Ontologies

Ontologies

Ulises Cortés

Sergio Álvarez

SID 2019

Knowledge Engineering and Machine Learning Group

UNIVERSITAT POLITÈCNICA DE CATALUNYA

https://kemlg.upc.edu

Contents

- Motivation
- Several views of ontologies
- Ontological agreements
- Ontology types & examples
- Ontology development
- Formal foundations of Ontologies
- Ontology Languages
- Ontologies and Knowledge Representation

Motivation: Communicating Agents…

- Mutual understanding:
  - Translation between representation languages
  - Share the language's semantic content
- Three components in communication:
  - **Interaction protocol**
    - *How* are conversations/dialogues structured?
  - **Communication Language**
    - *What* does each message means?
  - **Transport protocol**
    - *How* messages are actually sent and received by agents**?**

Communication and Knowledge Level

- Agents can be considered as (virtual) Knowledge Bases
- 3 representation levels
  - A language/formalism to represent domain knowledge
    - Ontology
  - A language to express propositions (to exchange knowledge)
    - Content language (for messages)
  - A language to express attitudes for those propositions
    - Agent Communication Language (for languages)

What is an ontology?

Webster:

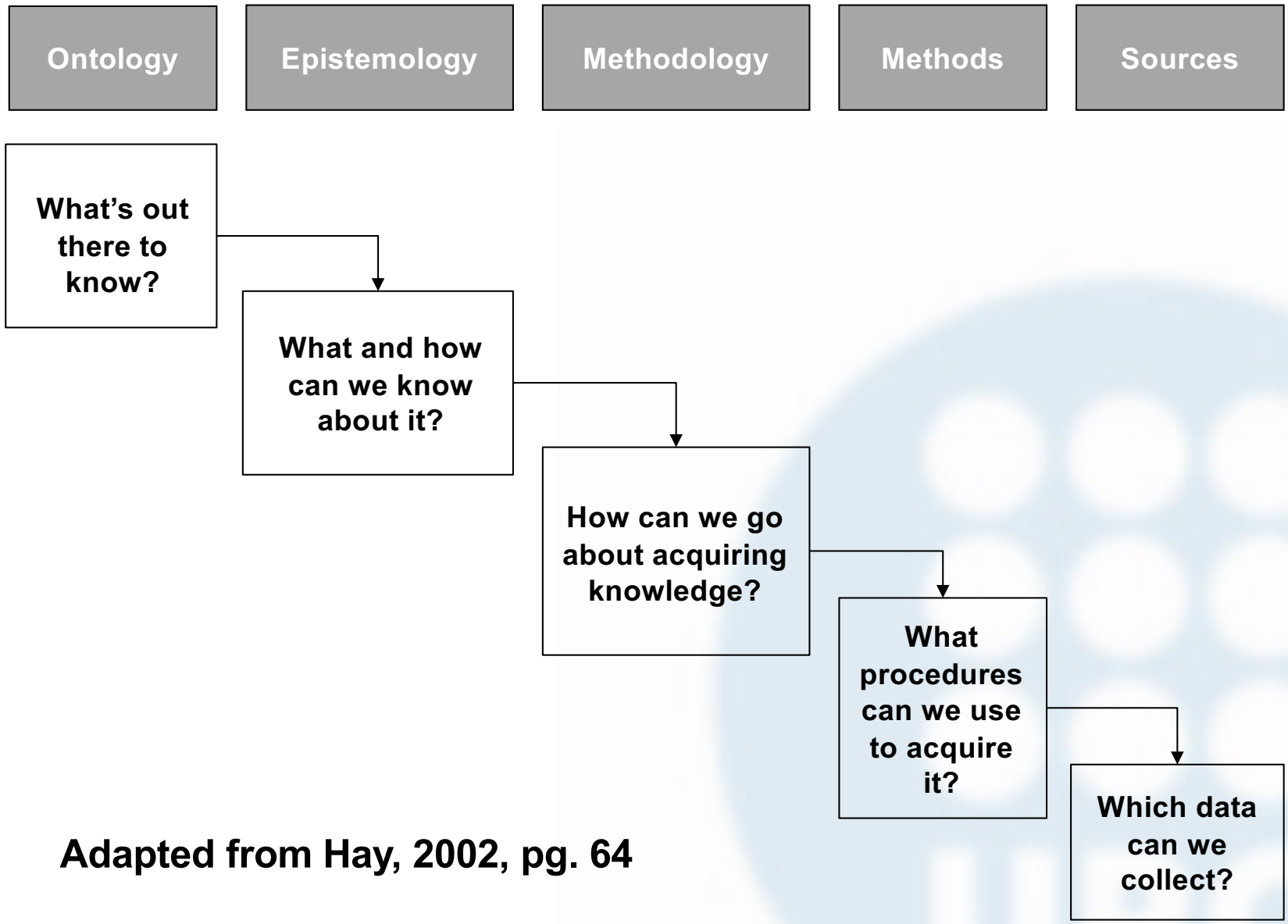
Main Entry: on·tol·o·gy
Pronunciation: än-'tä-l&-jE
Function: *noun*
Etymology: New Latin *ontologia*, from *ont-* + *-logia* -logy
Date: circa 1721

