:artificialSatelliteOf  $\subseteq$  dbo:satelliteOf

**Model-theoretic Semantics**

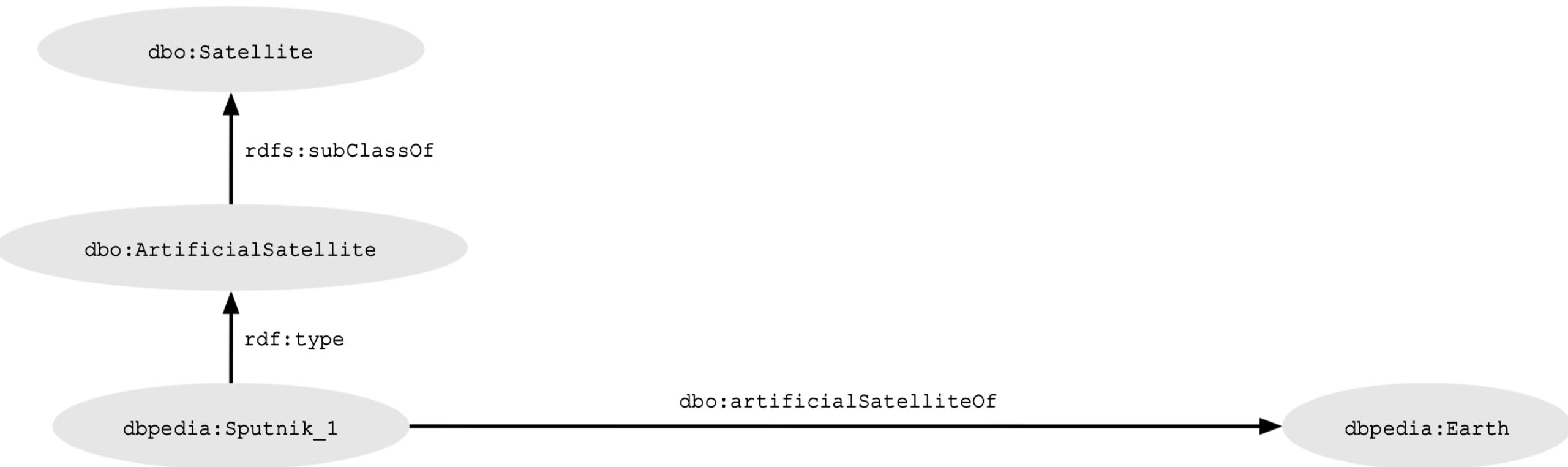


# Which Conclusions Can We Deduce With RDF(S)?

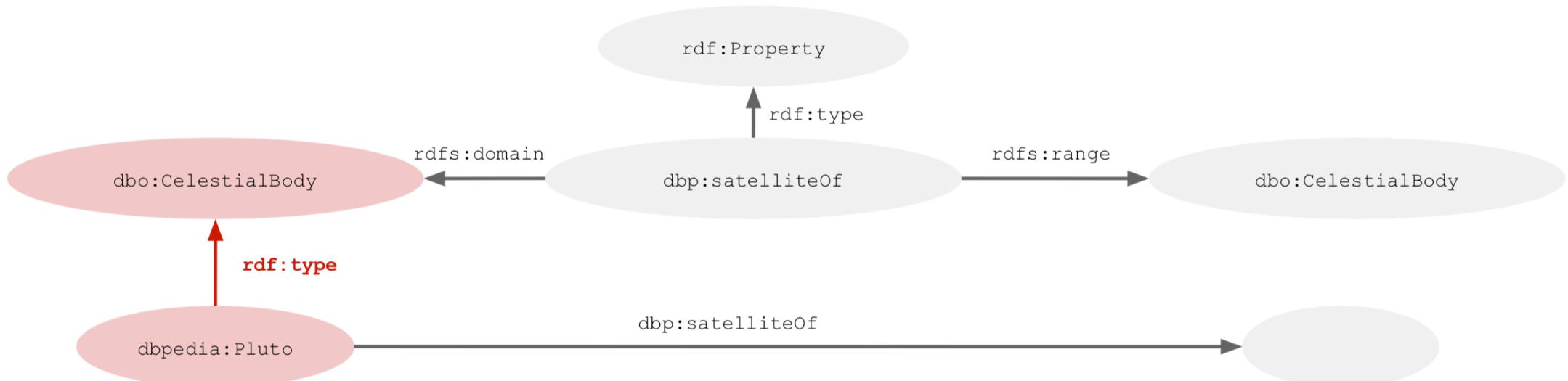




