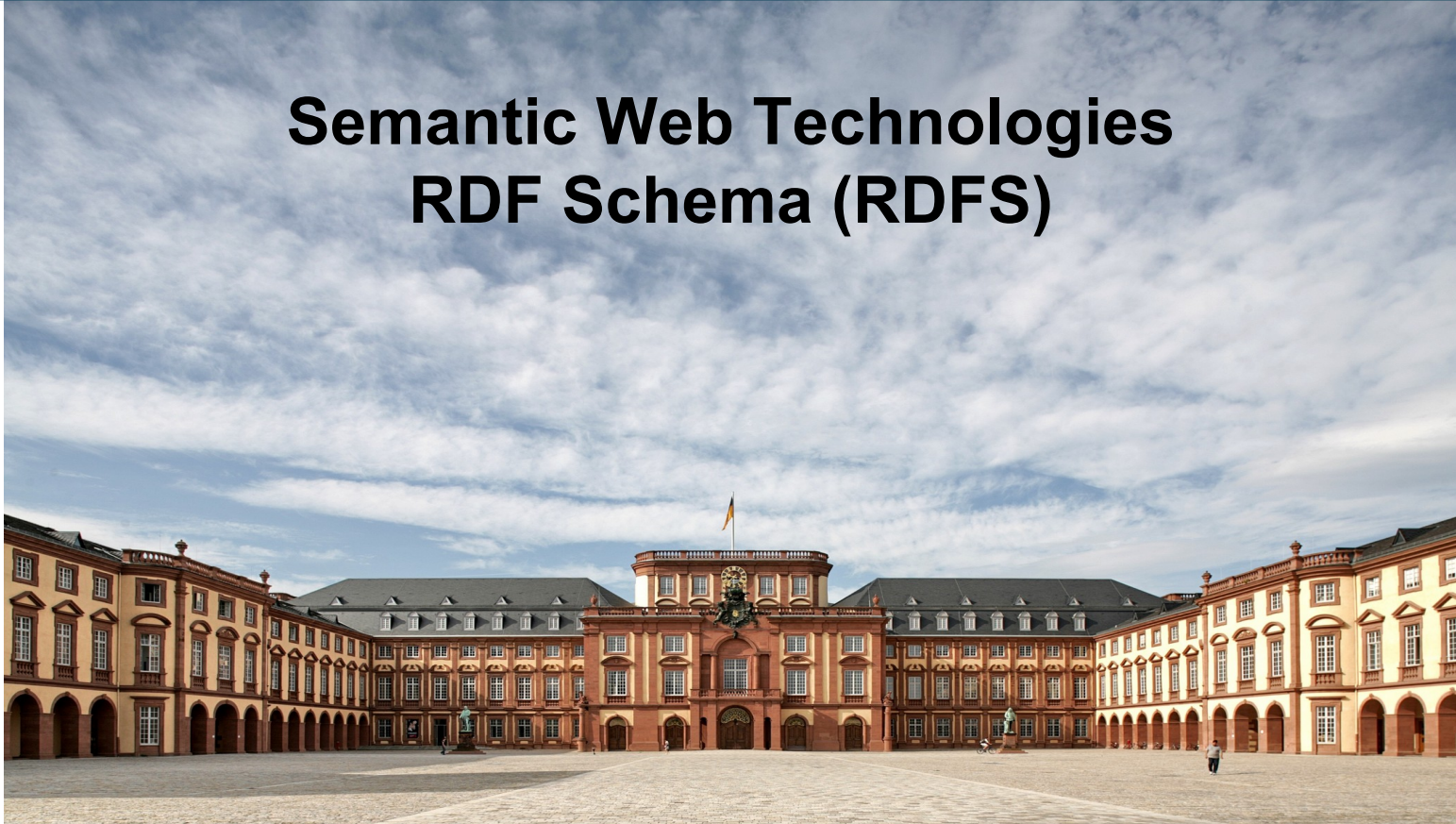


# Semantic Web Technologies

## RDF Schema (RDFS)



# Previously on “Semantic Web Technologies”

- Is RDF more powerful than XML?
- XML is a markup language for information
- In XML, arbitrary elements and attributes can be defined
- XML tag names are meaningless for a computer
- RDF is a markup language for information
- In RDF, arbitrary classes and predicates can be defined
- RDF class and predicate names are meaningless for a computer

# Today: Schemas and Ontologies

- They bring the Semantics to the Semantic Web (finally!)
  - Building simple ontologies with RDF Schema
  - Elements of RDF Schema
  - Automatic deduction with RDF Schema

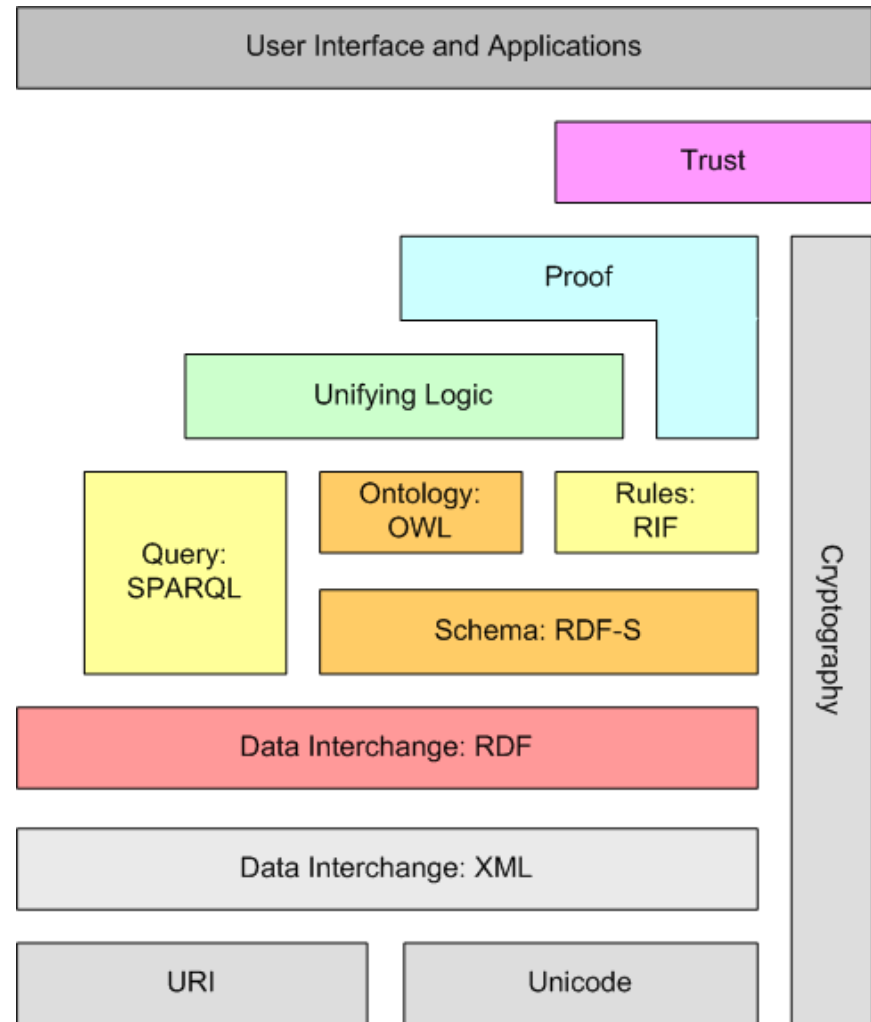
# Semantic Web – Architecture



here be dragons...

