Knowledge Modeling

IE650 Knowledge Graphs

Knowledge Modeling & Ontology Engineering

- How should the knowledge in a KG be modeled?
  - Which classes of entities do we have?
  - Which relations connect them?
  - Which constraints hold for them?

    → these questions are defined in the ontology of the knowledge graph
- How we have built ontologies so far
  - Read the requirements
  - Pick a starting point at random
  - Start playing around in Protégé
  - Trial and error driven
- That was rather "Ontology Hacking" than "Ontology Engineering"

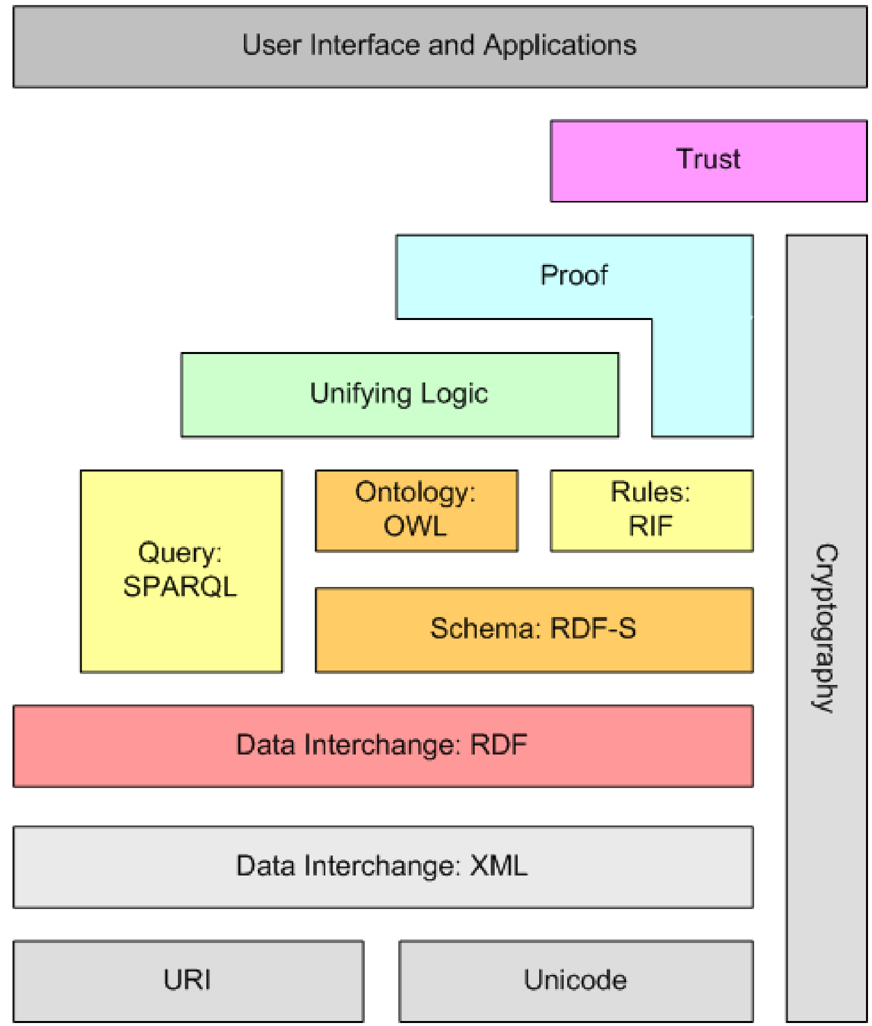
Semantic Web Technology Stack

UNIVERSITY OF MANNHEIM
Data and Web Science Group

here be dragons...

Knowledge Graphs
(This lecture)

Technical
Foundations



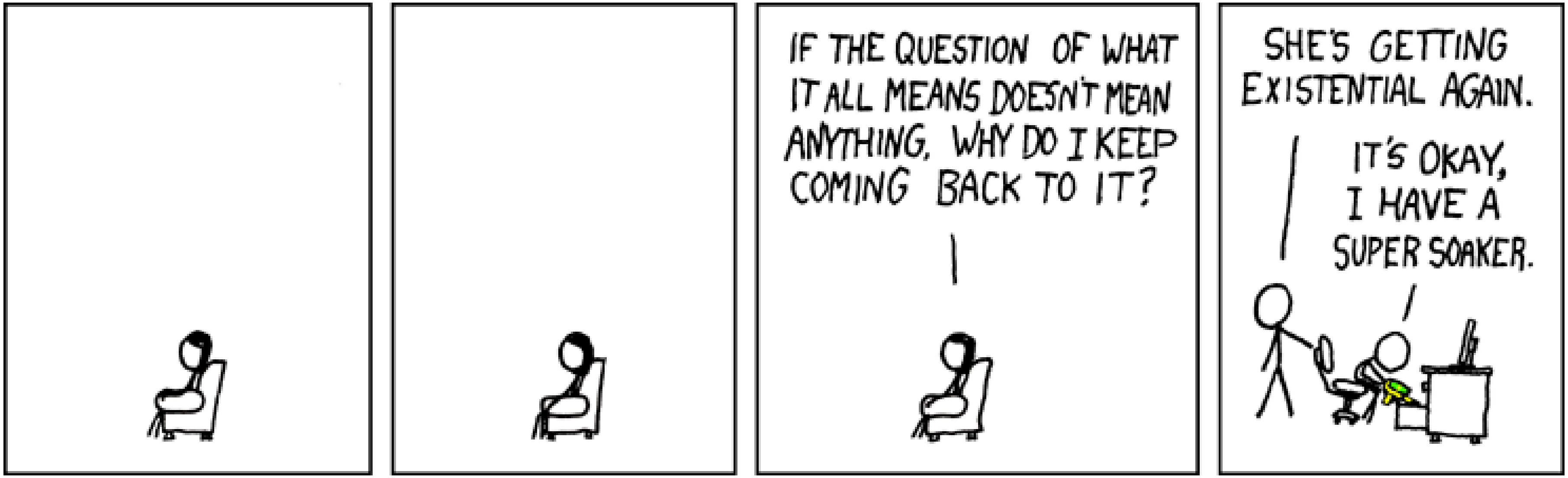
Berners-Lee (2009): *Semantic Web and Linked Data* http://www.w3.org/2009/Talks/0120-campus-party-tbl/

Knowledge Modeling & Ontology Engineering

- How to build ontologies?
  - Methodologies
- How to build *good* ontologies?
  - Best Practices
  - Design Patterns
  - Anti Patterns
  - Top Level Ontologies

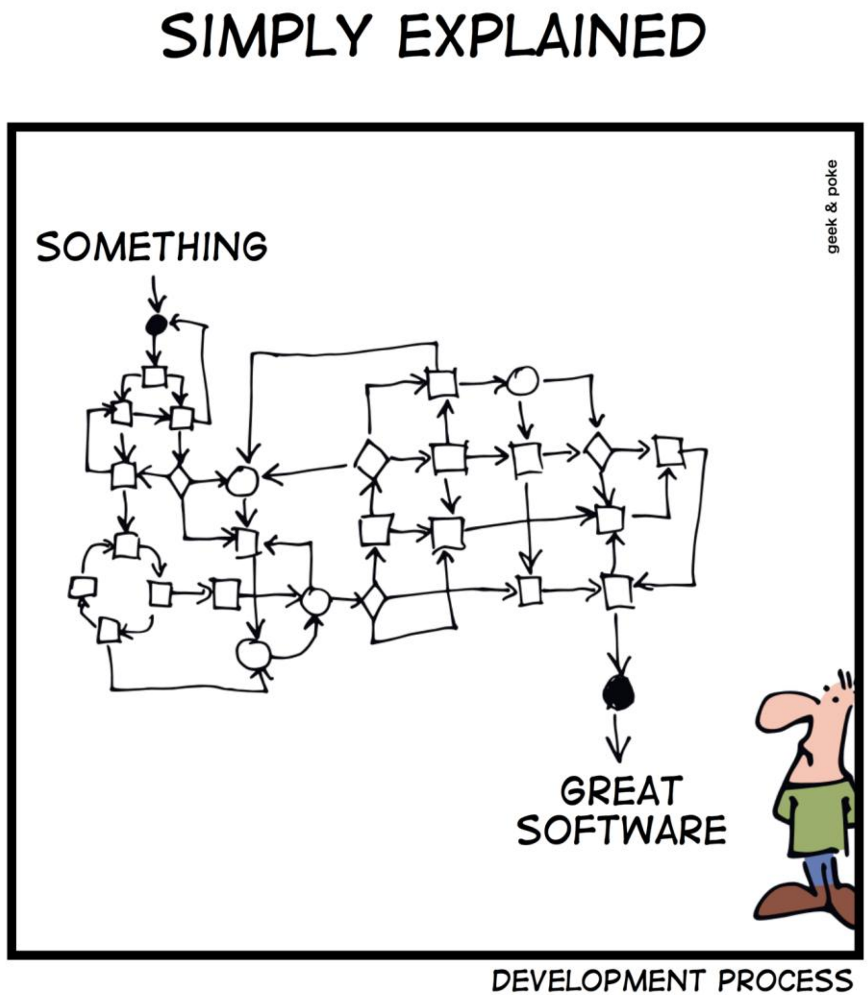
Warning

- Today's lecture contains a massive amount of philosophy (for computer scientists)



Methodologies

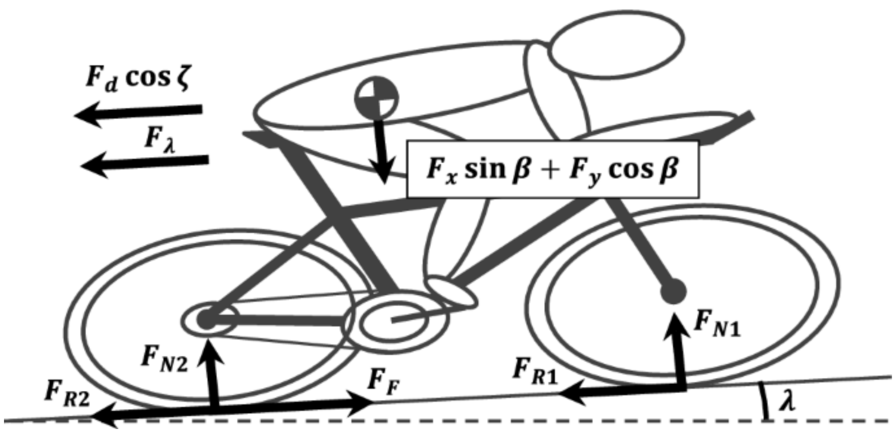
- Known, e.g., from Software Engineering