1 : a branch of metaphysics concerned with the nature and relations of being

2 : a particular theory about the nature of being or the kinds of existents

Ontology and Epistemology

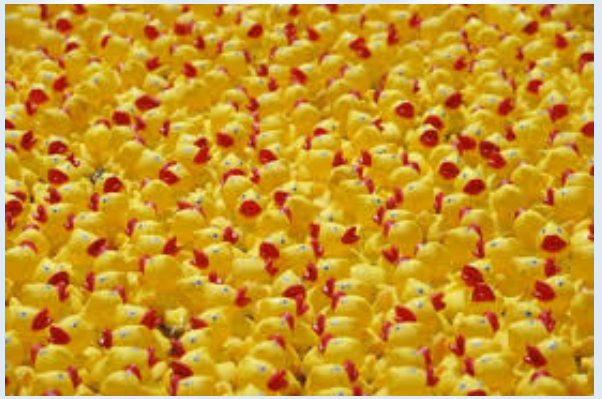
- **Ontology**: the question of ***what reality is like, the basic elements it contains***
- **Epistemology**: ***study of the criteria by which we can know what does and does not constitute warranted, or scientific, knowledge***
- **Methodology**: theories of gathering knowledge, *how* we can know what we are able to know



Adapted from Hay, 2002, pg. 64

Ontology and Epistemology

- The way we think the world is (**ontology**) influences: what we think can be known about it (**epistemology**); how we think it can be investigated (**methodology and research techniques**);
  - the kinds of theories we think can be constructed about it.
- *If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.*



Ontologies
Origin

- Ontology development is directly related to Philosophy.
- Aristotle coined the term Category as the word to describe the different classes things in the world can be divided into.
- The term Ontology is quite modern (XIX century). It comes from Greek *Ontos* (of being/to be) and *Logos* (word).
- Its use started to differentiate studies about the *categories of being* from those categories in biology.
- In fact categorization is a common task in several science areas (Philosophy, Biology, Medicine, Linguistics...)

Ontologies
Definitions

- “An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary” [Neches 91]
- “An Ontology is an explicit specification of a conceptualization” [Gruber 93]
- […] an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents.
- http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

Ontologies

- Ontology aims to study the categories that exist in a given domain.
- The result of this study is an ontology.
  - A catalogue of the different kinds of objects that we assume as existing in a given domain **D**, from the perspective of someone that uses a language **L** in order to talk about **D**.
- Elements in ontologies represent *predicates*, *constants*, *concepts* and *relationships*
- An ontology can be seen as the vocabulary that agents need to use in order to talk about a given domain.