Semantic Web  
Technologies  
(This lecture)

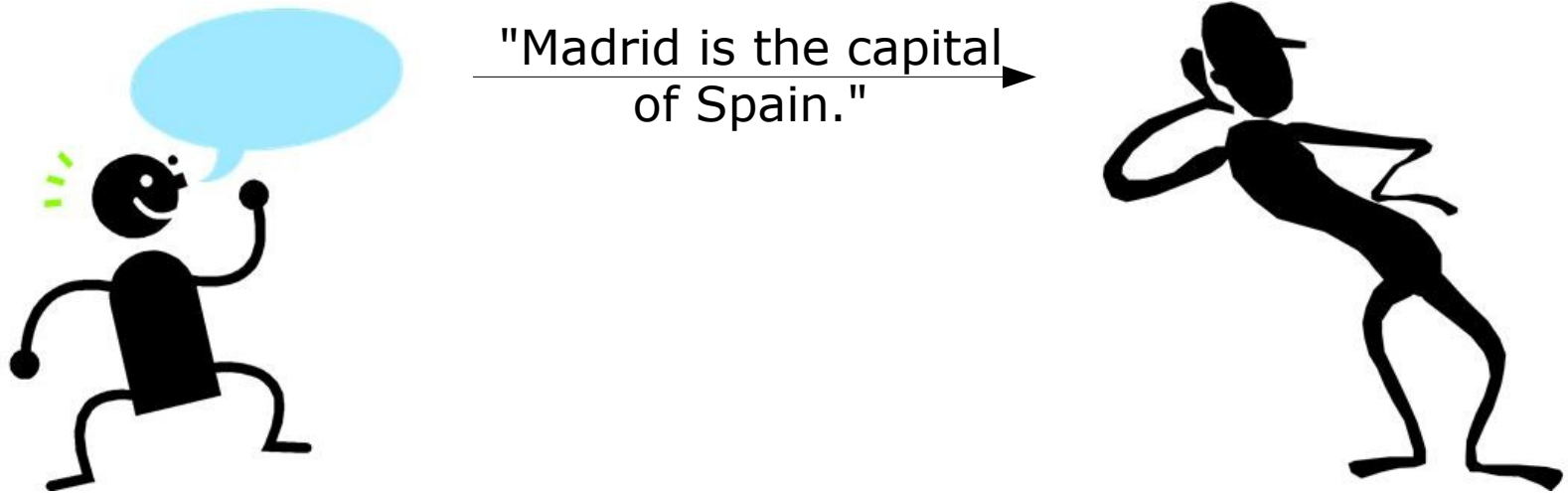
Technical  
Foundations



Berners-Lee (2009): *Semantic Web and Linked Data*  
<http://www.w3.org/2009/Talks/0120-campus-party-tbl/>

# What is Missing up to Now?

- Our mission: make computers understand information on the Web
- But what does *understand* actually mean?



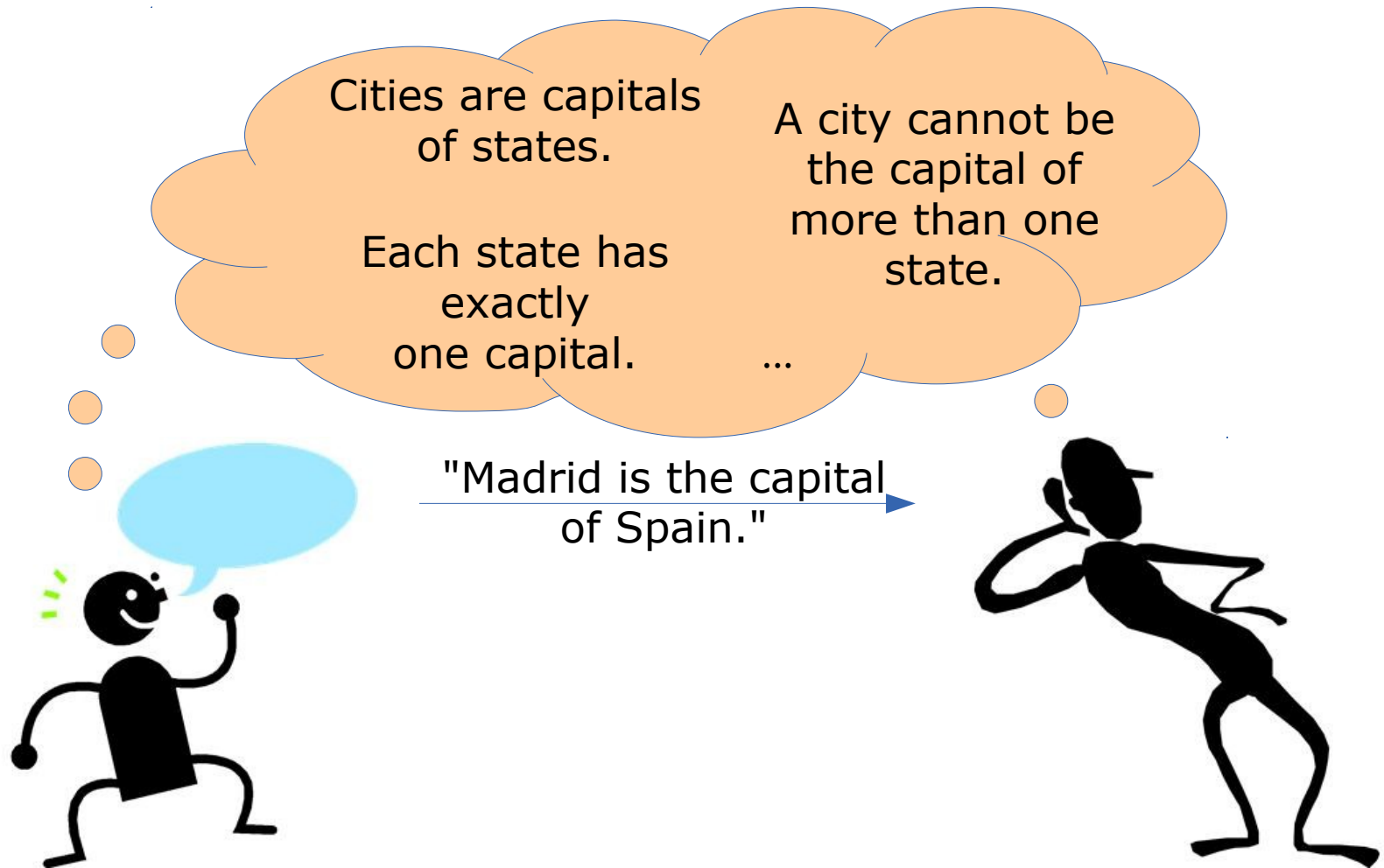
# Semantics

- Let's look at that sentence:
  - "Madrid is the capital of Spain."
- Published on the Semantic Web (i.e., using RDF):
  - `:Madrid :capitalOf :Spain .`
- How many pieces of information can we (i.e., humans) derive from that sentence?
  - (1 piece of information = 1 statement  $\langle S, P, O \rangle$ )
  - Estimations? Opinions?

# Semantics

- Let's look at that sentence:
  - "Madrid is the capital of Spain."
- We can get the following information:
  - "Madrid is the capital of Spain."
  - "Spain is a state."
  - "Madrid is a city."
  - "Madrid is located in Spain."
  - "Barcelona is not the capital of Spain."
  - "Madrid is not the capital of France."
  - "Madrid is not a state."
  - ...

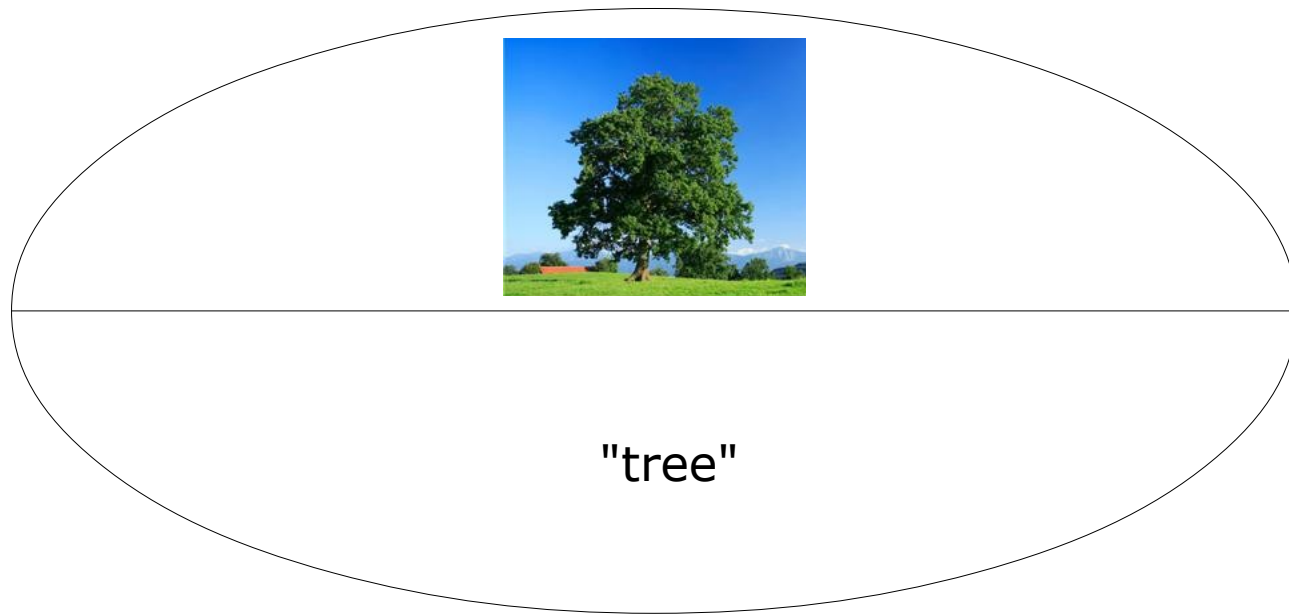
# How do Semantics Work?