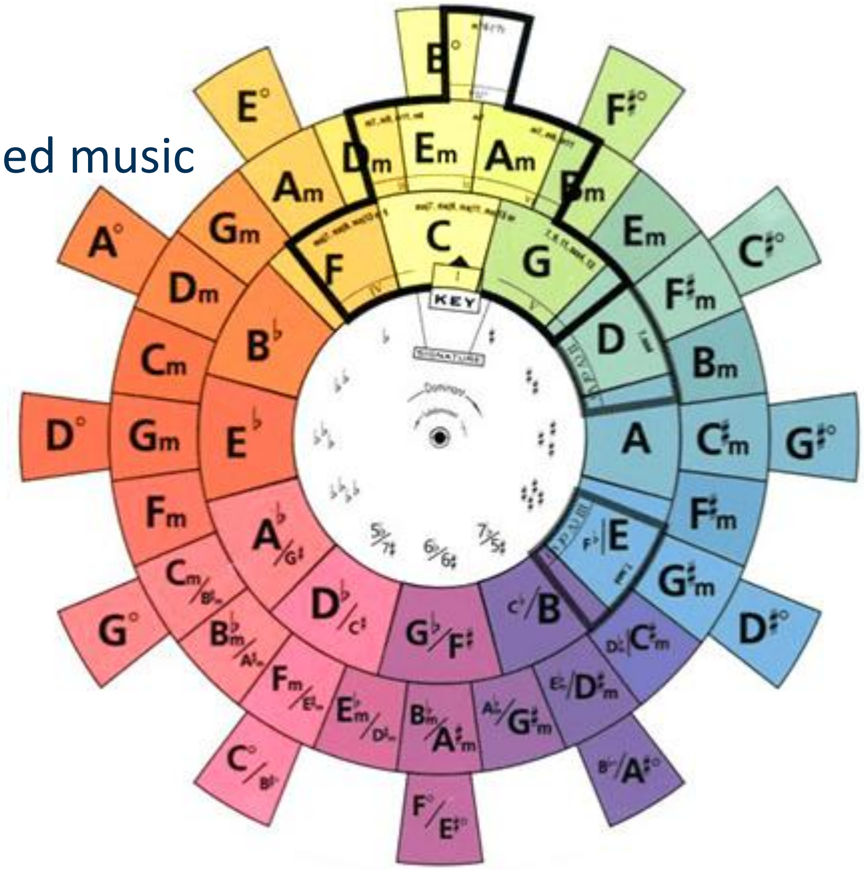


http://geekandpoke.typepad.com/geekandpoke/2012/01/simply-explained-dp.html

The SECI Model

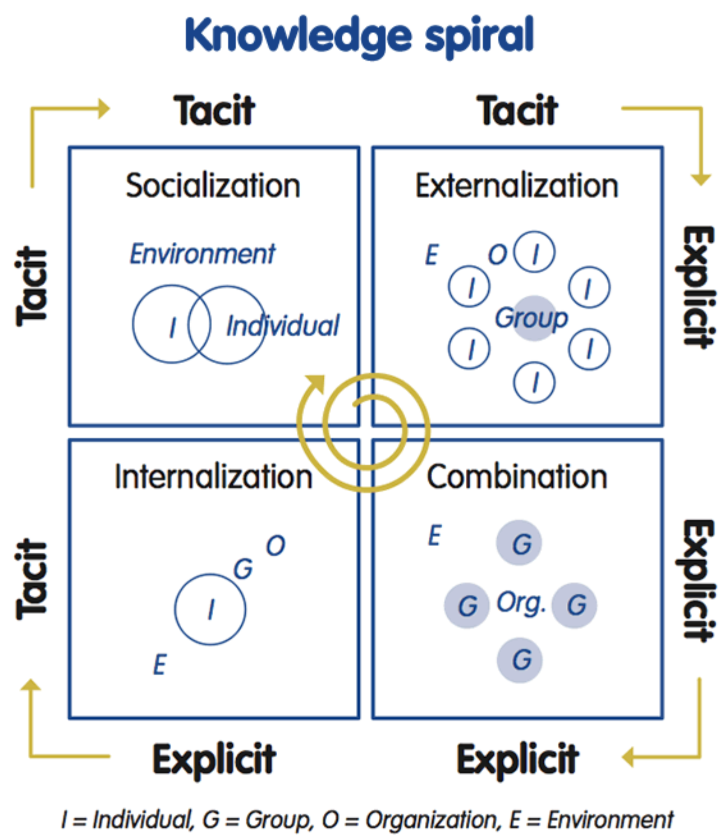
- Tacit Knowledge
  - intuitive, hard to formalize
  - e.g., riding a bike, playing improvised music
- Explicit knowledge
  - formalized
  - e.g., kinematics, music theory





The SECI Model

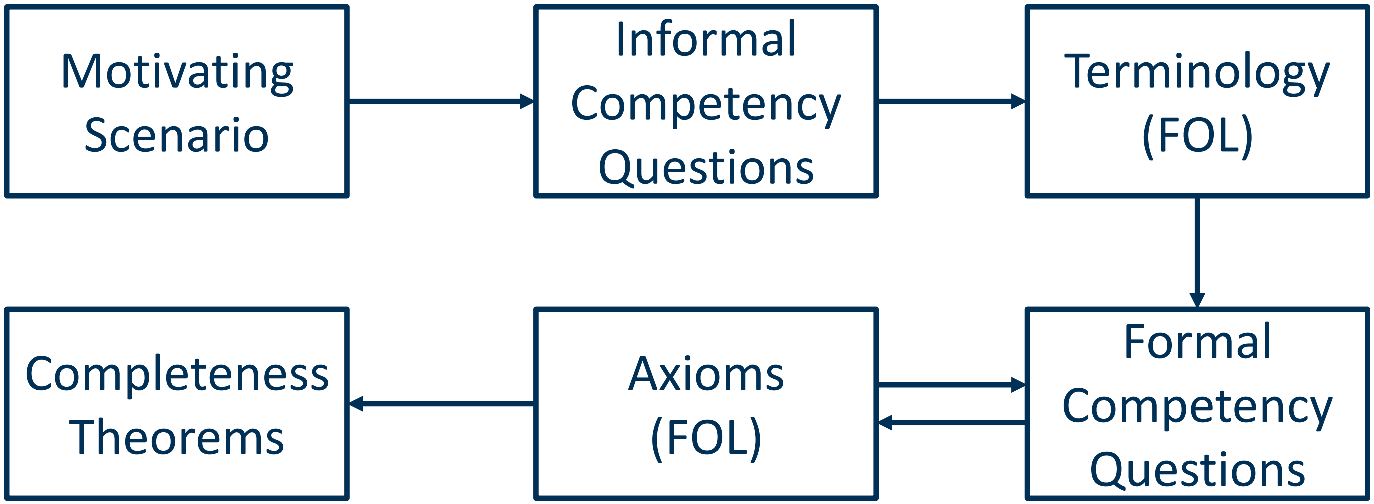
- Introduced by Nonaka & Takeuchi, 1990s
  - Tacit knowledge is created from explicit knowledge and vice versa
  - Knowledge creation is usually a cooperative process



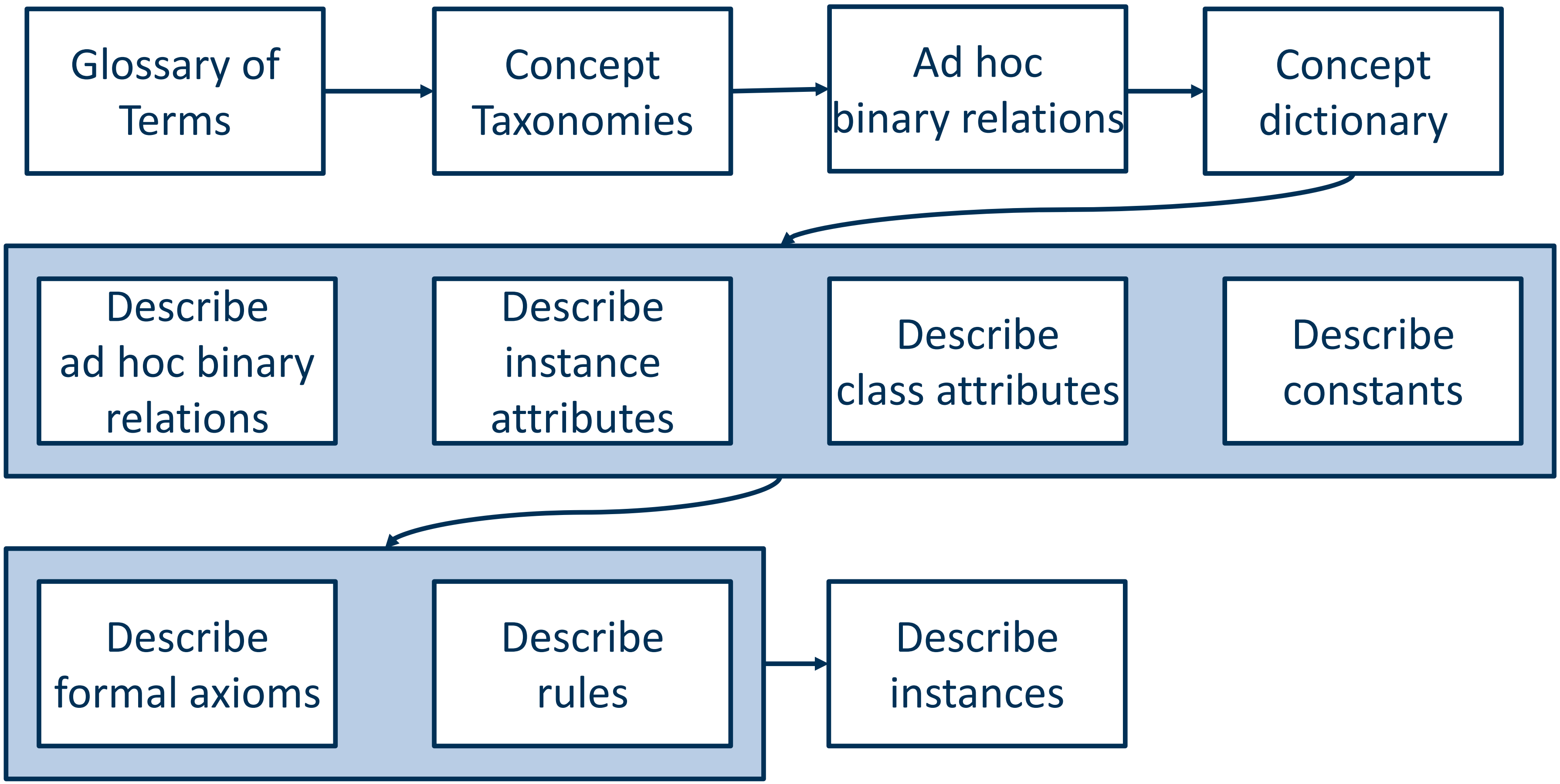
Bron: Nonaka/Peltokorpi (2006), Knowledge-based view of radical innovation: Toyota Prius Case

Grüninger & Fox's Methodology (1995)

- Informal competency questions: natural language
- Formal competency questions: e.g., SPARQL queries
  - with expected results
  - think: a built-in unit test



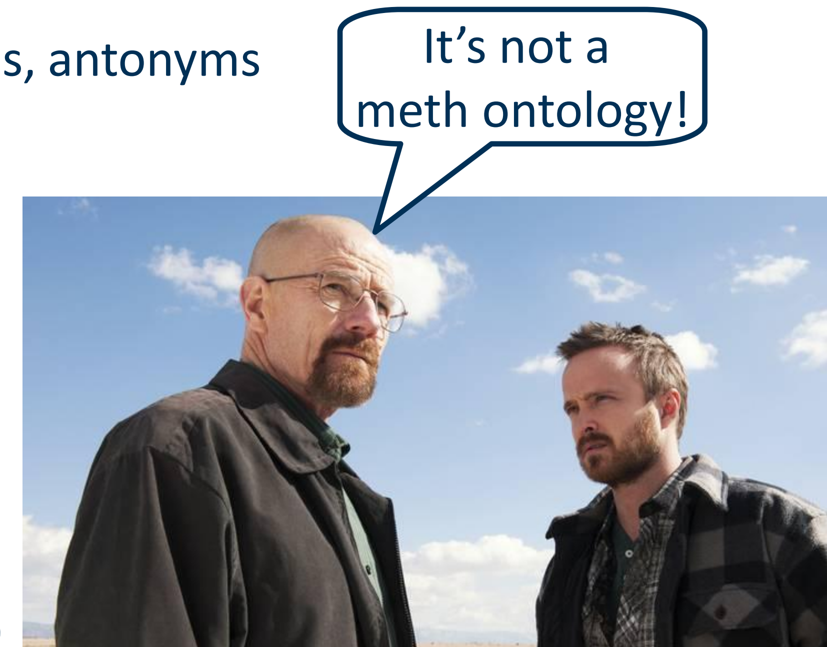
Methontology
(Fernández et al., 1997)



Gómez-Pérez et al. (2004): Ontological Engineering

Methontology (Fernández et al., 1997)

- Step by step from less to more formal ontologies
- Stepping back is allowed
- Documentation is produced along the way
- Glossary
  - Terms, descriptions, synonyms, antonyms
- Taxonomy
  - Sub class relations
- Ad hoc binary relations
  - a.k.a. ObjectProperties
- Concept dictionary
  - contains: terms, descriptions, relations, instances (optional)



Methontology (Fernández et al., 1997)

- Concept dictionary (example)

Concept name	Class attributes	Instance attributes	Relations
AA7462	--	--	same Flight as
American Airlines Flight	company Name	--	--
British Airways Flight	company Name	--	--
Five-star Hotel	number of Stars	--	--
Flight	--	--	same Flight as
Location	--	name size	is Arrival Place of is Departure Place of
Lodging	--	price of Standard Room	placed in
Travel	--	arrival Date company Name departure Date return Fare single Fare	arrival Place departure Place
Travel Package	--	budget final Price name number of Days travel Restrictions	arrival Place departure Place accommodated in travels in
USA Location	--	--	--

Gómez-Pérez et al. (2004): Ontological Engineering

Building *Good* Ontologies

- Real example SNOMED (a medical ontology)

```
:Finger :partOf :Hand .
:Hand :partOf :Arm .
:partOf a owl:TransitiveProperty .
:Surgery rdfs:subClassOf :Treatment .
:onBodyPart rdfs:domain :Treatment .
:onBodyPart owl:propertyChain (:onBodyPart,:partOf).
```

- This allows for inferences such as
  - An operation of the finger is also an operation of the hand (and an operation of the arm).
- So far, so good...