# Which Conclusions Can We Deduce With RDF(S)?

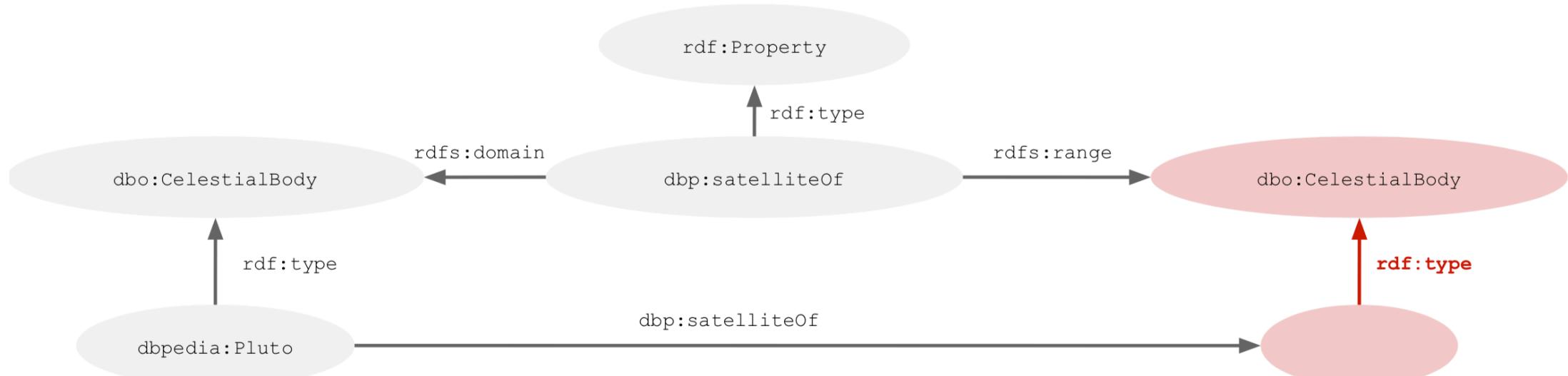


# Which Conclusions Can We Deduce With RDF(S)?



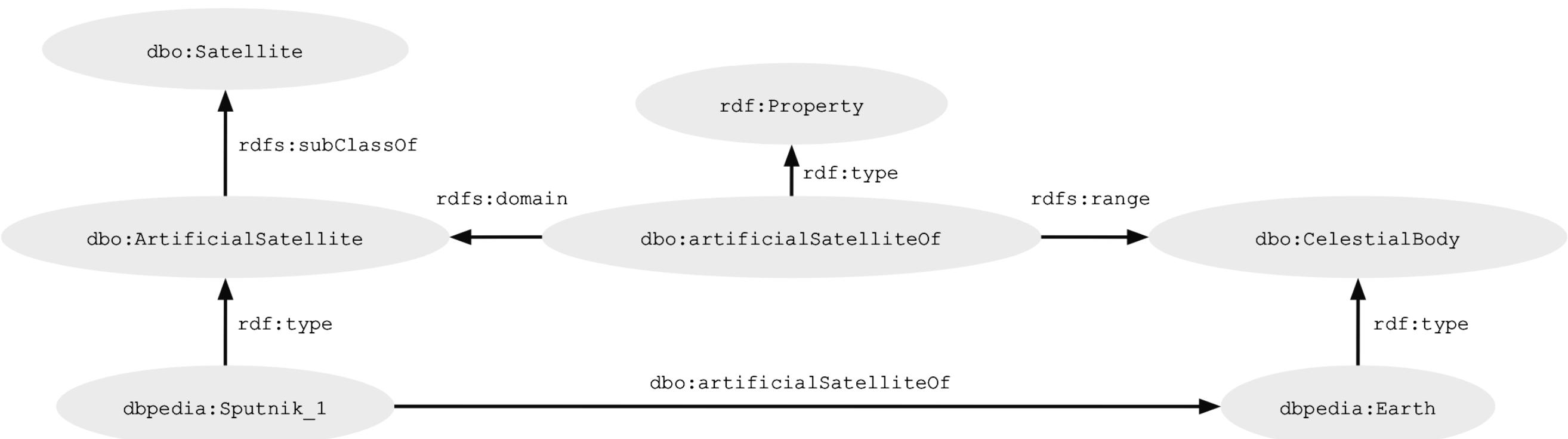
- Deduction of entity **class membership** from **domain** of one of its properties