Ontologies: some assumptions

- **Absolute truth** might never be found.
- Data, **evidence** and rational considerations shape knowledge.
- Research is a process of making claims and then testing, refining or abandoning some of them for other claims more strongly warranted.
- Research seeks to develop relevant, true statements that can serve to explain the situation that is of concern or that describes the causal relationship of interest.
- Researchers must examine their methods and conclusions and control or **limit bias**

Classes and Relations in Simple Time

Classes

- Day-Name
  - Day-Number
  - Duration
  - Hour-Number
  - Minute-Number
  - Month-Name
  - Month-Number
  - Second-Number
  - Time-Point
    - Calendar-Date
    - Calendar-Year
    - Universal-Time-Spec
  - Time-Range
  - Year-Number

Relations

- <
- >
- After
- After=
- Before
- Before=
- Disjoint-Time-Ranges
- During
- During=
- Equals
- Finishes
- Finishes=
- Meets
- Overlaps
- Overlaps=
- Start=
- Starts

Definition of Time Point

```
(def-class TIME-POINT (time-position)
  "A point in time"
  ((second-of :type second-in-time :max-cardinality 1 )
   (minute-of :type minute-in-time :max-cardinality 1 )
   (hour-of :type hour-in-time :max-cardinality 1 )
   (day-of :type day-in-time :max-cardinality 1)
   (month-of :type month-in-time :max-cardinality 1)
   (year-of :type year-in-time :max-cardinality 1 ))
  :constraint (and
                (not (and (month-of ?x 2)
                          (> (the ?day
                               (day-of ?x ?day))
                             29)))
                (not (and (member-of ?x (4 6 9 11))
                          (> (the ?day
                               (day-of ?x ?day))
                             30)))))
```

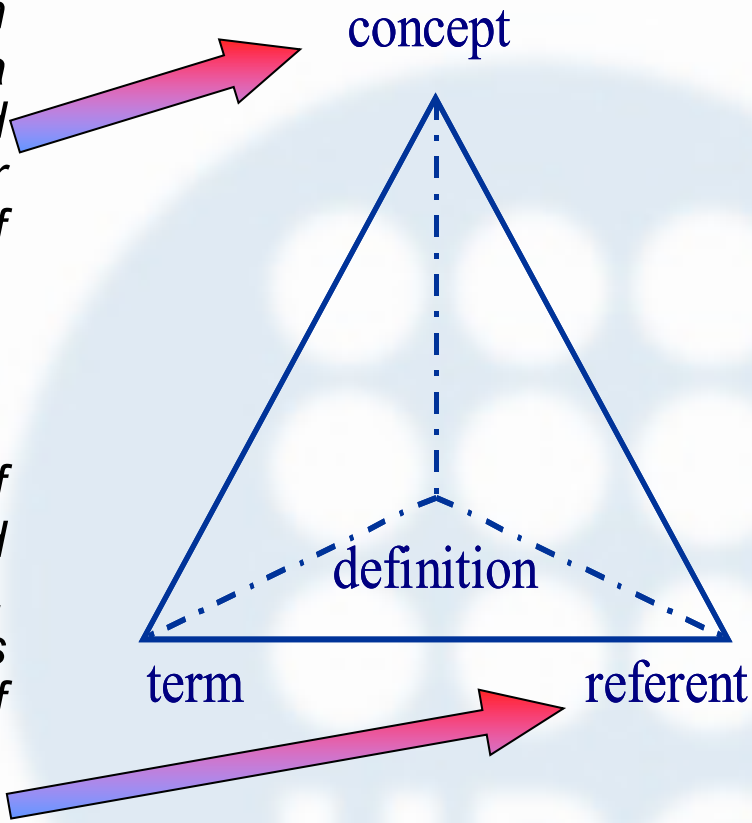
Ontologies vs. data models

- No strict line in between, but ontologies are
  - more general
  - more reusable
  - intended for multiple purposes, goals, and users
  - more easily shareable
  - take stand on semantics of concepts (as opposed to mere structure and integrity)

“Ontological” extension

- **In Information Science:**
  - *“An ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents.”* (Tom Gruber)

- **In Philosophy:**
  - *“Ontology is the science of what is, of the kinds and structures of objects, properties, events, processes and relations in every area of reality.”* (Barry Smith)



Ontologies in Logics

Motivation

- The ability to produce deductions from the information represented in an ontology is given by a Logic.
- Logic by itself tends to be neutral with respect to meaning:

$$\frac{P \rightarrow Q \quad P}{Q}$$

- The above reasoning refers to nothing unless we give meaning to the atoms (P = “it rains”, Q = “I’m wet”).
- When we combine a logic with an ontology, the ontology provides the logical formalism with the capability to express meanings with its statements.