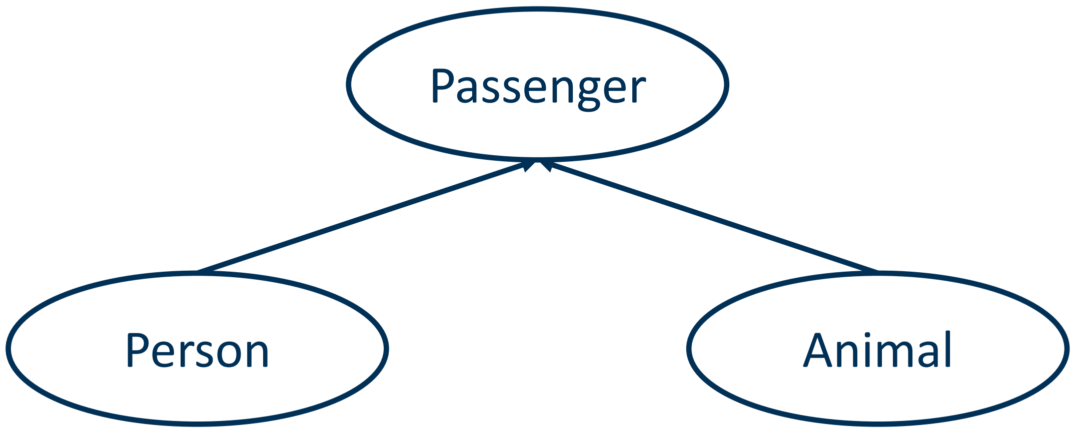
```
:Amputation rdfs:subClassOf :Surgery .
```

OntoClean

- A collection of analysis methods and tests
  - Does my class hierarchy make sense?
  - Tests: Rigidity, Identity, Unity (discussed in the following slides)
- Developed ~2000-2004 by Nicola Guarino and Chris Welty
  - Based on philosophical foundations

Rigidity

- Consider the following task:
  - *Build an ontology for public transport*
  - *"Passengers can be people and animals."*



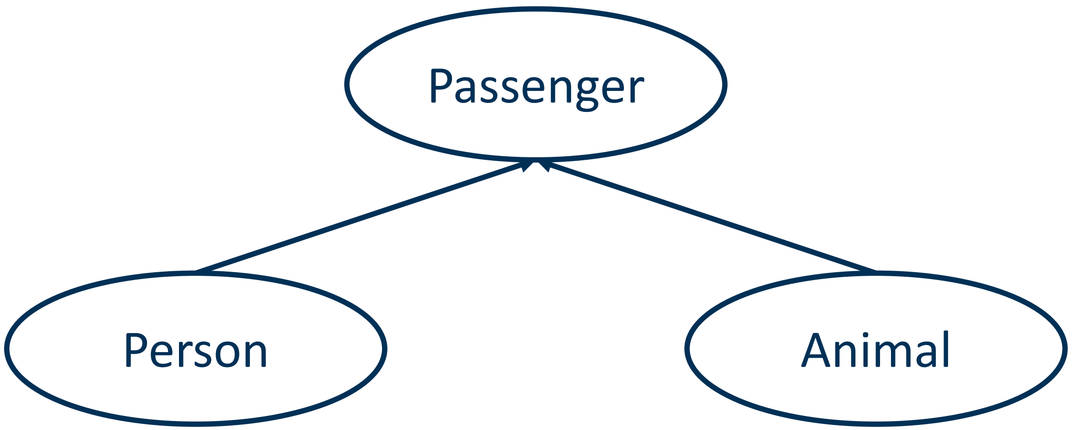
- How do you like this solution?

Rigidity

- OntoClean distinguishes *rigid* and *non-rigid* classes
  - If an entity belongs to a rigid class, this holds once and for all
    - i.e.: if the entity does not belong to that class anymore, it ceases to exist
  - This does not hold for non-rigid classes
- Examples for rigid classes
  - Person, mountain, company
- Examples for non-rigid classes
  - Student, stock company, town
  - Caterpillar and butterfly

Rigidity in OntoClean

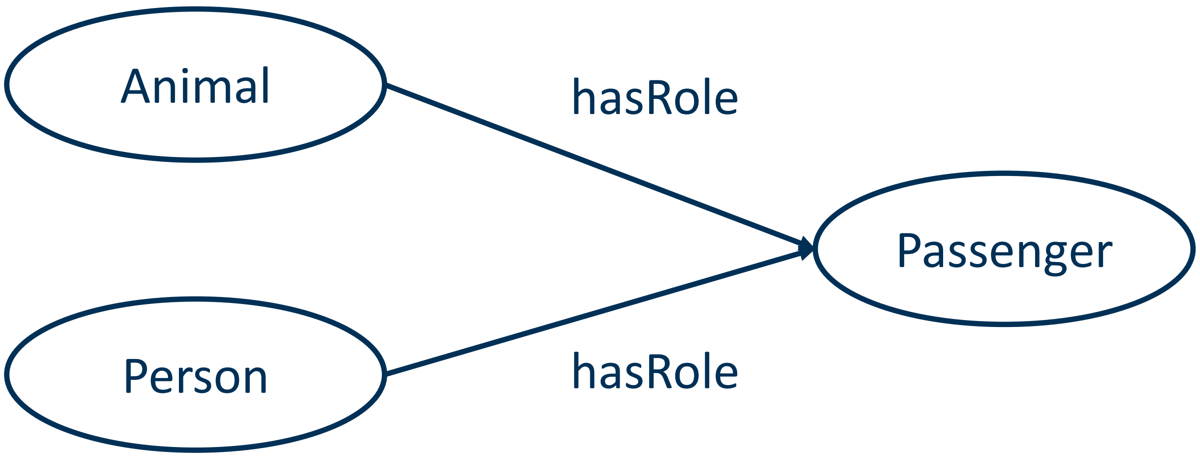
- OntoClean rule
  - Rigid classes must not be subclasses of non-rigid classes



- Assume that
  - `:peter a :Person .`
  - From that, we conclude that `:peter a :Passenger .`
  - This is probably unwanted

Rigidity in OntoClean

- Improved solution

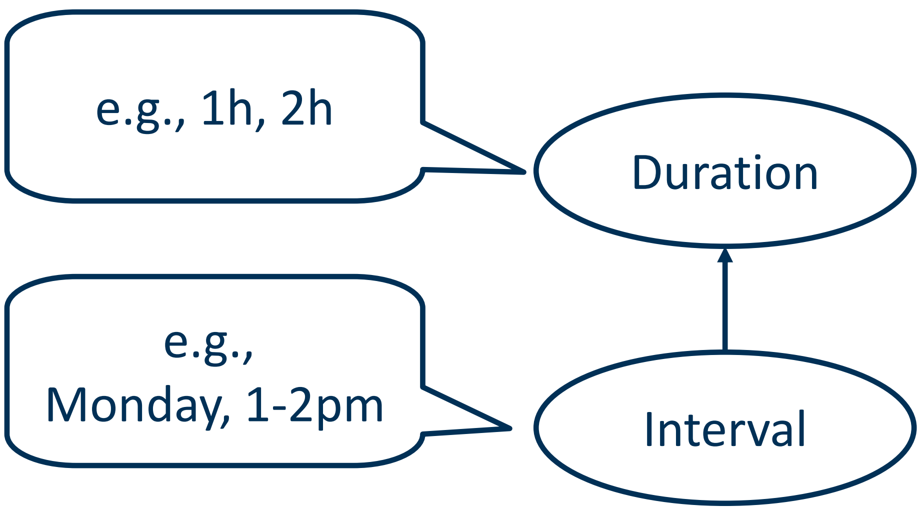


Rigidity in OntoClean

- Other typical rigidity problems
  - PhysicalObject > Animal
    - An entity may die and thus be no longer an animal
      - If we consider “living” as necessary for animals
    - The physical object (i.e., the body), however, still exists

Identity

- Consider the following task:
  - *Build an ontology for recording working times*
  - *"Time intervals are specific durations. A duration may be 1h, 2h, etc., a time interval may be Monday, 1-2pm, or Tuesday, 3-5pm."*



- How do you like this solution?

Identity

- Let us look at some instances
  - :1h a :Duration . :2h a :Duration . …
  - :Mo10-11 a :Interval . :Mo11-12 a :Interval . …

- Obviously, there are more instances of *Interval* than there are instances of *Duration*

- What does that mean?

Identity

- How do we know that two entities are the same
  - Some classes have criteria for identity
- Immatriculation number of students
- Tax number for citizens and companies
- Country codes
- …

Identity

- Since the subclass cannot be larger than the superclass, there must be instances that are the same

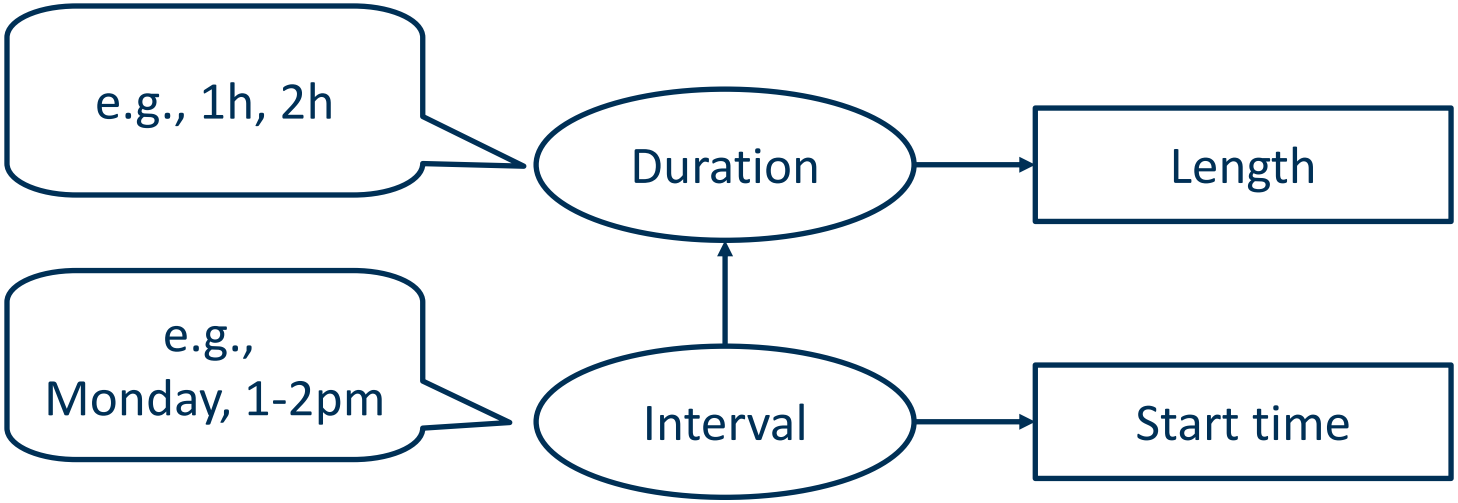
- Probably, we would expect a mapping such as
  - :Mo10-11 owl:sameAs :1h .
  - :Mo11-12 owl:sameAs :1h .

- From that, we conclude that
  - :Mo10-11 owl:sameAs :Mo11-12 .

- Do we really want that to hold?

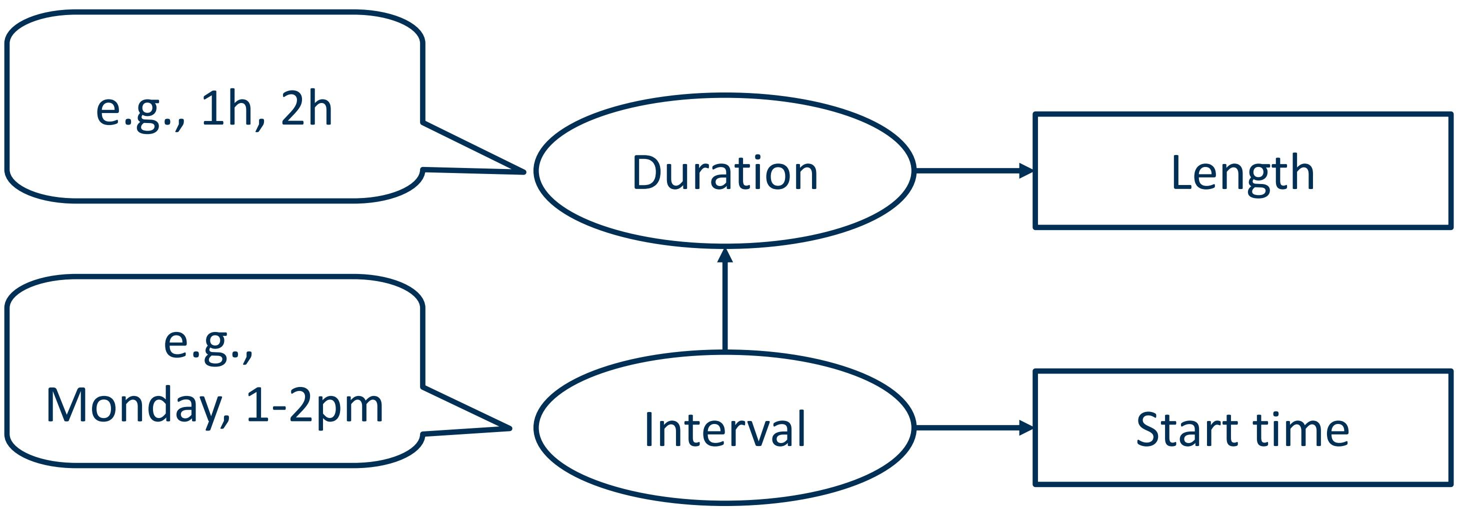
Identity

- We have to extend our ontology
- When are two durations the same?
  - If their length is the same
  - :1h owl:sameAs :60Min .