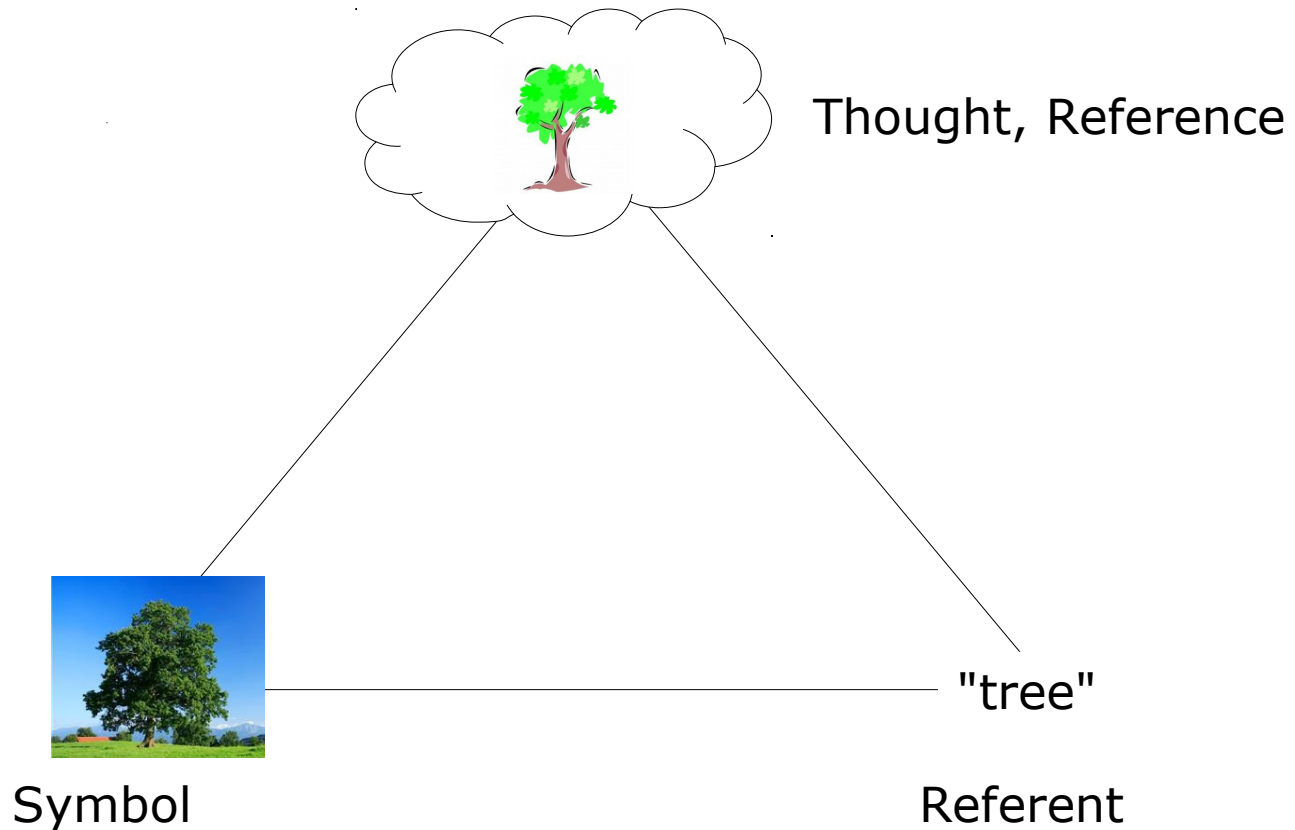
# An Excursion to Linguistics

- Saussure's idea of a *linguistic sign*
- Ferdinand de Saussure (1857-1913):
  - Signifier (signifiant) and signified (signifié) cannot be separated from each other



# An Excursion to Linguistics

- The triangle of reference



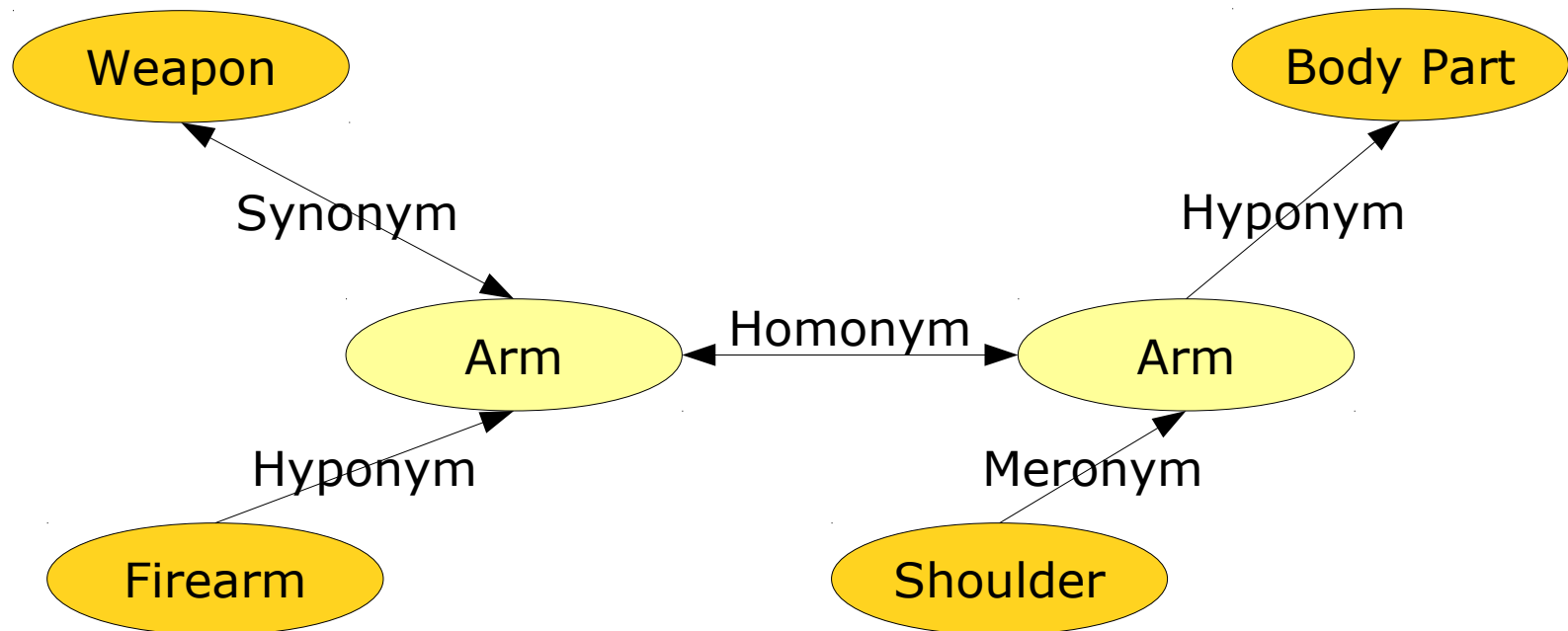
Charles Odgen (1923): *The Meaning of Meaning*.

# So, how do Semantics Work?

- Lexical semantics
  - Meaning of a word is defined by relations to other words
- Extensional semantics
  - Meaning of a word is defined by the set of its instances
- Intensional semantics, e.g., feature-based semantics
  - Meaning of a word is defined by features of the instances
- Prototype semantics
  - Meaning of a word is defined by proximity to a prototypical instance
- ...