Ontologies in Logics

Motivation

- This is specially useful in higher-level logics

- **First-Order Logic**:
  - `citizen(x)` → `human(x)`
  - What do '`citizen`' and '`human`' mean?
- **Dynamic Logic**
  - `[order(x,item,t)] ack-order-received(x,item)`
  - What is the meaning of the '`order`' action?
  - What do the '`ack-order-received`' predicate means?
- **Deontic Logic**
  - If `won(x, auct, item, pr)` → `O(pay(x,item,pr))`
  - What is the meaning of the '***won***' predicate?
  - What is the meaning of its second parameter?
  - What does '***pay***' mean?
  - Is there more than one way *to* **pay**?

Description Logic

Description Logics
- overcome the ambiguities of early semantic networks and frames
- first realized in the system KL-One [Brachman and Schmolze, 1985]

Well-studied and decidable (most DL languages)

Tight coupling between theory and practice

TBox and ABox

TBox: *terminology*

the vocabulary of an application domain:

Concepts: sets of individuals

Roles: binary relationships between individuals.

Examples:

Concepts: Person, Female, Mother

Role: hasChild, meaning that some person is the child of some other

ABox: *assertions*

about named individuals in terms of this vocabulary

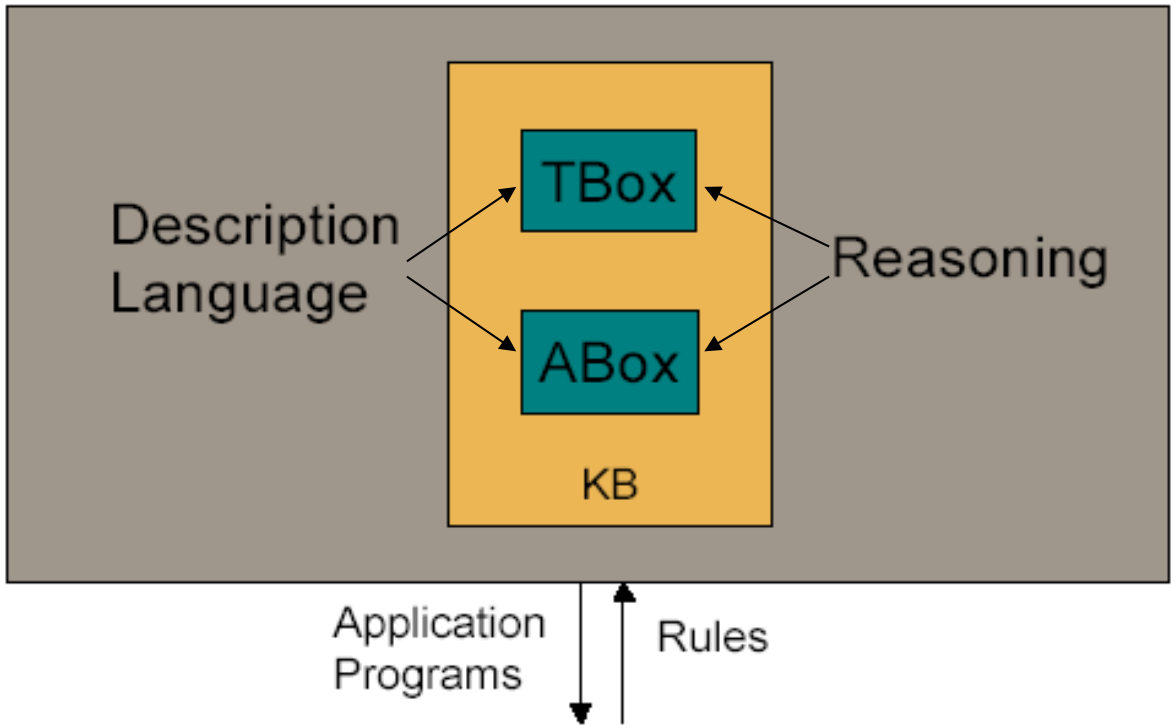
Example

Elizabeth and Charles are Persons. We write this as Person(Elizabeth), and Person(Charles).