Identity

- We have to extend our ontology
- When are two intervals the same?
  - If they have the same length *and the same start time*
  - :Mo13-14 owl:sameAs :Mo1pm-2pm .

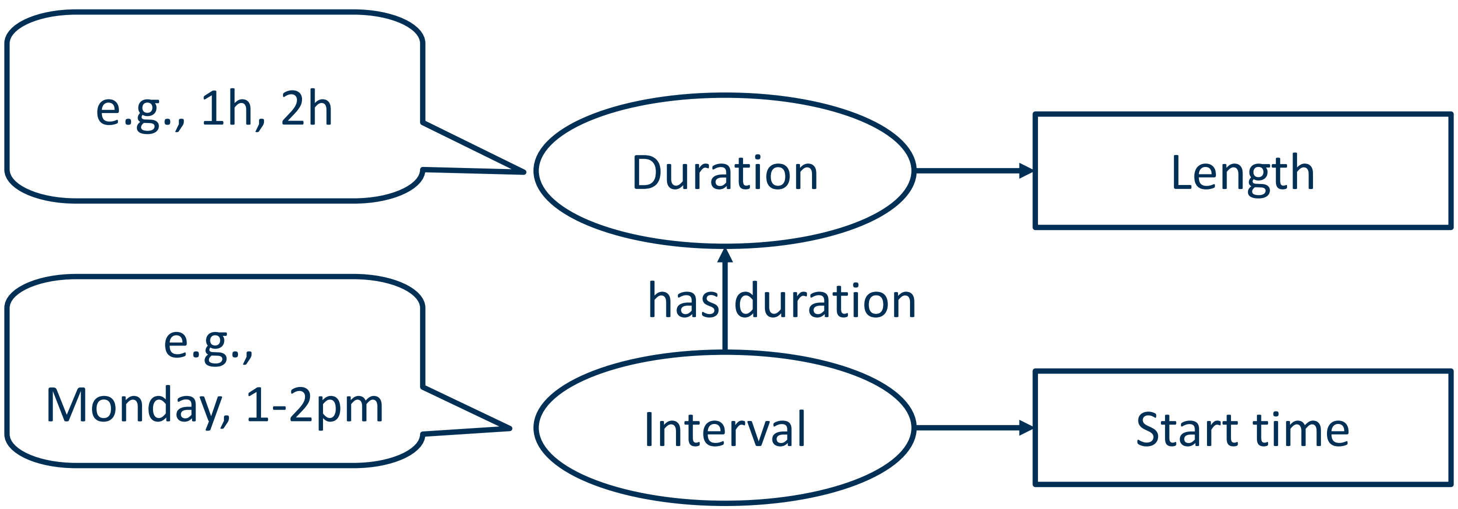


Identity in OntoClean

- Observation:
  - The identity criteria are of the two classes are different
- OntoClean rule:
  - If p is a subclass of q,
    then p must not have any identity criteria that q does not have

Identity in OntoClean

- Improved solution:
  - Replace subclass relation by another relation



Identity in OntoClean

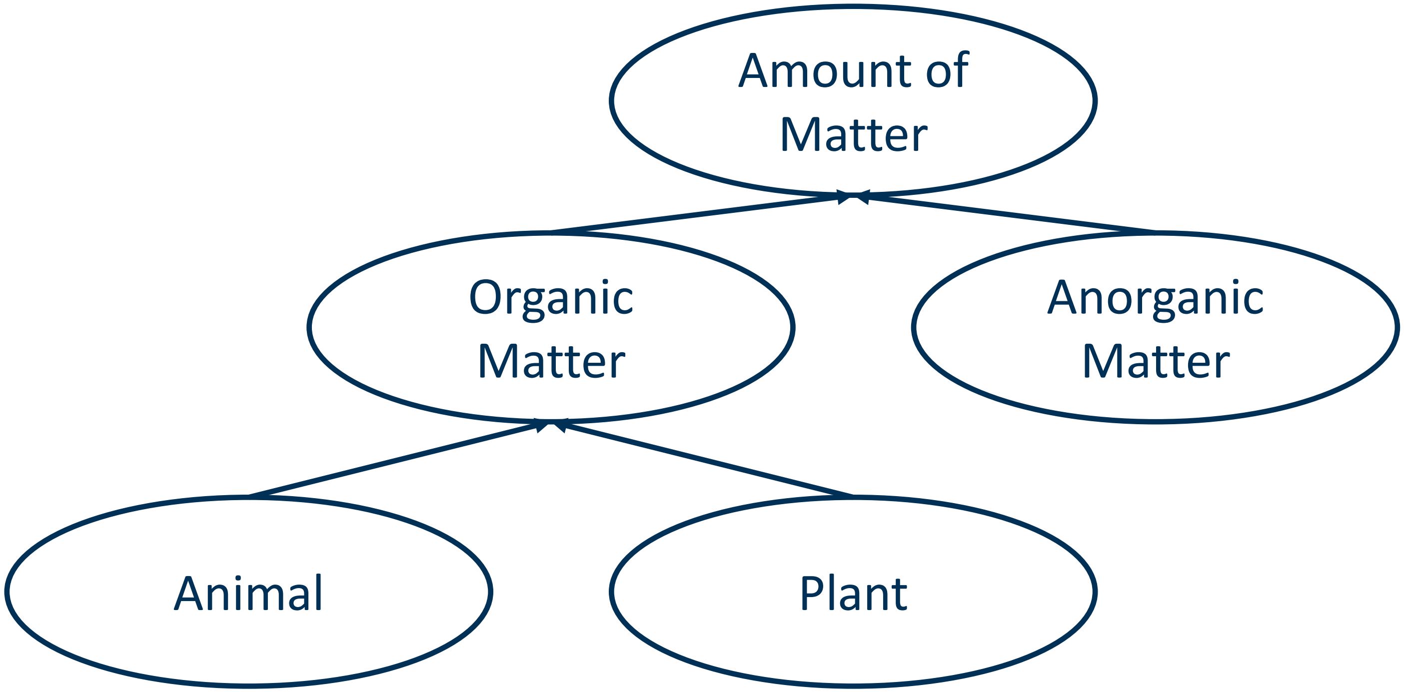
- Other typical problems
  - GeographicalObject > Country
    - Geographical objects and countries have different identity criteria
    - Geographical object: position/polygon
    - Country: government, constitution
    - OntoClean enforces a separation of the geographic and the social construct of a “country”
  - Book > Book edition
    - Book: Title, author
    - Book edition: ISBN, or title and author plus number of the edition
  - Book > Book copy
    - Book: ISBN
    - Book copy: inventory number

Unity

- For some classes, entities can be decomposed into instances of the same class
  - We call them “anti unity classes”
- Examples:
  - An amount of water into two amounts of water
  - A group into two sub groups
- Other classes only have “whole” instances → “unity classes”
  - e.g., people, cities
- For "whole" individuals, there is always a functional relation unambiguously relating a part to the whole
  - e.g., relating a body part to a person
  - e.g., relating a district to a city

Unity

- Assume that we defined



Unity

- Let us further assume that we defined*:
  - if we add two amounts of the same type of matter, the result is a larger amount of that type of matter

```
    :C rdfs:subClassof :AmountOfMatter
∧   :m1 a :C . :m2 a :C . :m3 :hasPart :m1, :m2 .
→   :m3 a :C .
```

*pretending this was possible in OWL, or using rules such as SWRL

Unity

- This leads to the following conclusion:

```
:fluffi a :Animal .
:schnuffi a :Animal .
:SetOfPetersPets :hasPart :fluffi, :schnuffi .

 → :SetOfPetersPets a :Animal .
```

- Do we want that?

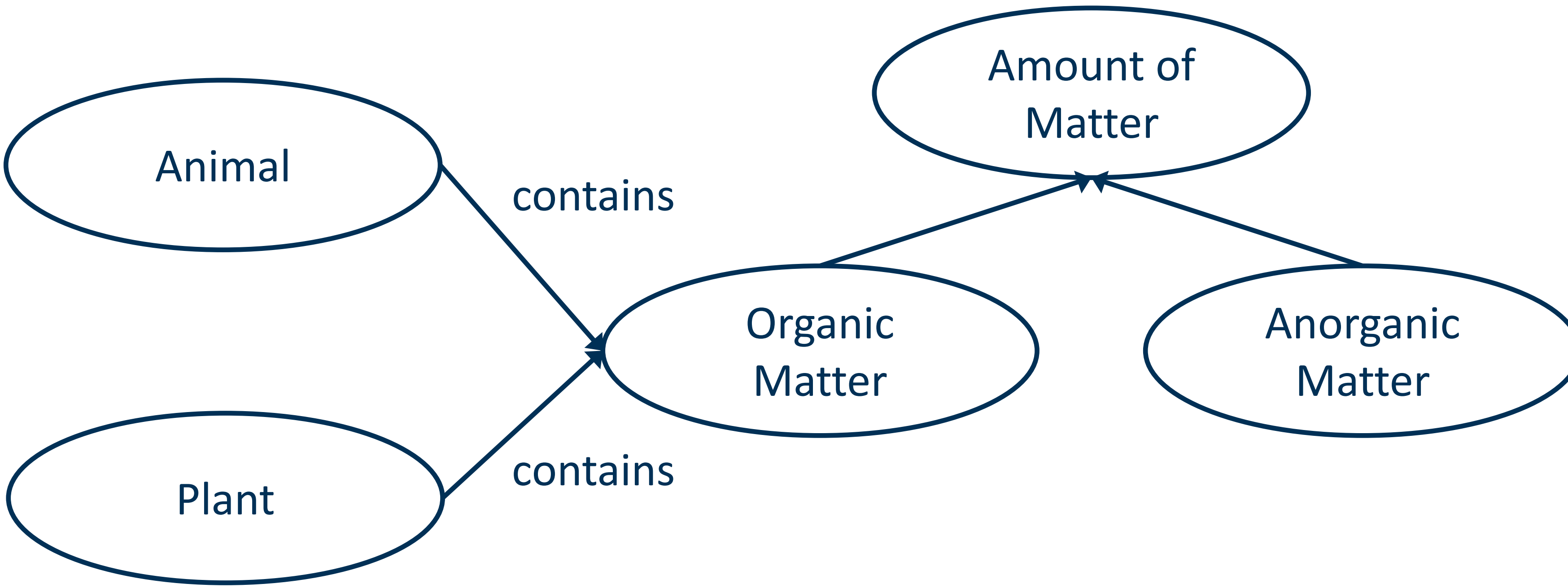
Unity in OntoClean

- OntoClean rule:
  - Unity classes may only have unity classes as their subclasses
  - Anti unity classes may only have anti unity classes as their subclasses

- In our example:
  - OrganicMatter is an anti unity class
  - Animal is a unity class

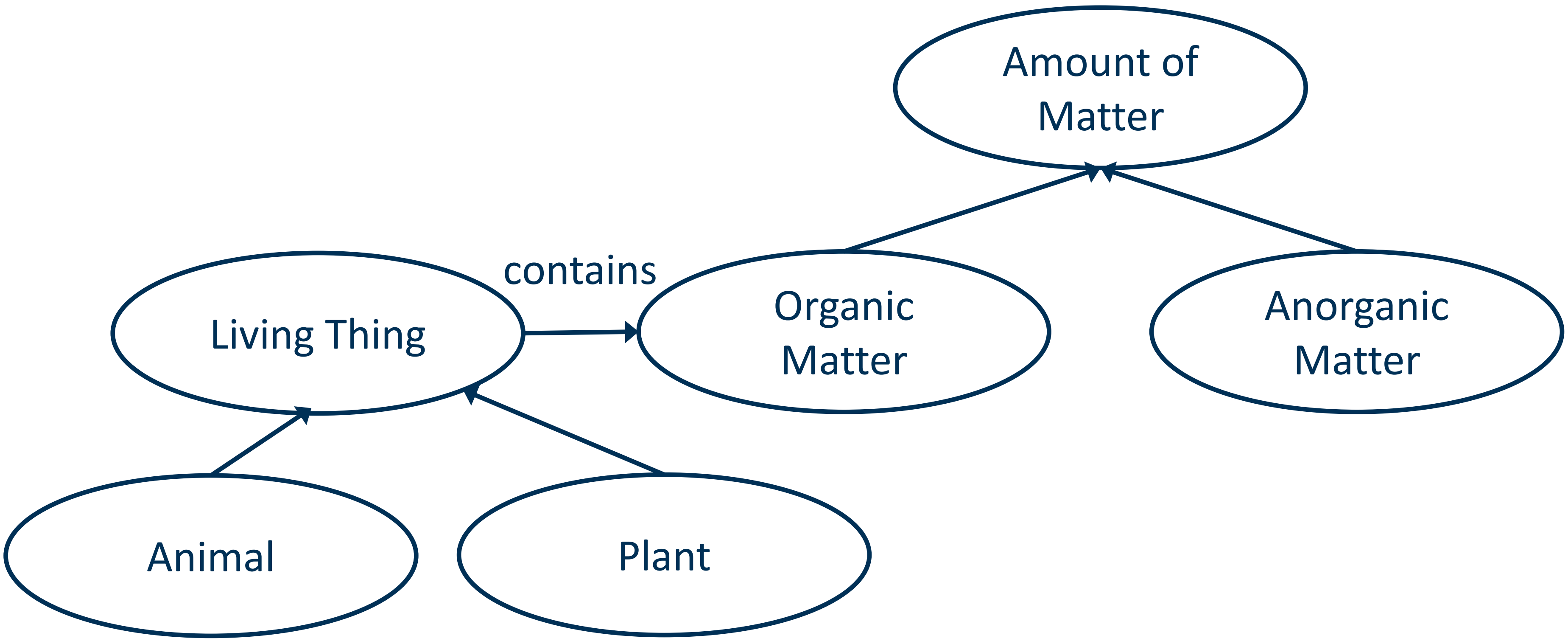
Unity in OntoClean

- Solution (again): replace subclass relation by a different relation



Unity in OntoClean

- Solution (again): replace subclass relation by a different relation



Summarizing OntoClean

- A number of tests that can be carried out on ontologies
  - Rigidity, Identity, Unity
  - Reveal possible mismodeling issues
  - Avoid nonsensical reasoning consequences

Ontology Design Patterns

- Origin of the term “design pattern”
  - Christopher Alexander (*1936)
  - Book "A Pattern Language" (1977)
- Architecture
  - Recurring problems
  - Standard solutions
    - With certain degrees of freedom
- Example
  - Problem: rain falls into the building
  - Solution: roof
    - Degrees of freedom: shed roof, saddle roof, hip roof...