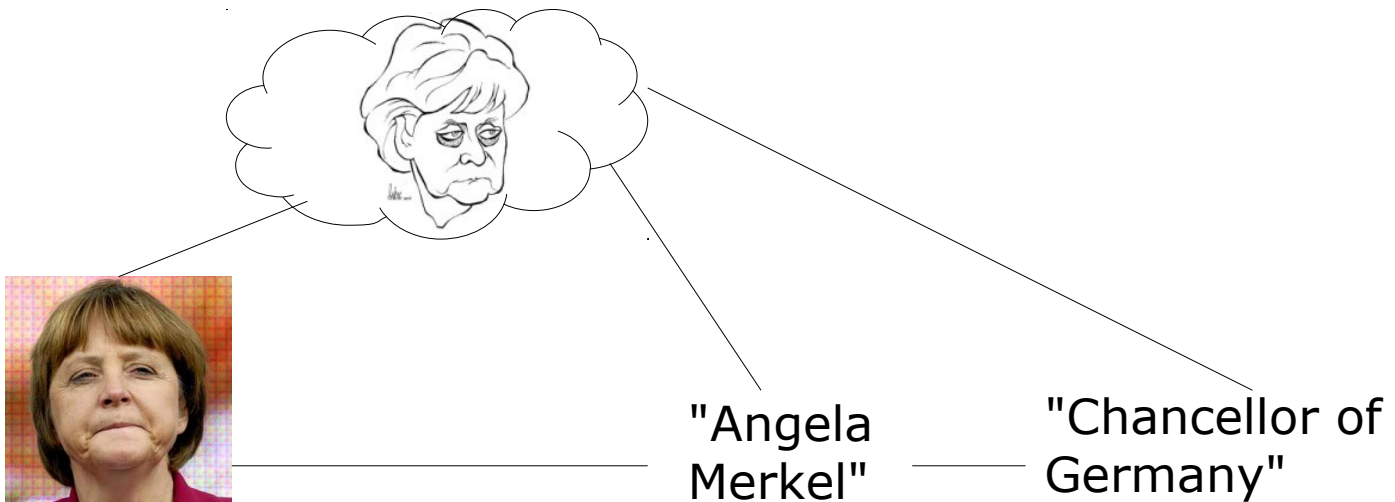
# Lexical Semantics

- Defining semantics by establishing relations between words



# Extensional Semantics

- Listing instances
  - EU members are Austria, Belgium, Bulgaria, ..., Sweden, UK.
- *Angela Merkel == Chancellor of Germany*
  - both terms have the same extension



# Intensional Semantics

- Describes features of things, i.e., *semes*
- A seme is a feature that distinguishes the meaning of two words

Word	has wings	can swim	has fur	can fly
Duck	+	+	-	+
Bird	+	O	-	O
Bee	+	-	-	+
Dolphin	-	+	-	-
...				

# Intensional vs. Extensional Semantics

- Intensionally different things can have the same extension
- Classic example: morning star and evening star

Word	Celestial body	bright	visible in the morning	visible in the evening
Morning star	+	+	+	-
Evening star	+	+	-	+
...				

- both have the same extension (i.e., Venus)

# Intensional vs. Extensional Semantics

- The extension can change over time without the intension changing
  - e.g., “student”
  - does that change the semantics?
- Intension may also change over time
  - technological achievements (e.g., intension of *ship*)
  - changes in moral values (e.g., intension of *marriage*)
- Extension may also be empty, e.g.
  - Unicorn
  - Martian
  - Yeti (?)



# Intensional vs. Extension Semantics

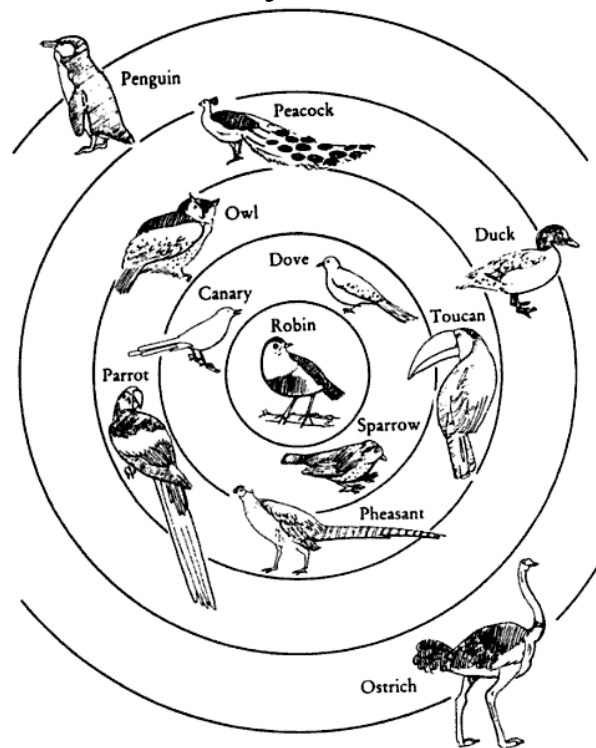
- ...explained by two well-known experts in the field :-)

# Prototype Semantics

- A small experiment:
  - Close your eyes, and imagine a bird!

# Prototype Semantics

- So far, intensional and extensional semantics are based on boolean logics (i.e., there's only “true” and “false”)
- Prototype Semantics: a more fuzzy variant



Jean Aitchison: Words in the Mind (1987)

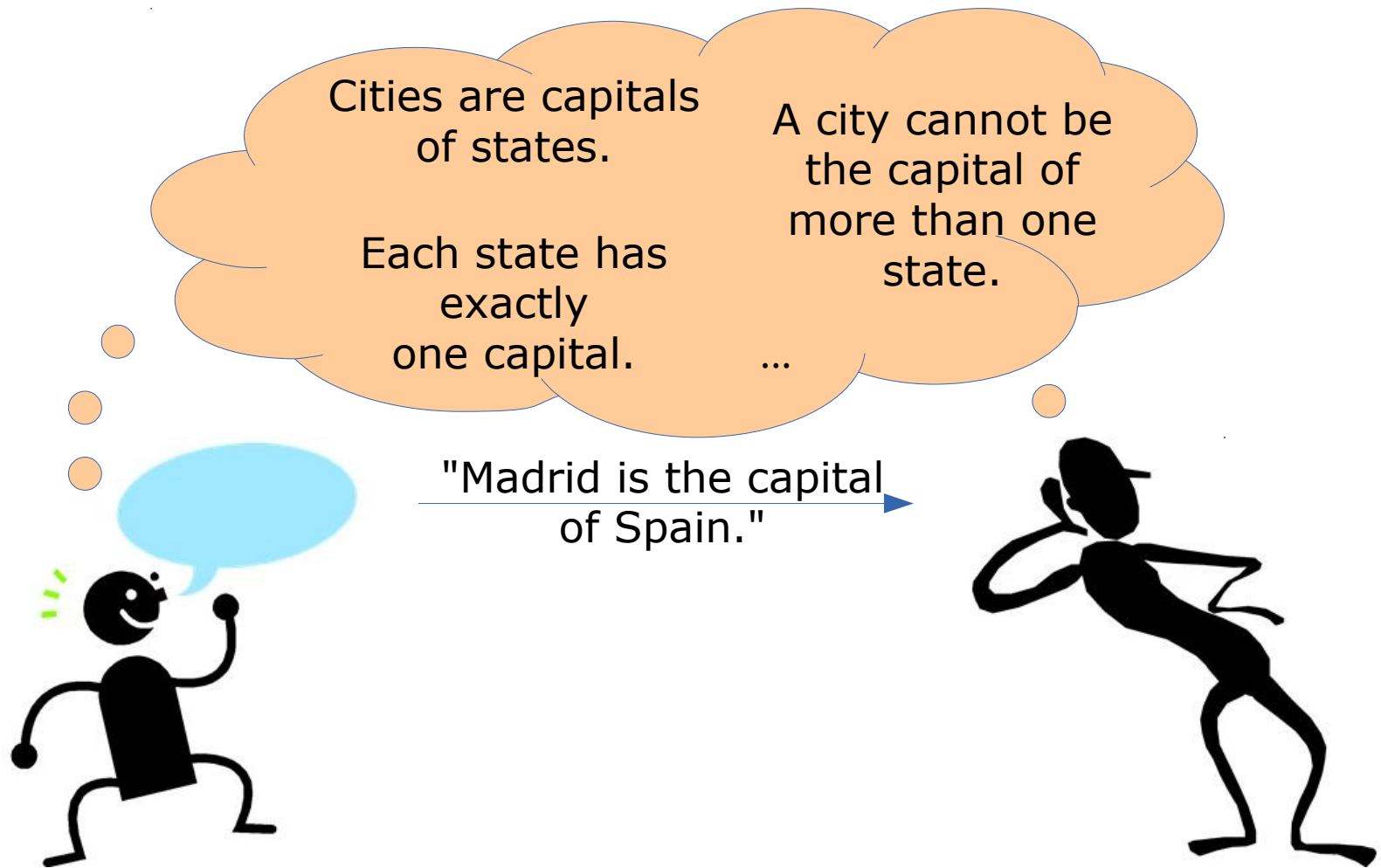
# How do Semantics Work?