# Which Conclusions Can We Deduce With RDF(S)?

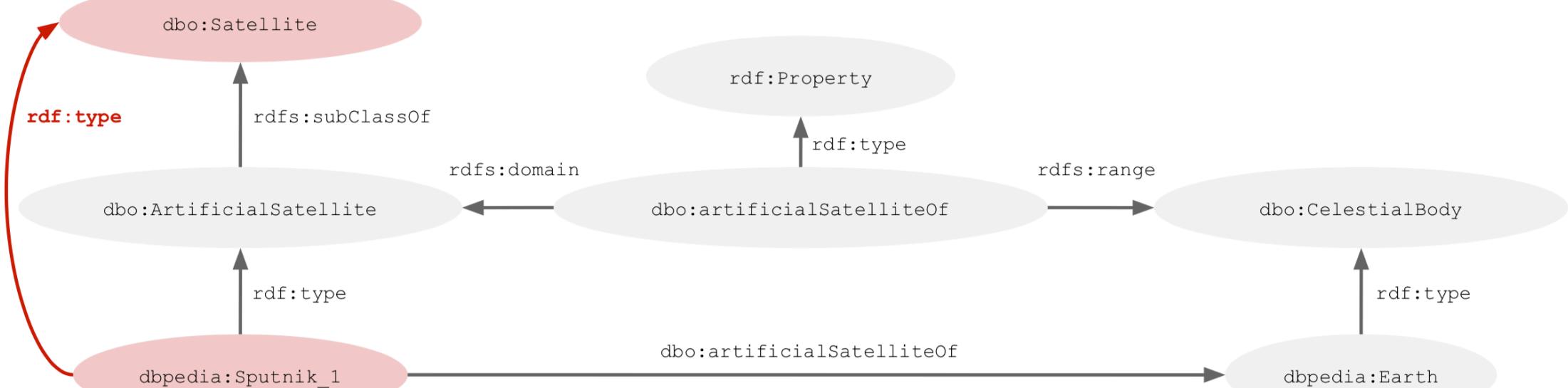


- Deduction of entity **class membership** from the **range** of one of its properties