Types of Ontology Design Patterns

- Presentation Patterns
  - e.g., naming conventions
- Logical Patterns
  - Domain independent
  - Always specific to a language (e.g., OWL DL)
- Content Patterns
  - Domain dependent
  - Language independent
- Transformation Patterns
  - e.g., how to transform an ontology from one language to the other

Presentation Patterns

- Typical ontology naming conventions
- Use CamelCase
  - CityInNorthernEurope
- Classes start with capital letters, always use singular nouns
  - City, Country
- Properties start with small letters, use a verb, allow unambiguous reading direction
  - isLocatedIn, isCapitalOf
- Instances start with a capital letter
  - Paris, France
- Provide labels for each class, property, and instance
- ...

Logical Patterns

- Example: ternary relation
- Statement to express: r(X,Y,Z)
- Pattern:

```
:R a owl:Class .
:hasR a owl:ObjectProperty .
:rComp1 a owl:ObjectProperty .
:rComp2 a owl:ObjectProperty .
:X :hasR [
  a :R ;
  :hasComp1 :Y;
  :hasComp2 :Z
].
```

Content Pattern

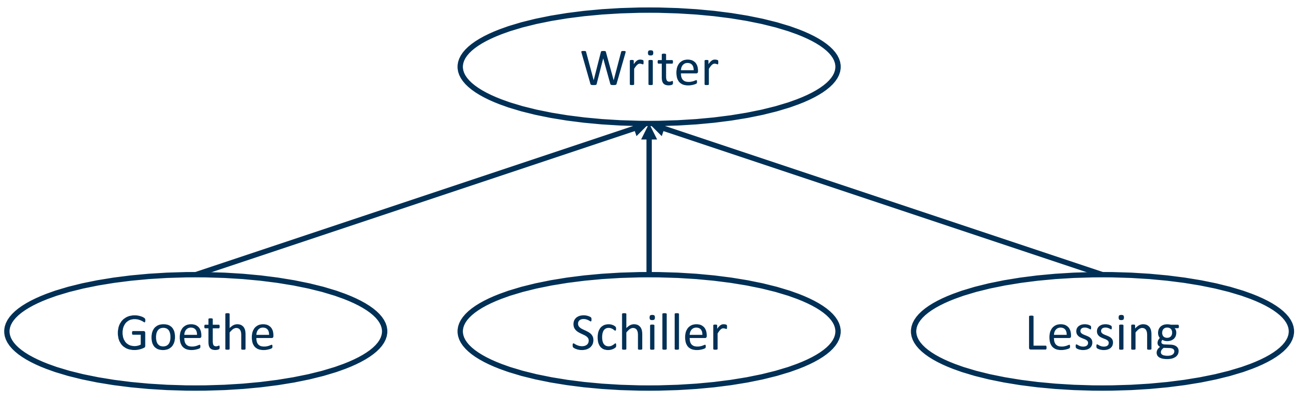
- Example: Roles taken at a time
  - e.g.: Angela Merkel was the chancellor of Germany from 2005 to 2021
- Competency Question:
  - Who had a certain role at a given time?
- Specializes
  - ternary relation

Anti-Patterns

- Things that should not be done
  - But are often done
  - ...and cause some problems
- Possible causes
  - Not thought about each and every consequence
  - Little/wrong understanding of RDF/OWL principles

Anti-Patterns: Rampant Classism

- Typical problem:
  - What should be an instance, what should be class?



Anti-Patterns: Rampant Classism

- This is an extreme case...

```
:Goethe rdfs:subClassOf :Writer .
:Faust rdfs:subClassOf :Drama .
:Goethe :authorOf :Faust .
```

- What can we conclude from that?

- Nothing with a DL reasoner,
  because this is not proper DL!

Anti-Patterns: Rampant Classism

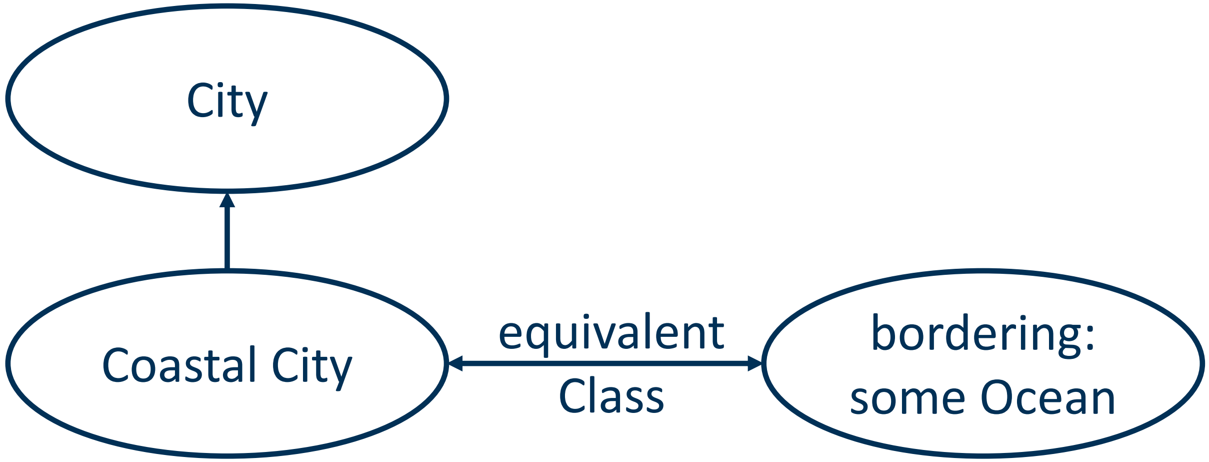
- How to distinguish classes and instances
- For every class, there must be (one or more) instance(s)
  - What should be instances of *Goethe*?
  - Are there any sentences like “X is a Goethe”?

- Sub class relations must make sense
  - Pattern: “Every X is a Y”
  - “Every Goethe is a Writer”?

Anti-Patterns: Exclusivity

- Given the following specification:
  - *Cities bordering an ocean are coastal cities.*

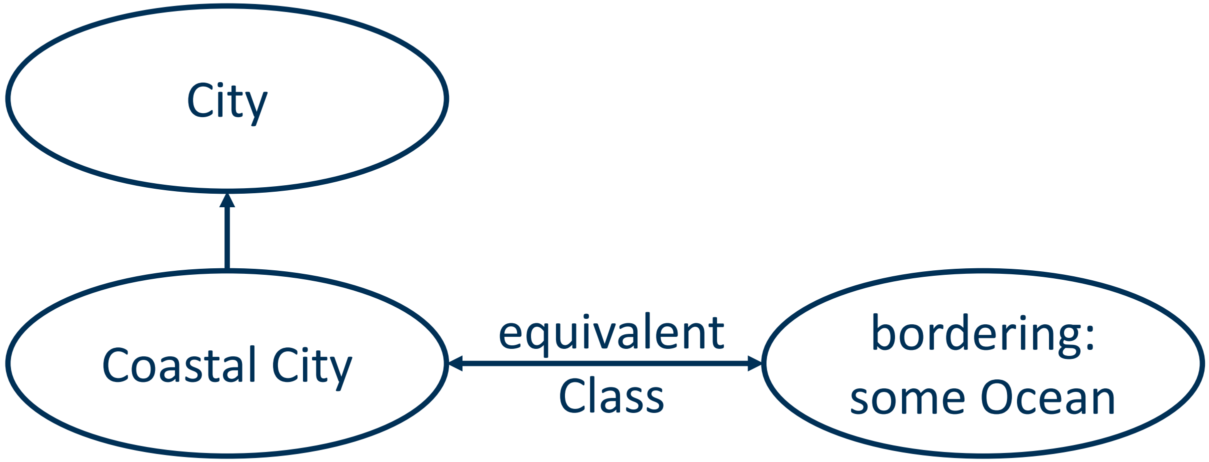
- Modeled in OWL, e.g.



Anti-Patterns: Exclusivity

- In OWL:

```
:CoastalCity rdfs:subClassOf :City ;
             owl:equivalentClass [
                 a owl:Restriction ;
                 owl:onProperty :bordering ;
                 owl:someValuesFrom :Ocean
             ] .
```



Anti-Patterns: Exclusivity

- Now with instances:

```
:Hamburg a :City .
:Hamburg :bordering :AtlanticOcean .
:AtlanticOcean a :Ocean .
```

→ `:Hamburg a :CoastalCity .`

- So far, so good.

```
:Germany a :Country .
:Germany :bordering :AtlanticOcean .
:AtlanticOcean a :Ocean .
```

→ `:Germany a :CoastalCity .`

→ `:Germany a :City .`

Anti-Patterns: Exclusivity

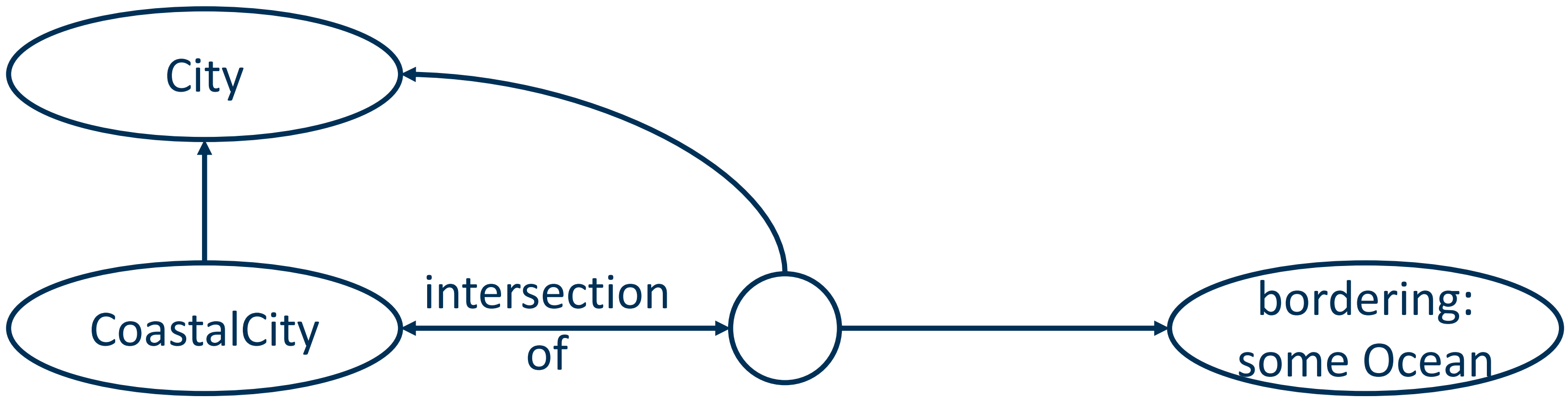
- What is happening here?
  - Ontology was built *exclusively* for a domain
  - e.g., cities
  - Breaks if used in another context (here: countries)
- Recap: Semantic Web Principles
  - AAA (Anybody can say Anything about Anything)
  - i.e., statements should work in different contexts