- We have learned: Semantics define the meaning of words
- That is what we do in the Semantic Web
  - using methods from lexical, intensional, and extensional semantics

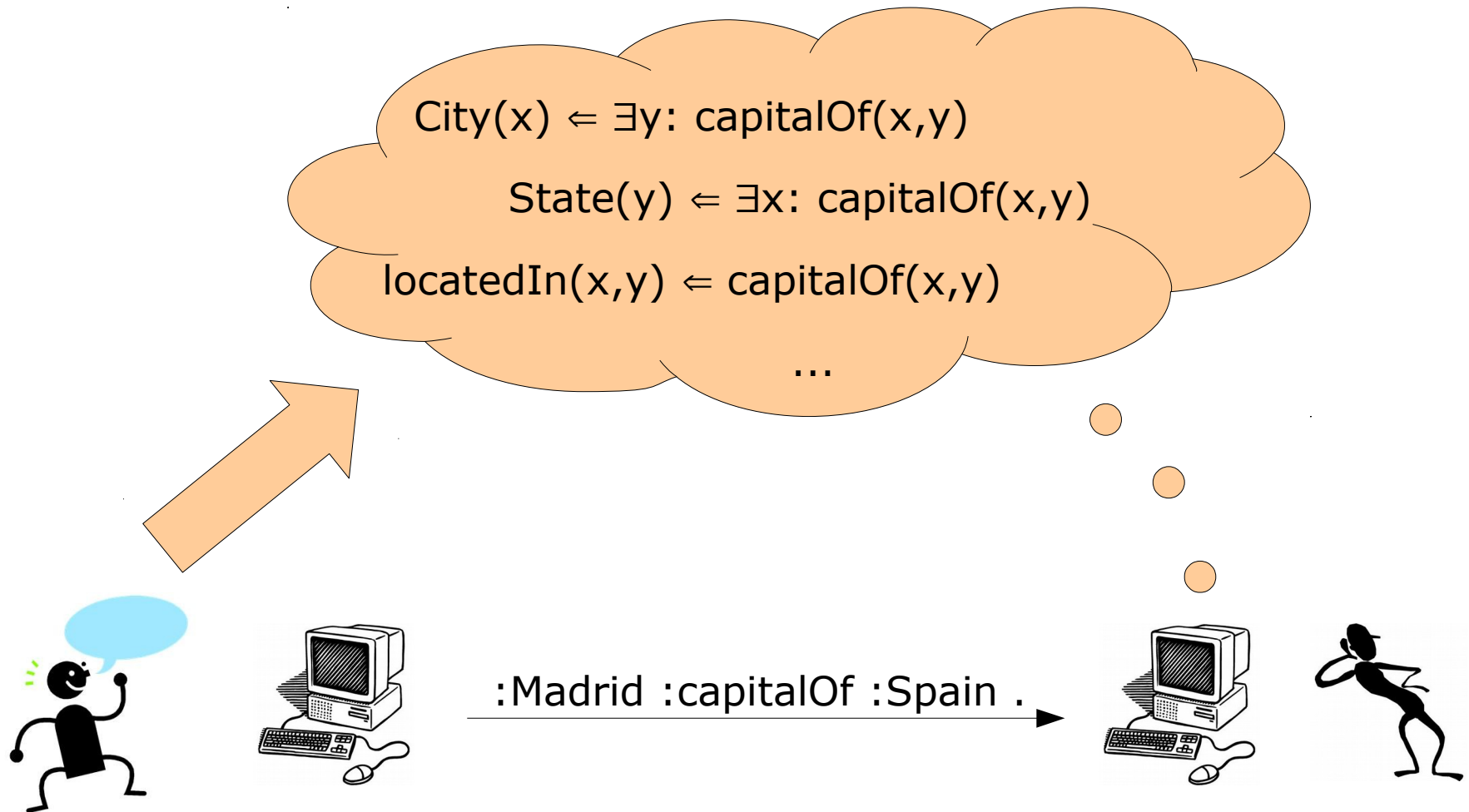


<http://walkinthewords.blogspot.com/2008/05/linguistic-cartoon-favorites-semantics.html>

# How do Semantics Work?



# Semantics in the Semantic Web



# Ontologies



$\text{City}(x) \Leftarrow \exists y: \text{capitalOf}(x,y)$

$\text{Country}(y) \Leftarrow \exists x: \text{capitalOf}(x,y)$

$\text{locatedIn}(x,y) \Leftarrow \text{capitalOf}(x,y)$

...

- "An ontology is an explicit specification of a conceptualization."<sup>1</sup>
- Ontologies encode the knowledge about a domain
- They form a common vocabulary
  - and describe the semantics of its terms

<sup>1</sup> Gruber (1993): *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*.  
In: International Journal Human-Computer Studies Vol. 43, Issues 5-6, pp. 907-928.

# What is an Ontology?

- Ontology (without *a* or *the*) is the philosophical study of being
  - greek: *όντος* (things that are), *λόγος* (the study)
  - A sub discipline of philosophy
- In computer science (with *a* or *the*)
  - a formalized description of a domain
  - a shared vocabulary
  - a logical theory



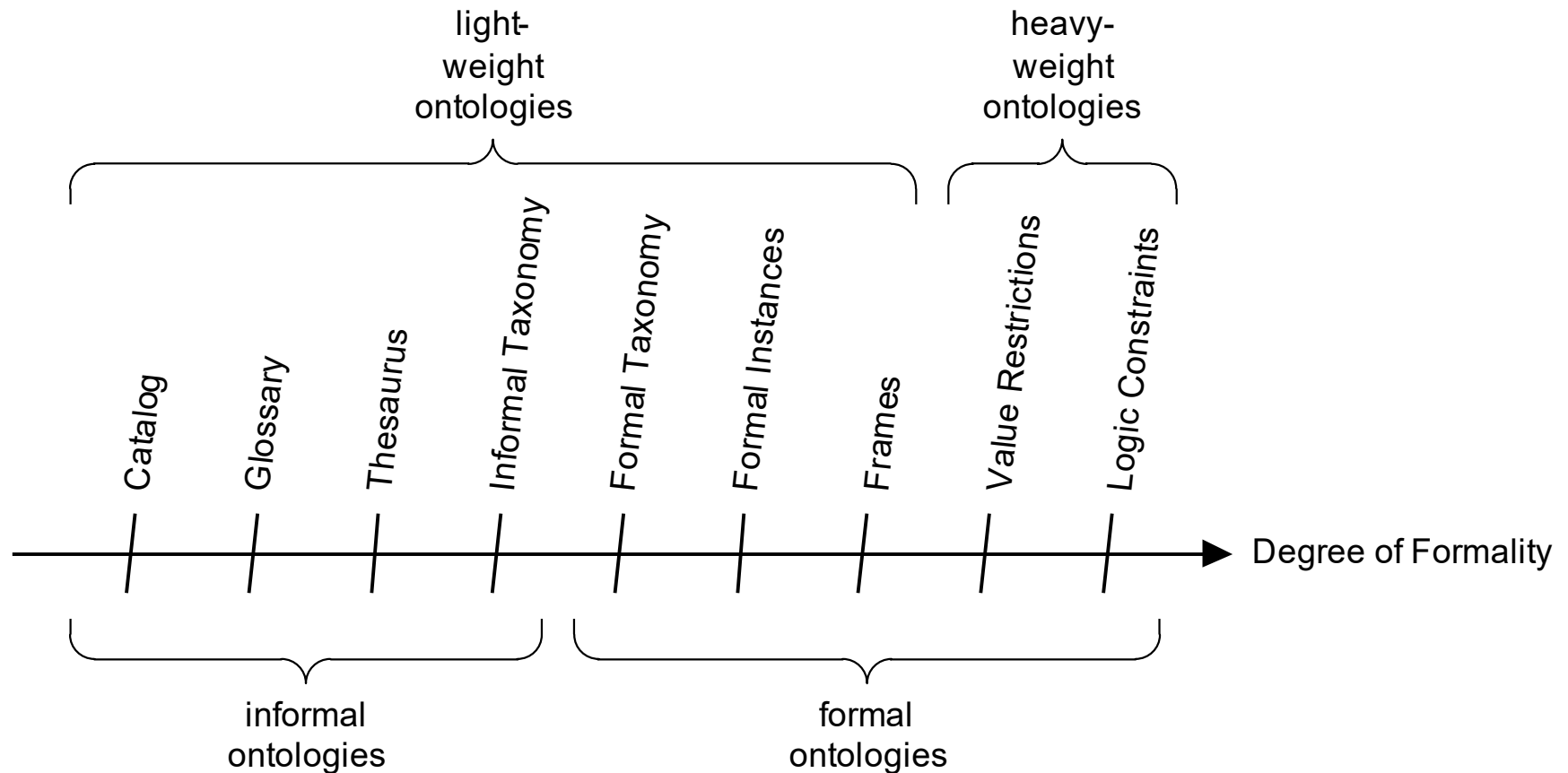
# Ontologies – Further Definitions

- Guarino und Giaretta (1995):  
"a logical theory which gives an **explicit, partial** account of a conceptualization"
- Uschold und Gruninger (1996):  
"**shared** understanding of some domain of interest"  
"an **explicit** account or representation of some **part of** a conceptualisation"
- Guarino (1998):  
"a set of **logical axioms** designed to account for the intended meaning of a vocabulary"