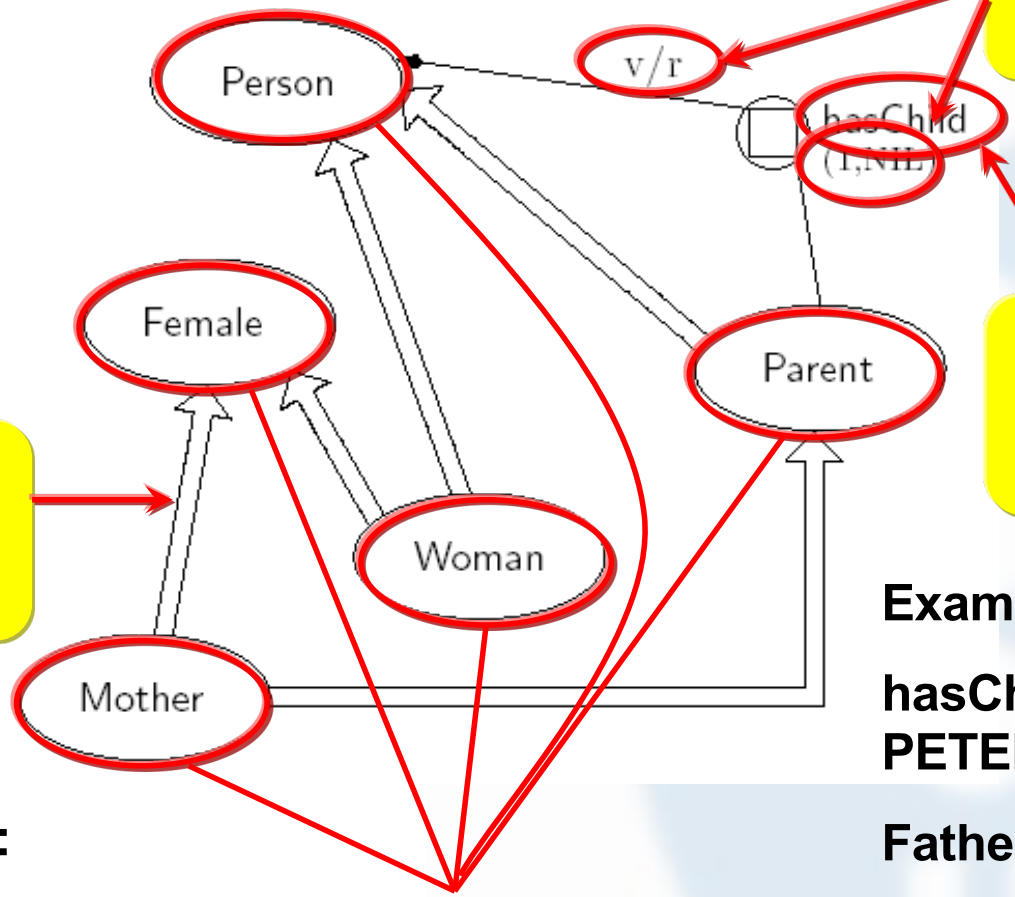
Individuals, like "myCar", have attributes, like "color", and those attributes have values, like "red". When this happens we say that red is the colorOf attribute of myCar. We write this as colorOf(myCar, red).

Knowledge Base = TBox + ABox

Architecture of a DL System



An Example about Family Relationships



value restriction(v/r): (1, NIL)

a *role* defines a property of *Parent: hasChild.Person*

"IS-A" relationship defines a hierarchy over concepts

Examples in ABox:

hasChild(MARY, PETER)

Father(PETER)

Examples in TBox:

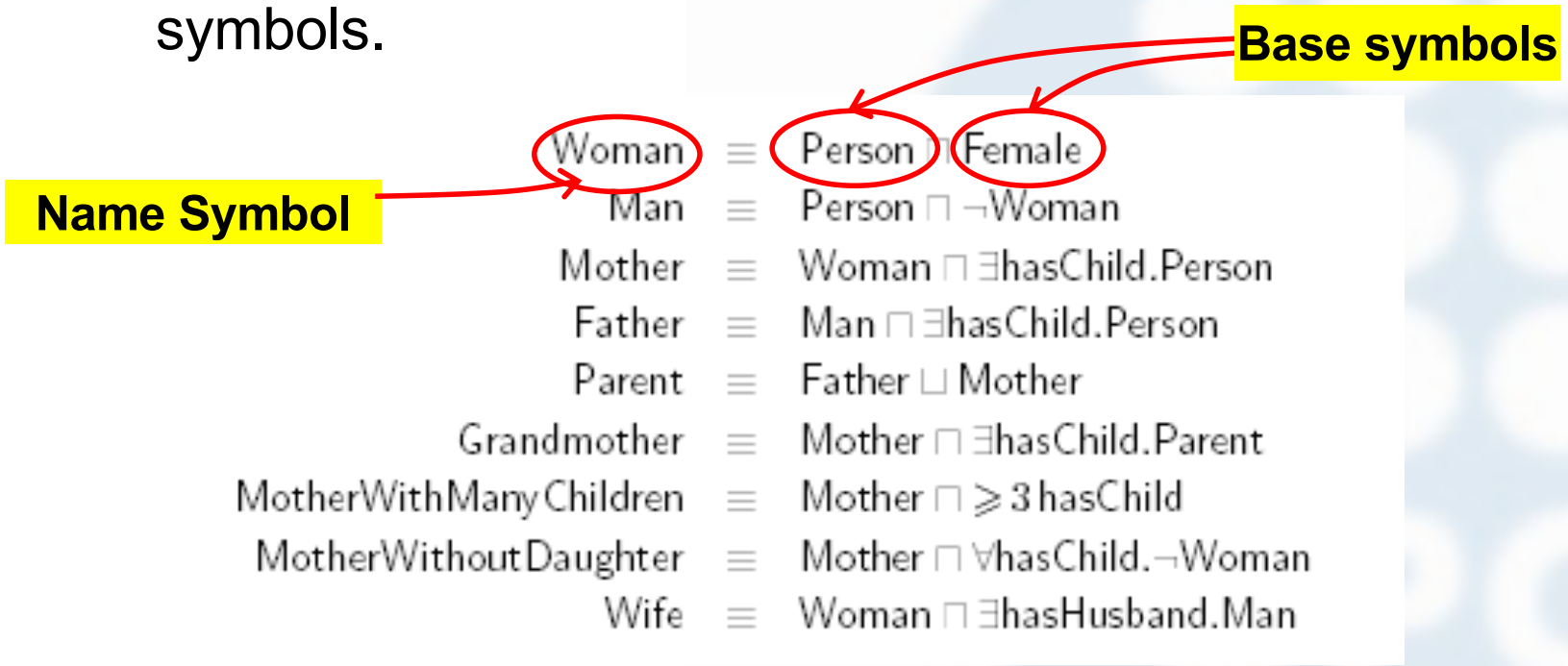
Woman ´ Person u Female

Mother ´ Woman u hasChild.Person

These are concepts

Name Symbols vs. Base Symbols

- atomic concepts occurring in a TBox **T** can be divided into two sets, ***name symbols*** $\mathbf{N_T}$ (or defined concepts) and ***base symbols*** $\mathbf{B_T}$ (or primitive concepts, occur only on the right-hand side)
- a ***base interpretation*** for **T** only interprets the base symbols.



Ontologies in Knowledge Engineering

Motivation

- To enable human experts to model and analyse their problem solving expertise; this expertise analysis should be suitable to feed into the conceptual knowledge system design process.

- To enable and support reuse of previous constructed knowledge bases. Reuse meaning *knowledge decompilation*.