# Which Conclusions Can We Deduce With RDF(S)?

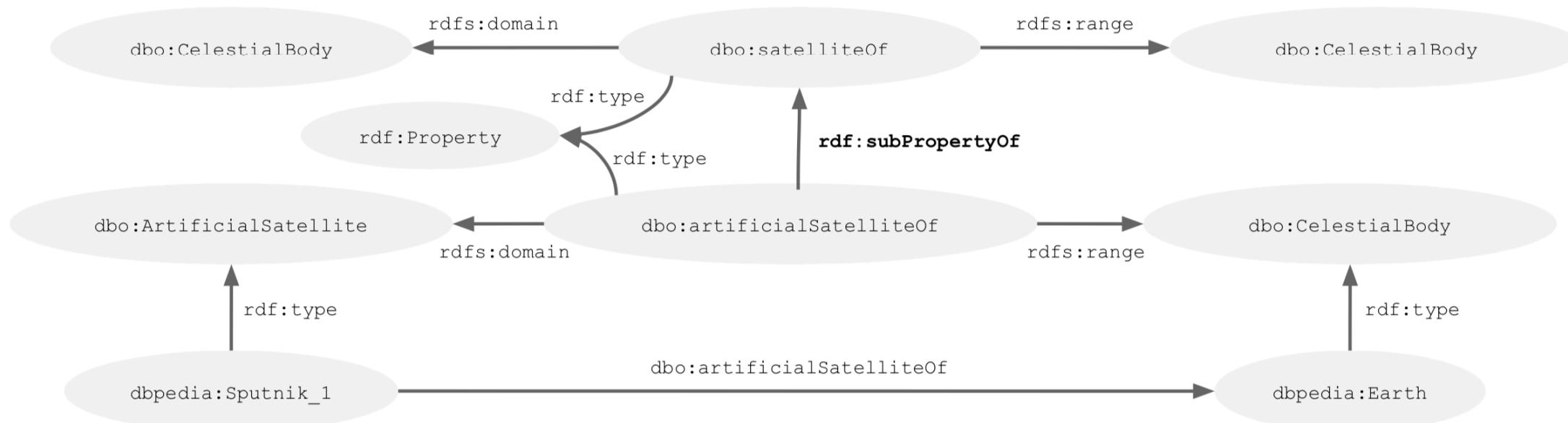


# Which Conclusions Can We Deduce With RDF(S)?

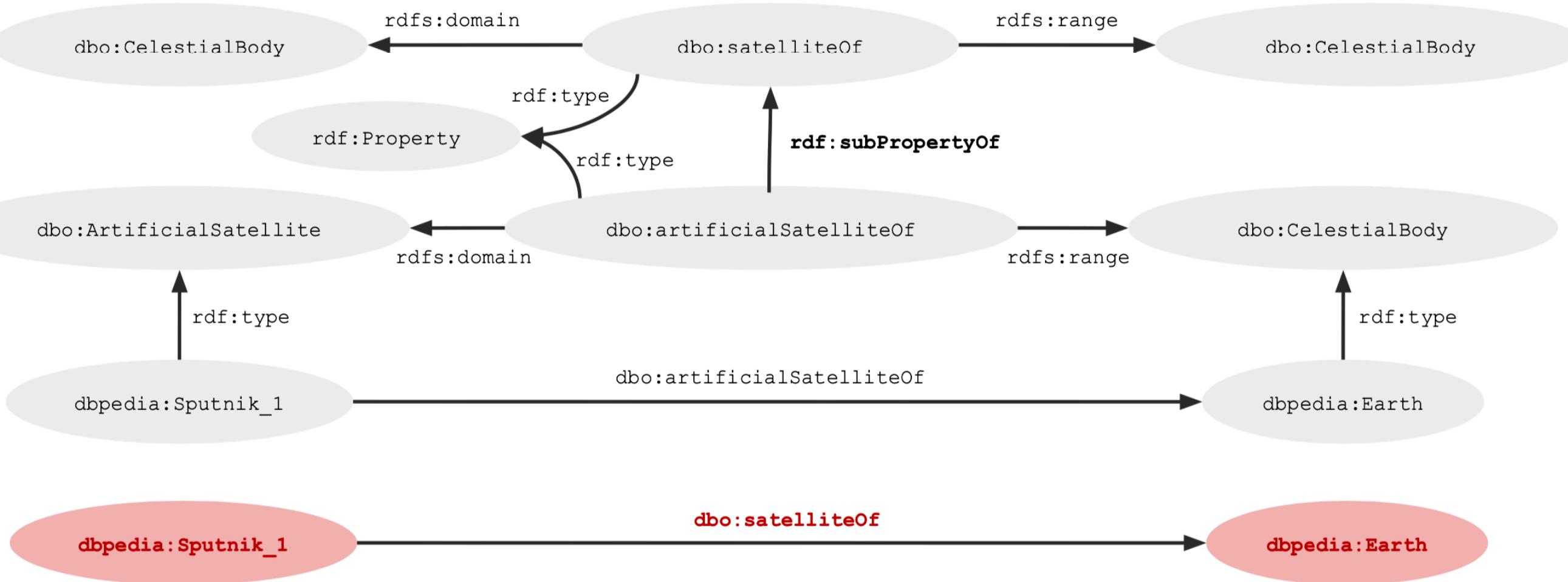


- Deduction of entity **superclass membership** from a **class hierarchy**.

# Which Conclusions Can We Deduce With RDF(S)?



# Which Conclusions Can We Deduce With RDF(S)?



- Deduction of entity **new facts** from **subproperty** relationships.