# Essential Properties of Ontologies

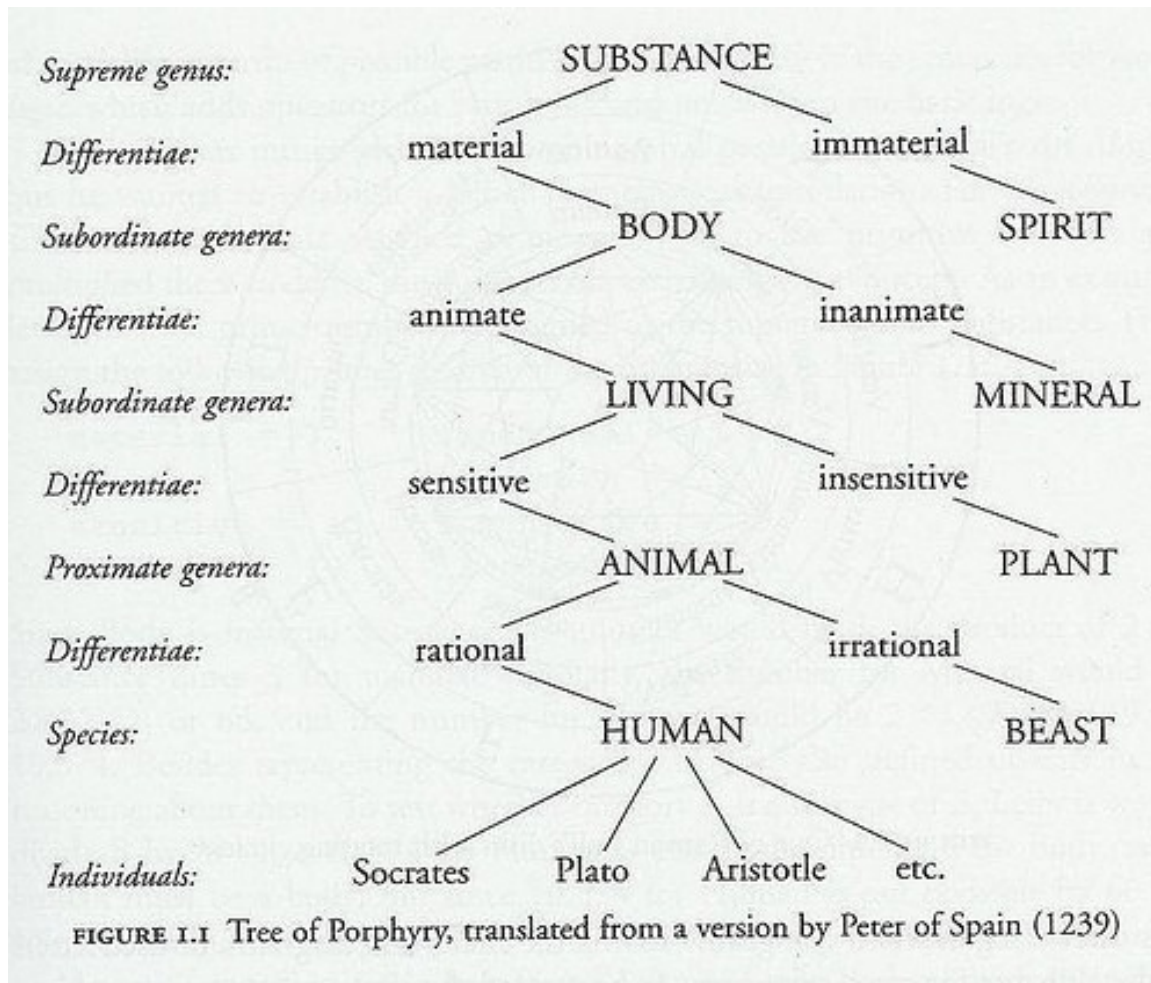
- Explicit
  - Meaning is not “hidden” between the lines
- Formal
  - e.g., using logic or rule languages
- Shared
  - Martin Hepp: "Autists don't build ontologies"
  - An ontology just for one person does not make much sense
- Partial
  - There will (probably) never be a full ontology of everything in the world

# Classifications of Ontologies



Lassila & McGuinness (2001): *The Role of Frame-Based Representation on the Semantic Web*.  
In: Linköping Electronic Articles in Computer and Information Science 6(5).

# The Oldest Ontology



Porphyry, Greek philosopher, ca. 234-305

# Encoding Simple Ontologies: RDFS

- A W3C Standard since 2004
- Most important element: classes



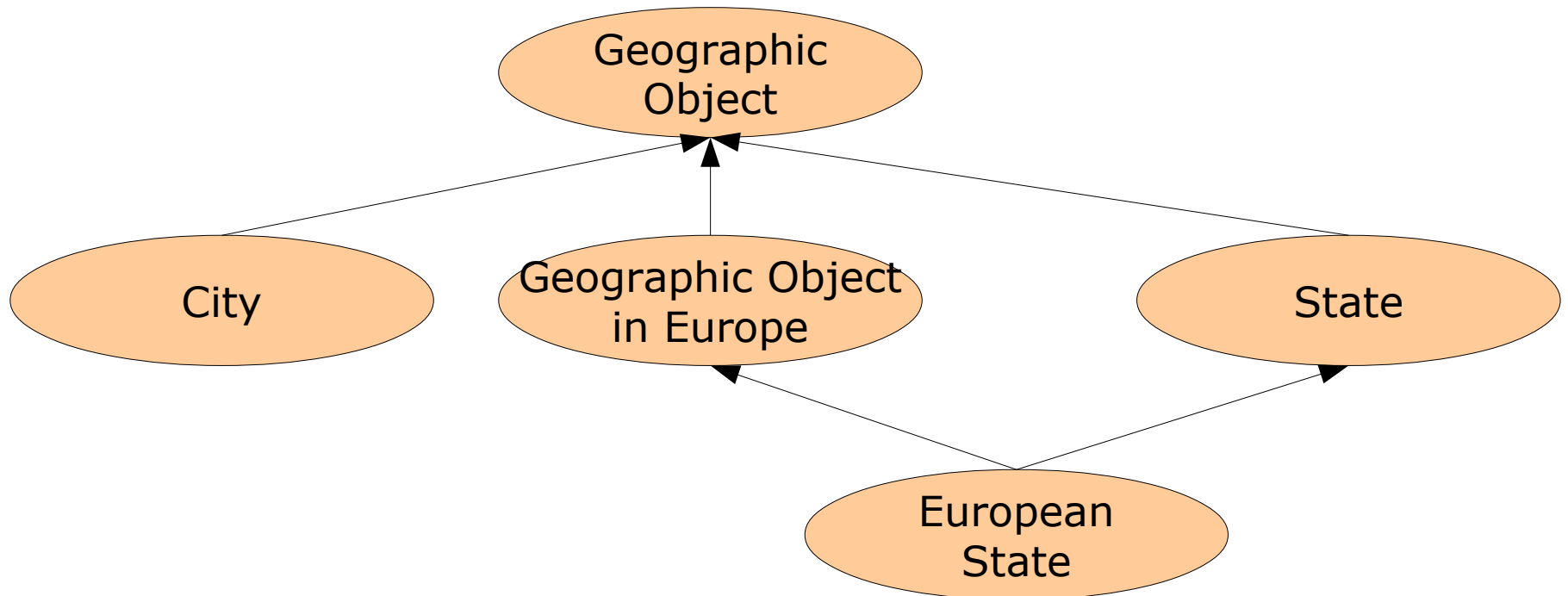
```
:State a rdfs:Class .
```

- Classes form hierarchies

```
:EuropeanState rdfs:subClassOf :State .
```

# Class Hierarchies in RDF Schema

- Multiple inheritance is possible



Convention for this course: unlabeled arrows = `rdfs:subClassOf`

# Properties in RDF Schema

- Properties are the other important element
- resemble two-valued predicates in predicate logic

```
:Madrid :capitalOf :Spain .  
:capitalOf a rdf:Property .
```