- Another example:
  - Every person is married to at most one other person

Anti-Patterns: Exclusivity

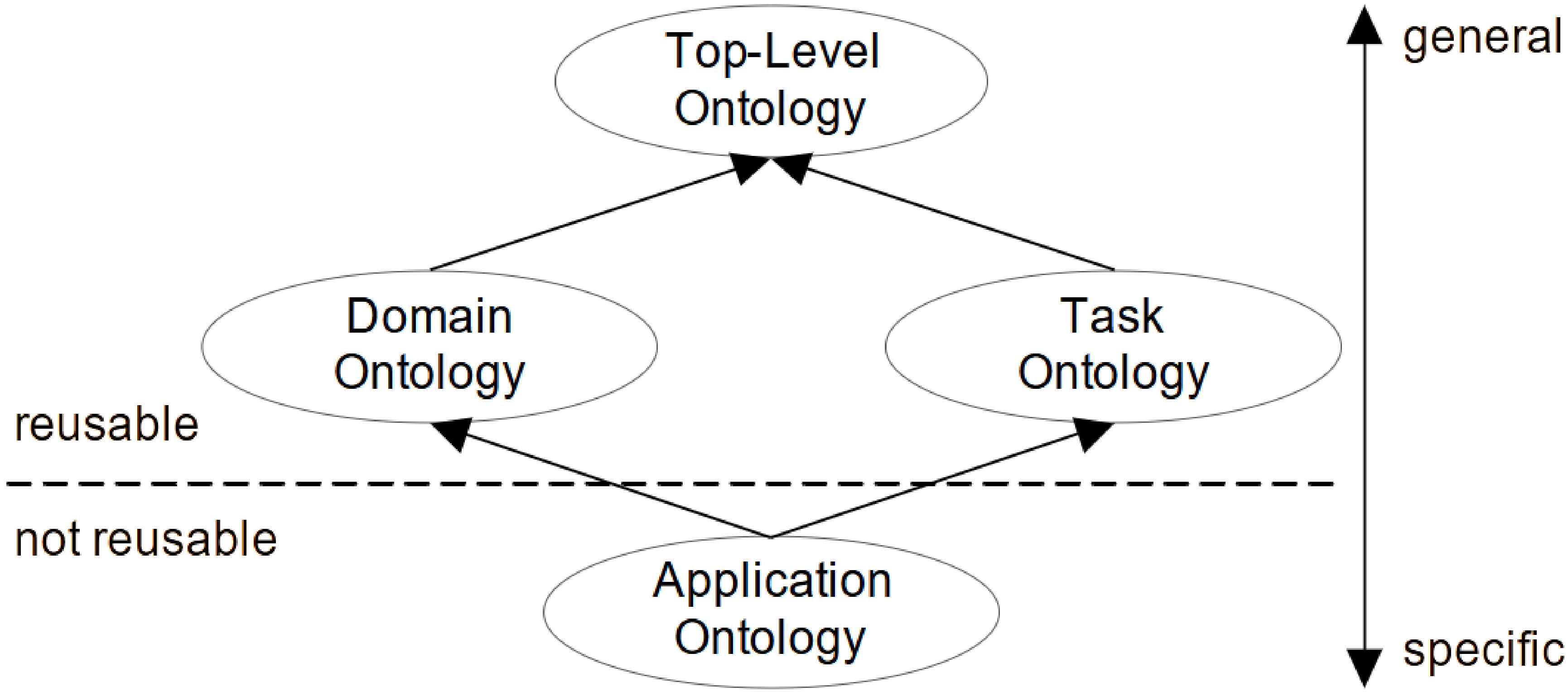
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Possible Solution:




```
:CoastalCity owl:intersectionOf ( :City [
       a owl:Restriction ;
       owl:onProperty :bordering ;
       owl:someValuesFrom :Ocean
     ]
) .
```

Classification of Ontologies



Guarino: Formal Ontology and Information Systems (1998)

Top Level Ontologies

- Top Level Ontologies
  - Domain independent
  - Task independent
  - Very general
- Goal
  - Reuse
  - Semantic clarity
  - Modeling guidance (i.e., avoid bad modeling)
  - Interoperability

History

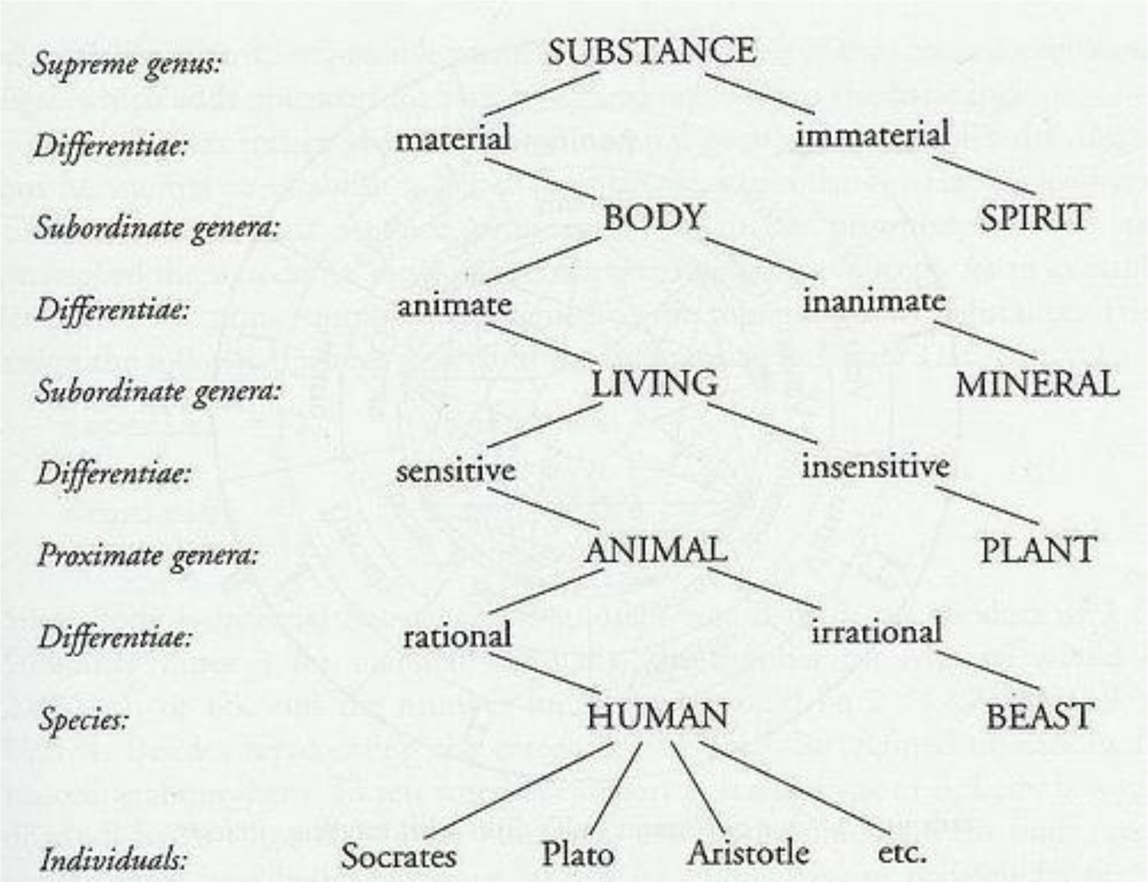
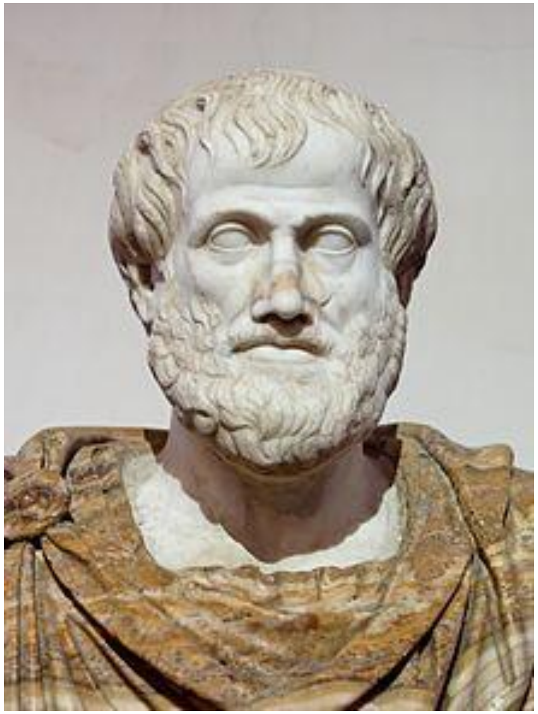


FIGURE 1.1 Tree of Porphyry, translated from a version by Peter of Spain (1239)

Porphyry, Greek philosopher, ca. 234-305

History

- One of the oldest top level ontologies
  - Aristotle (384-322)
- Four basic categories of existence



Aristotle's Ontological Square

- Example: „white coffee mugs“

	not substantial	substantial
universal	Category II the color “white”	Category III the category of white coffee mugs
particular	Category I the white color of a particular coffee mug	Category IV a particular white coffee mug

Basic Categories for Top Level Ontologies

- Abstract vs. concrete entities
- Abstract entities do neither have a temporal nor a spatial dimension
  - Numbers
  - Units of measure
- Concrete entities do at least have a temporal dimension, i.e., a time span at which they exist (spatial is optional)
  - Things (books, tables, …)
  - Events (lectures, tournaments, ...)

Basic Categories for Top Level Ontologies

- 3D vs. 4D view
- 3D view
  - Things extend in space
  - At every point in time, they are completely present
- 4D view
  - Things extend in time and space
  - At a given point in time, they can also be partially present
- Actual vs. possible entities
  - Actualism: only existing entities are included in an ontology
  - Possibilism: all possible entities are included in an ontology

Basic Categories for Top Level Ontologies

- Co-location
  - Can multiple entities exist in the same place?
- This should be easy...
  - 3D view: no
  - 4D view: yes, but not at the same time
- ...but it is not that trivial
  - Example: a statue and the amount of clay from which it was made
- Do statues even exist?
  - Or is there only clay in the shape of a statue?
  - ...and if both exist, should they belong to the same category?
  - Another example: a hole in a piece of Swiss cheese
- Do holes even exist?
  - Or are there only perforated objects?

John Sowa's Top Level Ontology

- An “older” top level ontology (1990s)
- Three distinctions form twelve basic categories
  - Physical vs. Abstract
    - Things that exist in time (and potentially in space)
    - Things that do not
  - Continuant vs. Occurent
    - Things that exist as a whole at each point in time
    - Things that partially exist at each point in time
  - Independent vs. Relative vs. Mediating
    - Things that can exist on their own
    - Things that require other things to exist
    - “Third” things that relate two others

John Sowa's Top Level Ontology

- These three distinctions create twelve basic classes of objects
  - All of them are disjoint

	Physical		Abstract	
	Continuant	Occurent	Continuant	Occurent
Independent	Object	Process	Schema	Script
Relative	Juncture	Participation	Description	History
Mediating	Structure	Situation	Reason	Purpose

John F. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Foundations (1999)

John Sowa's Top Level Ontology

- Which categories do those entities belong to?
  - The building B6 23-25
  - Today's Knowledge Graphs lecture
  - The semester break between HWS 2025 and FSS 2026
  - Your motivation to be here today

	Physical		Abstract	
	Continuant	Occurent	Continuant	Occurent
Independent	Object	Process	Schema	Script
Relative	Juncture	Participation	Description	History
Mediating	Structure	Situation	Reason	Purpose

John F. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Foundations (1999)

DOLCE

- *Descriptive Ontology for Linguistic and Cognitive Engineering*
- One of the most well known top level ontologies
  - Originally developed in the EU WonderWeb project (2002-2004)
  - Strong philosophical foundation
- Modular design
  - Basic ontologies: 37 classes, 70 relations
  - All modules: ~120 classes, ~300 relations

Basic Distinctions in DOLCE

UNIVERSITY OF MANNHEIM
Data and Web Science Group

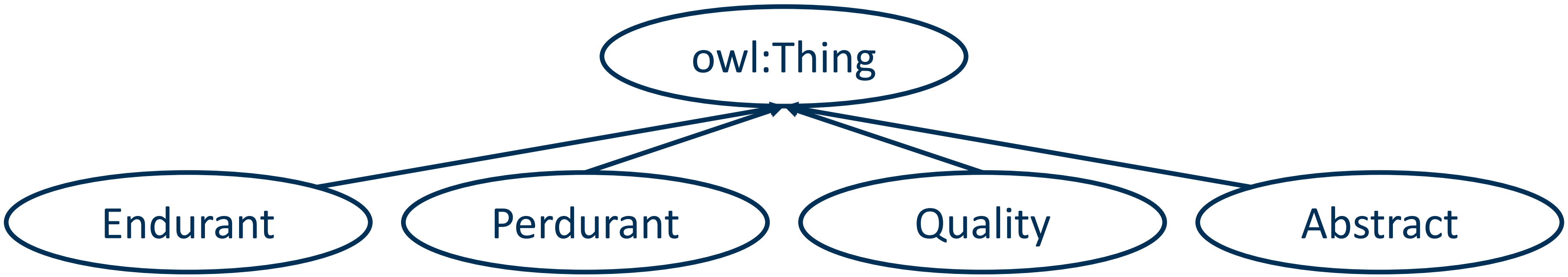
- Particulars, universals, and quantities
- Universals (think: categories): can have instances
  - “City”, “University”
- Particulars (think: individuals): cannot have instances
  - "Mannheim", "Mannheim University"
- Qualities: describe an instance
  - e.g., color of a book, height of a person
  - Are neither particulars nor universals
  - Cannot exist without an instance

DOLCE: Basic Assumptions

- A top level ontology of *particulars*
  - For both actual and possible entities (possibilistic view)
- 4D
  - Some entities may have a temporal dimension
- Co-location
  - Is allowed
  - restriction: not two entities of the same kind at the same spatial and temporal location
    - Not: two statues
    - But: a statue and an amount of clay

Top Hierarchy of DOLCE

- Four pairwise disjoint classes



Masolo et al. (2003): Ontology Library (final). WonderWeb Deliverable D18.