- Properties also form hierarchies

```
:capitalOf rdfs:subPropertyOf :locatedIn .
```

# Domains and Ranges of Properties

- In general, properties exist independently from classes
  - i.e., they are *first class citizens*
  - this is different than OOP or ERM
- Defining the domain and range of a property:
  - `:capitalOf rdfs:domain :City .`
  - `:capitalOf rdfs:range :Country .`
- Domain and range are inherited by sub properties
  - They can also be further restricted



# Predefined Properties

- We have already seen

`rdf:type`

`rdfs:subClassOf`

`rdfs:subPropertyOf`

`rdfs:domain`

`rdfs:range`

# Further Predefined Properties

- Labels:

```
:Germany rdfs:label "Deutschland"@de .
```

```
:Germany rdfs:label "Germany"@en .
```

- Comments:

```
:Germany rdfs:comment "Germany as a political  
entity."@en .
```

- Links to other resources:

```
:Germany rdfs:seeAlso <http://www.deutschland.de/> .
```

- Link to defining schema:

```
:Country rdfs:isDefinedBy  
<http://foo.bar/countries.rdfs> .
```

# URIs vs. Labels

- A URI is only a unique identifier
  - it does not need to be interpretable

`http://www.countries.org/4327893`

- Labels are made for human interpretation
- ...and can come in different languages:

`countries:4327893 rdfs:label "Deutschland"@de .`

`countries:4327893 rdfs:label "Germany"@en .`

`countries:4327893 rdfs:label "Tyskland"@sv .`

`...`

# URIs vs. Labels

- Labels and comments can also be assigned to RDFS elements:

```
:Country a rdfs:Class .  
:Country rdfs:label "Land"@de .  
:Country rdfs:label "Country"@en .  
  
:locatedIn a rdf:Property .  
:locatedIn rdfs:label "liegt in"@de .  
:locatedIn rdfs:label "is located in"@en .  
:locatedIn rdfs:comment "refers to geography"@en .
```

# RDF Schema and RDF

- Every RDF Schema document is also an RDF document
- This means: all properties of RDF also hold for RDFS!

- Non-unique Naming Assumption

```
schema1:Country a rdfs:Class .
```

```
schema2:State a rdfs:Class .
```

- Open World Assumption

```
:Country rdfs:subClassOf :GeographicObject .
```

```
:City rdfs:subClassOf :GeographicObject .
```

# Our First Ontology

- States, cities, and capitals

```
:State a rdfs:Class .
```

```
:City a rdfs:Class .
```

```
:locatedIn a rdf:Property .
```

```
:capitalOf rdfs:subPropertyOf :locatedIn .
```

```
:capitalOf rdfs:domain :City .
```

```
:capitalOf rdfs:range :State .
```

```
:Madrid :capitalOf :Spain .
```

Definition of the  
Terminology  
(T-Box)

Definition of the  
Assertions  
(A-box)

# What do We Gain Now?

```
:Country a rdfs:Class .  
:City a rdfs:Class .  
:locatedIn a rdfs:Property .  
:capitalOf rdfs:subPropertyOf :locatedIn .  
:capitalOf rdfs:domain :City .  
:capitalOf rdfs:range :Country .
```



:Madrid :capitalOf :Spain .



# What do We Gain Now?

```
:Madrid :capitalOf :Spain .  
+ :capitalOf rdfs:domain :City  
→ :Madrid a :City .
```

```
:Madrid :capitalOf :Spain .  
+ :capitalOf rdfs:range:Country  
→ :Spain a :Country .
```

```
:Madrid :capitalOf :Spain .  
+ :capitalOf rdfs:subPropertyOf :locatedIn .  
→ :Madrid :locatedIn :Spain .
```

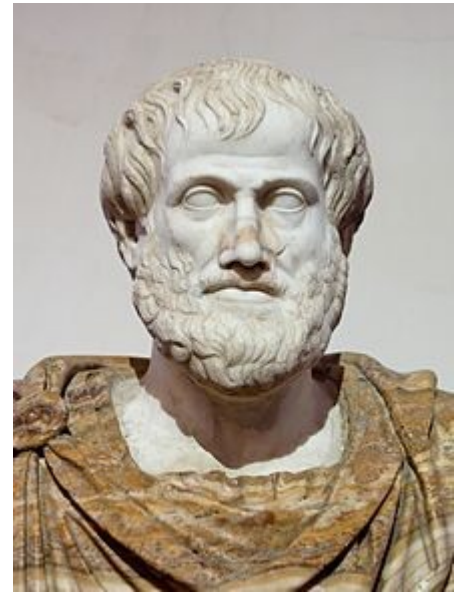


# Reasoning with RDF

- RDF Schema allows for *deductive* reasoning on RDF
- This means:
  - given facts and rules,
  - we can derive new facts
- The corresponding tools are called *reasoner*
- Opposite of deduction: *induction*
  - deriving models from facts
  - see, e.g., lectures on data mining and machine learning

# A Bit of History

- Aristotle (384 – 322 BC)
- Syllogisms
  - Deriving facts using rules
- Example:
  - All men are mortal.
  - Socrates is a man.
  - Socrates is mortal.



# A Bit of History



Cartoon Copyright: Randy Glasbergen, <http://www.glasbergen.com/>

# Interpretation and Entailment

- Entailment
  - The set of all consequences of a graph
- Mapping a graph to an entailment is called *interpretation*
- Simplest Interpretation:
  - $\langle s, p, o \rangle \in G \rightarrow \langle s, p, o \rangle \in \text{Entailment}$
- This interpretation creates all statements explicitly contained in the graph.
- But the *implicit* statements are the interesting ones!

# Interpretation using Deduction Rules

- RDF interpretation can be done using RDFS deduction rules
- Those create an entailment
  - using existing resources, literals, and properties
  - creating additional triples like <s,p,o>
  - e.g.,
    - <Madrid, rdf:type, City>
    - <Madrid, located\_in, Spain>
- Note:
  - no *new* resources, literals, or properties are created!

# Reasoning with Deduction Rules

- Deduction rules are an interpretation function
- Simple reasoning algorithm (a.k.a. *forward chaining*):

```
Given: an RDF Graph G
a set of deduction rules R
Entailment E = G
Repeat
    M := { }
    For all rules in R
        For each statement S in E
            Apply R to S
            If E does not contain consequence
                Add consequence to M
    Add all elements in M to E
until M = { }
```

# Deduction Rules for RDF Schema (1)

ID	Condition	Consequence
rdf1	<code>s p o .</code>	<code>p rdf:type rdf:Property .</code>
rdfs1	<code>s p l .</code> <code>l is a Literal</code>	<code>l rdf:type rdfs:Literal .</code>
rdfs2	<code>s p o .</code> <code>p rdfs:domain c .</code>	<code>s rdf:type c .</code>
rdfs3	<code>s p o .</code> <code>p rdfs:range c .</code>	<code>o rdf:type c .</code>
rdfs4a	<code>s p o .</code>	<code>s rdf:type rdfs:Resource .</code>
rdfs4b	<code>s p o .</code> <code>o is a URI or blank node</code>	<code>o rdf:type rdfs:Resource .</code>
rdfs5	<code>p1 rdfs:subPropertyOf p2 .</code> <code>p2 rdfs:subPropertyOf P3 .</code>	<code>p1 rdfs:subPropertyOf p3 .</code>
rdfs6	<code>p rdf:type rdf:Property .</code>	<code>p rdfs:subPropertyOf p .</code>

W3C (2004): *RDF Semantics*. <http://www.w3.org/TR/rdf-mt/>

# Deduction Rules for RDF Schema (2)

ID	Condition	Consequence
rdfs7	<code>p1 rdfs:subPropertyOf p2 .</code> <code>s p1 o .</code>	<code>s p2 o .</code>
rdfs8	<code>c rdf:type rdfs:Class .</code>	<code>c rdfs:subClassOf</code> <code>rdfs:Resource .</code>
rdfs9	<code>s rdf:type c1 .</code> <code>c1 rdfs:subClassOf c2 .</code>	<code>s rdf:type c2 .</code>
rdfs10	<code>c rdf:type rdfs:Class .</code>	<code>c rdfs:subClassOf c .</code>
rdfs11	<code>c1 rdfs:subClassOf c2 .</code> <code>c2 rdfs:subClassOf c3 .</code>	<code>c1 rdfs:subClassOf c3 .</code>
rdfs12	<code>p rdf:type rdfs:container-</code> <code>MembershipProperty .</code>	<code>p rdfs:subPropertyOf</code> <code>rdfs:member .</code>
rdfs13	<code>d rdf:type rdfs:Datatype .</code>	<code>d rdfs:subClassOf</code> <code>rdfs:Literal .</code>

W3C (2004): *RDF Semantics*. <http://www.w3.org/TR/rdf-mt/>



# Forward Chaining

- Let's reconsider this example

```
:State a rdfs:Class .  
:State rdfs:subClassof :GeographicEntity .  
:City a rdfs:Class .  
:City rdfs:subClassOf :GeographicEntity .  
:capitalOf rdfs:subPropertyOf :locatedIn .  
:capitalOf rdfs:domain :City .  
:capitalOf rdfs:range :State .  
  
:Madrid :capitalOf :Spain .
```

# Applying Deduction Rules

- Another Example

```
:Employee a rdfs:Class .  
:Employee rdfs:subClassOf :Human .  
:Room a rdfs:Class .  
:worksIn rdfs:subPropertyOf :hasOffice .  
:hasOffice rdfs:domain :Employee .  
:hasOffice rdfs:range :Room .  
  
:Tim :worksIn :D0815 .
```

# Applying Deduction Rules

- Example:

```
:Tim :worksIn :D0815 .  
:worksIn rdfs:subPropertyOf :hasOffice .
```

ID	Condition	Consequence
rdfs7	p1 rdfs:subPropertyOf p2 . s p1 o .	s p2 o.

```
→ :Tim :hasOffice :D0815 .
```

# Applying Deduction Rules

- Example:

```
:Tim :hasOffice :D0815 .  
:hasOffice rdfs:domain :Employee .
```

ID	Condition	Consequence
rdfs2	s p o . p rdfs:domain c .	s rdf:type c .

```
→ :Tim rdf:type :Employee .
```

# Applying Deduction Rules

- Example:

```
:Tim rdf:type :Employee.  
:Employee rdfs:subClassOf :Human .
```

ID	Condition	Consequence
rdfs9	<pre>s rdf:type c1 . c1 rdfs:subClassOf c2 .</pre>	<pre>s rdf:type c2 .</pre>

→ :Tim rdf:type :Human .

# What if there are Multiple Domains/Ranges?

- Example for social networks:

```
:knows rdfs:domain :Person .
```

```
:knows rdfs:domain :MemberOfSocialNetwork .
```

- What should be the semantics here?
  - Everybody who knows someone  
is a person *and* a member of a social network
  - Everybody who knows someone  
is a person *or* a member of a social network

# The Rules will Tell Us

```
                :knows rdfs:domain :Person.                (a0)
                :knows rdfs:domain :MemberOfSocialNetwork . (a1)
                :Peter :knows :Stephen .                    (a2)
(rdfs2+a0+a2)   :Peter rdf:type :Person .                    (a3)
(rdfs2+a1+a2)   :Peter rdf:type :MemberOfSocialNetwork .    (a4)
                ...
```

- This chain works for each object
  - it is always contained in both classes
    - i.e., the intersection semantics hold

# What have We Gained?

- Let's look at that sentence:
  - "Madrid is the capital of Spain."
- We can get the following information:
  - "Madrid is the capital of Spain." ✓
  - "Spain is a state." ✓
  - "Madrid is a city." ✓
  - "Madrid is located in Spain." ✓
  - "Barcelona is not the capital of Spain." ✗
  - "Madrid is not the capital of France." ✗
  - "Madrid is not a state." ✗
  - ...



# What we Cannot Express (up to Now)

- "Every state has *exactly one* capital"
  - Property cardinalities
- "Every city can only be the capital of one state."
  - Functional properties
- "A city cannot be a state at the same time."
  - Class disjointness
- ...
- For those, we need more expressive languages than RDFS!

# What we Cannot Express (up to Now)

- "Every state has *exactly* one capital"
  - i.e., "A state *cannot* have more than one capital."
- "Every city can only be the capital of one state."
  - i.e., "A city *cannot* be the capital of two *different* states."
- "A city *cannot* be a state at the same time."

# What we Cannot Express (up to Now)

- Note: there is no negation in RDF and RDFS
- This means, we cannot produce any contradictions
  - This makes reasoning easy
  - But it also restricts the utility
  - Example:
    - Mammals do not lay eggs
    - Penguins lay eggs
    - Penguins are not mammals
- We will get to know formalisms that support negation
  - and learn how to do reasoning with them

# What we Cannot Express (up to Now)

- The missing negation perfectly fits the AAA principle
  - Anybody can say anything about anything
- ...and the Open World Assumption
- Any new knowledge will always fit to the knowledge that is already there
  - This principle is called “monotonicity”

# What we Cannot Express (up to Now)

- Kurt Gödel (1906-1978)
- Logic systems are either
  - not very powerful or
  - not free of contradictions
- RDF Schema belongs to the first class



# What we Cannot Express (up to Now)

- Jim Hendler (\*1957)
- "A little semantics goes a long way."



# Just a moment

- "We cannot produce any contradictions"
- so what about
  - :Peter a :Baby .
  - :Peter a :Adult .
- That is a contradiction!
- Well, it is – for us human beings
- But a computer will not know
  - Non-unique name assumption!

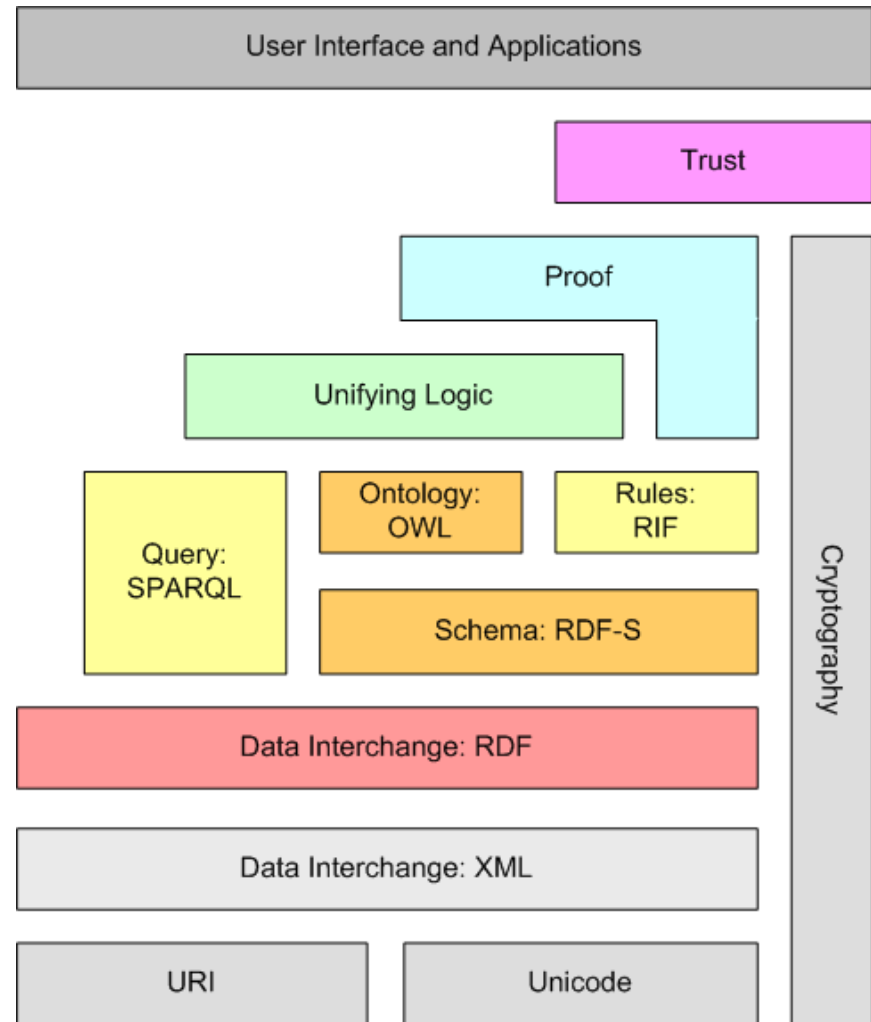
# Semantic Web – Architecture



here be dragons...

Semantic Web  
Technologies  
(This lecture)

Technical  
Foundations



Berners-Lee (2009): *Semantic Web and Linked Data*  
<http://www.w3.org/2009/Talks/0120-campus-party-tbl/>



# Questions?