Endurants vs. Perdurants

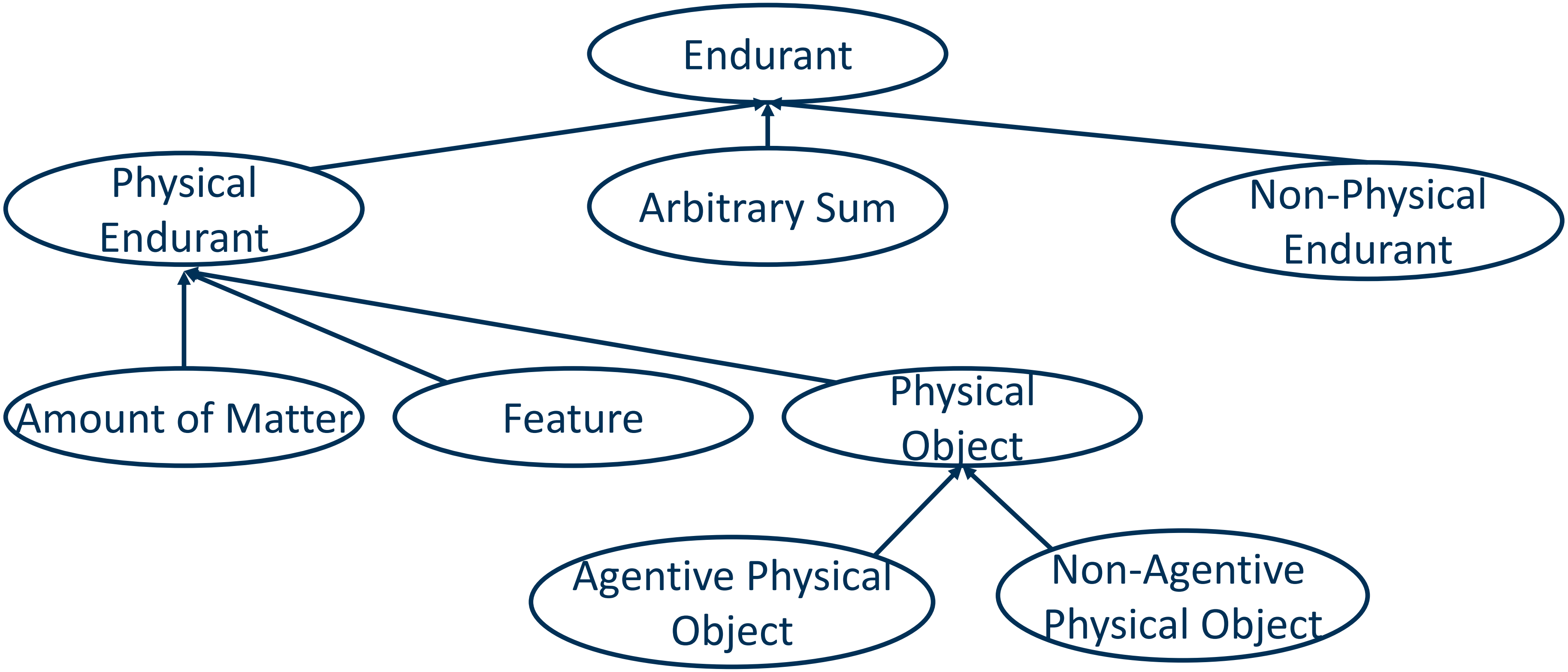
- Endurants exist in time
  - Think: things like people, books, ...
    - May also be non-physical: organizations, pieces of information
  - Are always fully present at each point in time during their existence
- Perdurants "happen" in time
  - Think: events and processes
  - Only exist partially at each point in time during their existence
    - i.e., previous and future parts of the perdurant
      may not (yet|anymore) exist at a given point in time
- Qualities are attached to endurants and perdurants
- Abstracts: numbers, units of measure, etc.

Endurants vs. Perdurants

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Endurants take part in perdurants
  - Actively (Reader and reading)
  - Passively (Book and reading)
  - DOLCE defines various types of participation
- Endurants only consist of endurants, perdurants only consist of perdurants
  - Books consist of pages, cover, ...
  - Reading consists of perceiving, turning pages, ...

Endurants in DOLCE (1)

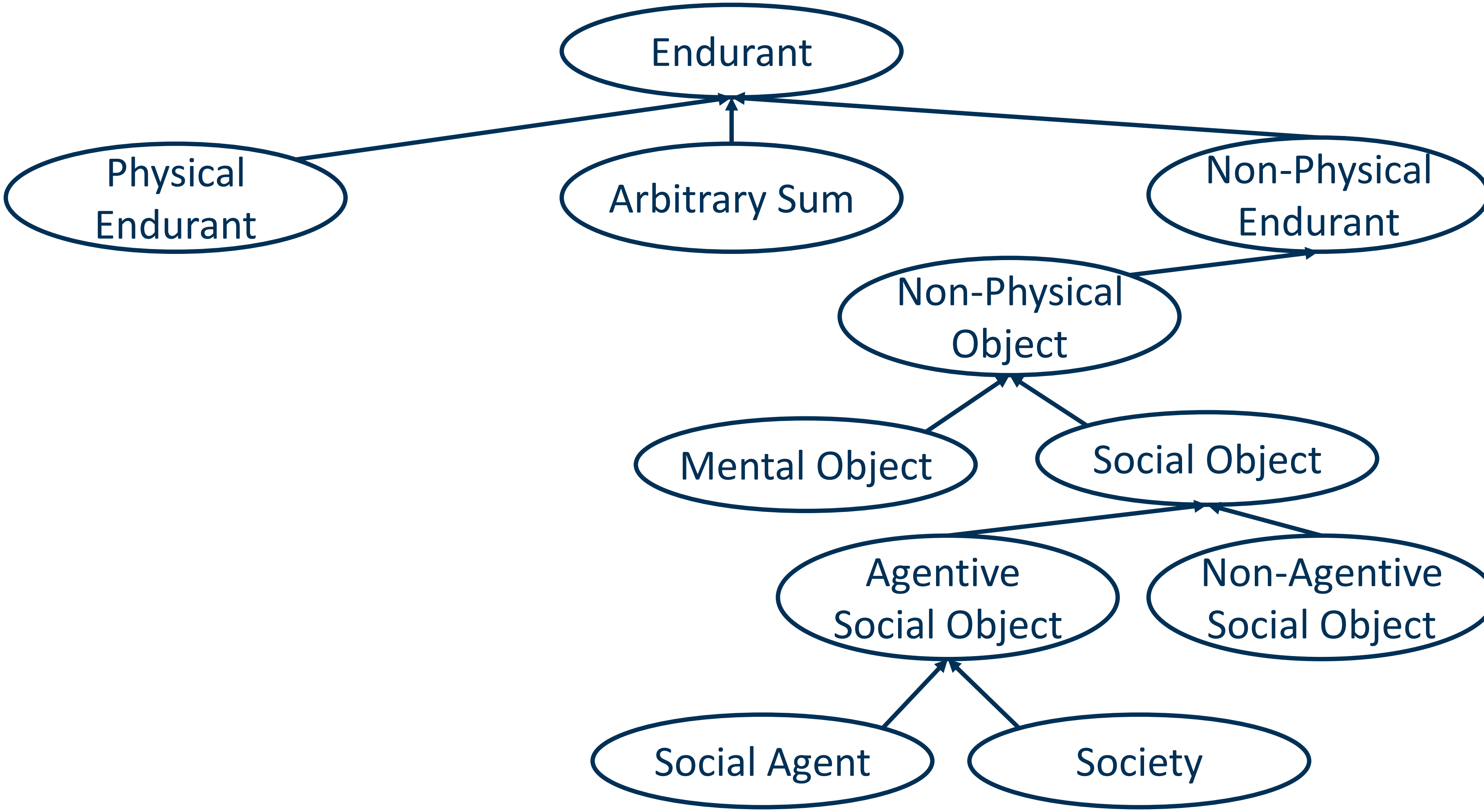


Masolo et al. (2003): Ontology Library (final). WonderWeb Deliverable D18.

Distinguishing Endurants

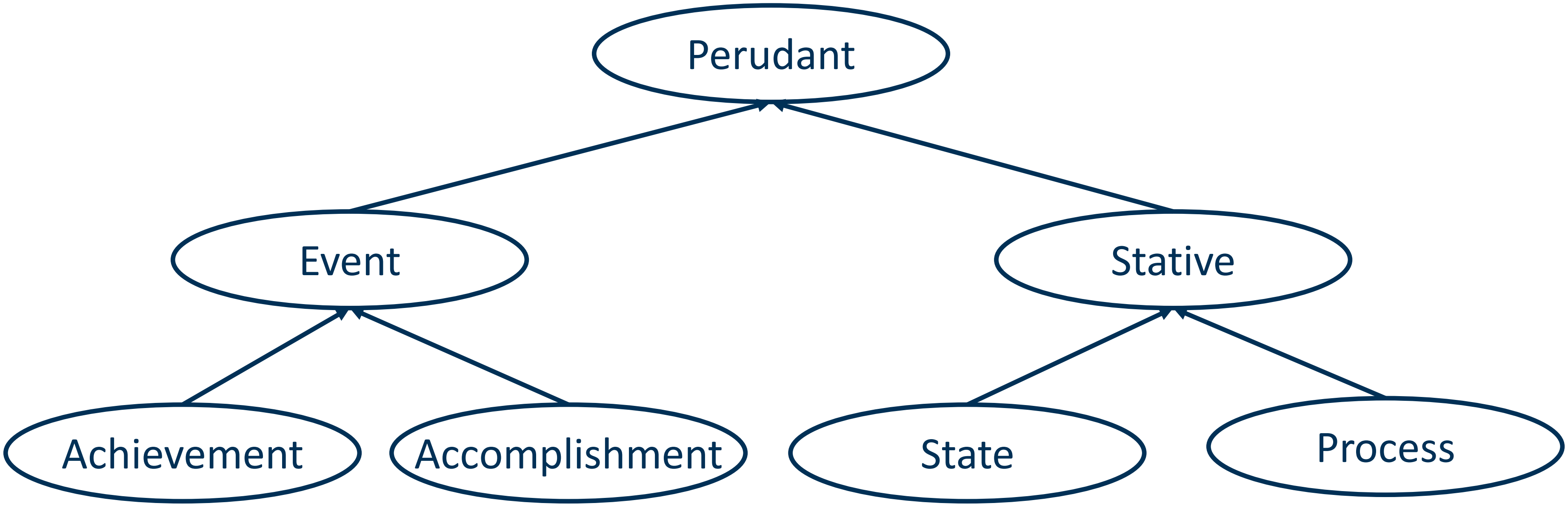
- Amount of Matter vs. Phyiscal Object
  - Amount of Matter is *“mereologically invariant”*
  - i.e., a part of an AoM is still an AoM
    - A part of “some water” is still “some water”
    - But a part of a cup is (likely) not a cup
  - cf. unity/anti unity in OntoClean
- Features
  - Cannot exist without a physical endurant
  - e.g., holes, fringes

Endurants in DOLCE (2)



Masolo et al. (2003): Ontology Library (final). WonderWeb Deliverable D18.

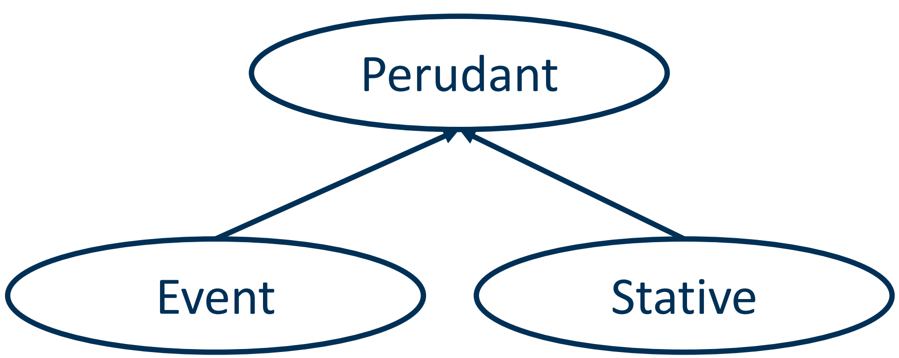
Perdurants in DOLCE



Masolo et al. (2003): Ontology Library (final). WonderWeb Deliverable D18.

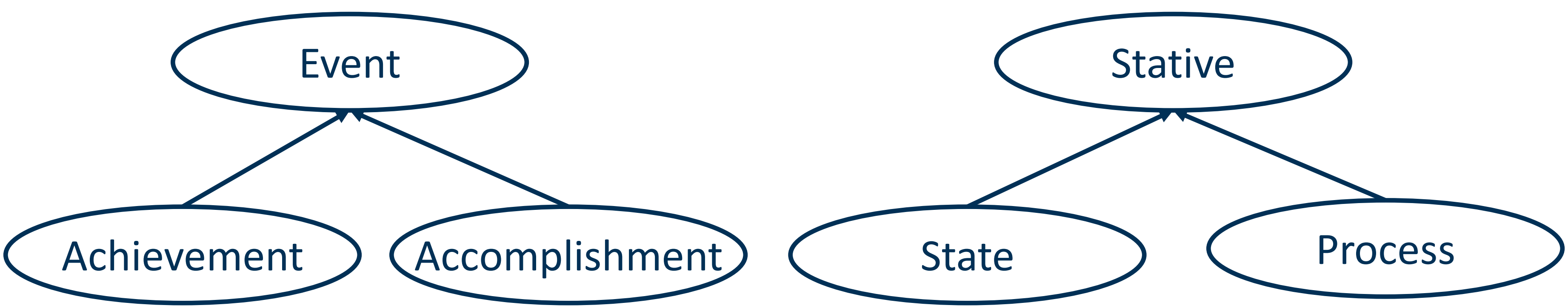
Distinguishing Perdurants

- Events vs. Statives
  – The sum of two consecutive statives is a (longer) stative
    - The sum of two times “sitting around” is “sitting around for a longer time”
    - But: the sum of two times “flying to the moon” is not “flying to the moon for a longer time”



Distinguishing Perdurants

- Achievement vs. Accomplishment
  - Achievements non-dividable ("Reaching the border")
  - Accomplishments are dividable (“Going to China”)
- State vs. Process
  - States only consist of states of the same type (like “sitting around”)
  - Processes may consist of processes of different types
    - e.g., “studying” consists of “listen to lecture”, “work on project”, “present results”, “write paper”...



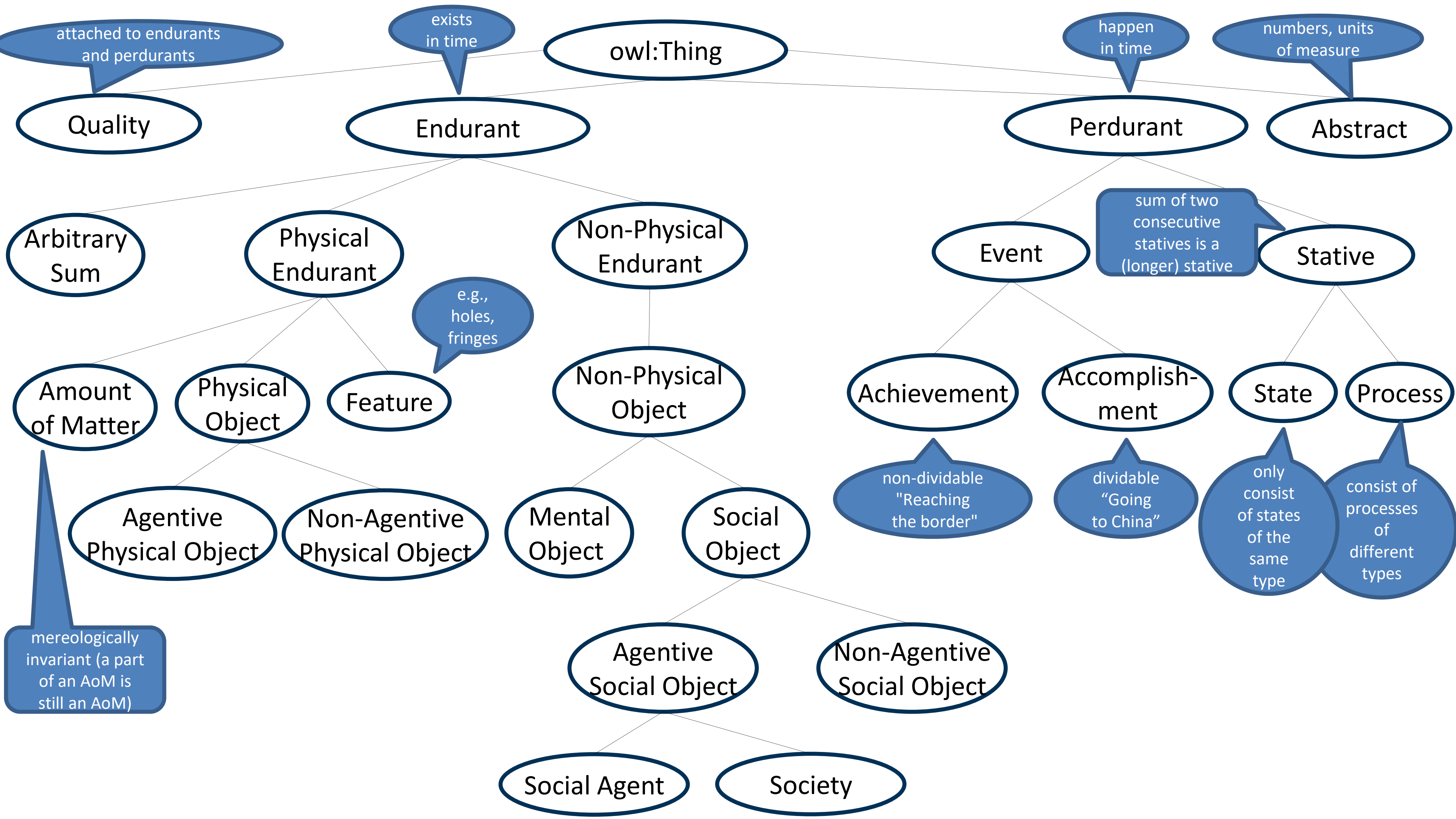
Qualities

- Basic distinction
  - Quality is a property of an entity
  - Quality space is the set of possible values of the quality
- Qualities need entities
  - In general, all particulars can have qualities
  - Qualities only exist as long as the entity exists

Qualities

- Example:
  - Color is a quality
  - RBG is a quality space
- "Two cars have exactly the same color"
  - Every car has got its own quality “color”
  - Both qualities have the same value in the quality space
- Why should each car have its own quality?
  - Qualities only exist as long as the entity they belong to
  - Otherwise, the second car would have no more color once the first car ceases to exist

DOLCE - Overview



Other Top Level Ontologies

- SUMO: Suggested Upper Merged Ontology
  - Around 1,000 classes
  - Strong formalization in KIF (Knowledge Interchange Format)
- Cyc: stems from *EnCyClopedia*
  - Own language (CycL)
  - Top Level and deep general ontology
  - ~250,000 classes
  - OpenCyc: discontinued, but still available
- PROTO: PROTo ONtology
  - General top level+ upper level, different domain extensions
  - ~300 classes, ~100 relations

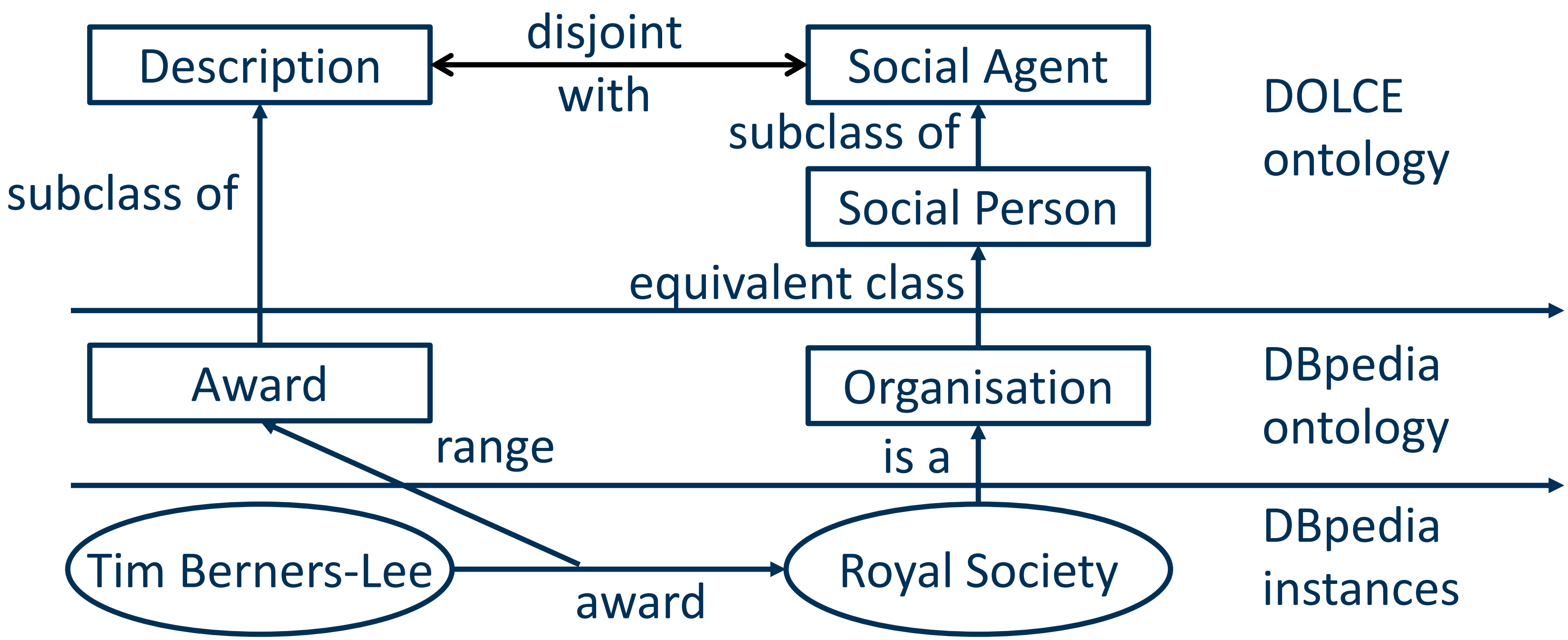
Comparison

- Size: CyC >> SUMO > PROTON > DOLCE
- Level of formalization: SUMO > DOLCE > CyC > PROTON
- Radically different definitions
- Example: time interval
  - In DOLCE: a region (abstract)
  - In SUMO: a quantity (abstract)
  - In PROTON: a happening (~DOLCE:Perdurant)
  - In CyC: e.g., a TemporalThing (~DOLCE:Perdurant) and an IntangibleIndividual (~DOLCE:NonPhysicalEndurant)

- Different top level ontologies are, in general, incompatible!

Example: Usage of DOLCE for DBpedia

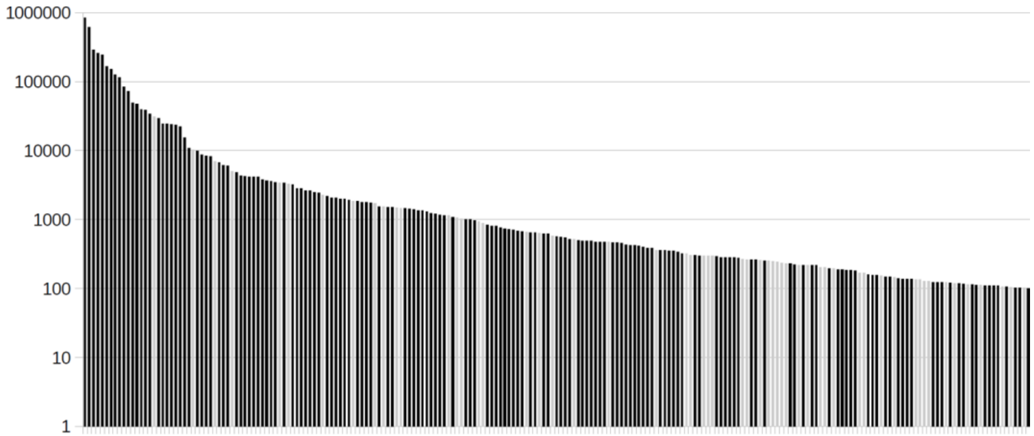
- DBpedia classes and properties
  - are defined as subclasses and -properties of DOLCE since 2014
  - gain: more formal definitions (e.g., domains/ranges, disjointness, ...)



Example: Usage of DOLCE for DBpedia

- 2015 study (Gangemi & Paulheim):
  - 24.4% of all assertions in DBpedia violate DBpedia+DOLCE
  - only 0.7% if only DBpedia ontology is used

- Results
  - identification of typical error clusters
  - refactoring of DBpedia ontology



Wrap-Up

- *Ontology Engineering:* Developing *good* ontologies
  - Given some utility, e.g., correctness of reasoning
- Methodologies, e.g., Methontology
- OntoClean
  - Systematic debugging of ontologies
- Design Patterns & Anti Patterns
  - Small reusable building blocks
  - Common mistakes to avoid
- Top Level Ontologies
  - Basic categories
  - Help structuring ontologies

Questions?