Ontologies in Knowledge Engineering

Motivation

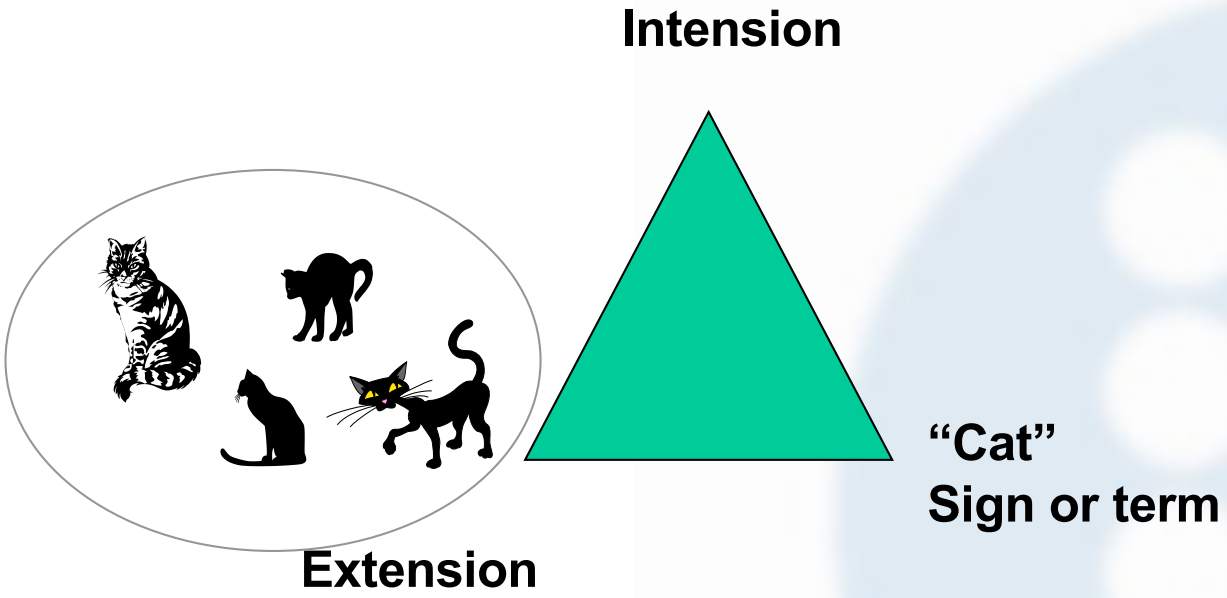
- Ontologies divide domain knowledge from operational knowledge
  - Allows to independently develop the techniques and algorithms to solve a problem from the concrete knowledge about the problem
- They allow analysis over domain knowledge
  - Once we have a knowledge specification, it can be analysed by means of formal methods (correctness, completeness …)

Ontologies in Distributed AI

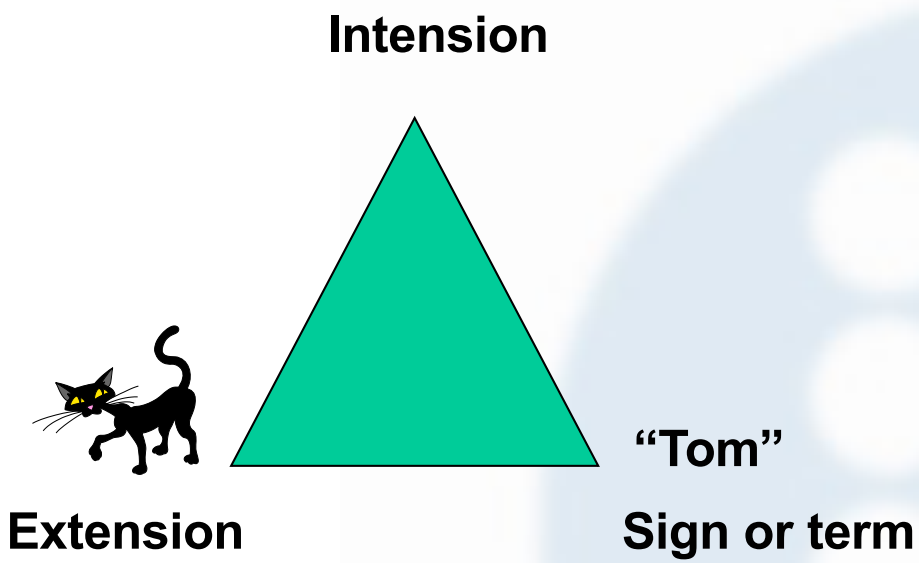
Motivation

- To allow sharing an interpretation of information structure between people/agents
  - By creating an ontology about a domain, agents can understand each other (unambiguously) and know what the other means with each message
- To allow knowledge reuse
  - Create a domain description which can be used by other applications which should use/share knowledge about that domain
- To make explicit the interpretations about the domain
  - Interpretations about concepts, predicates… can be compared. If conflicts arise, a common interpretation can be agreed upon.

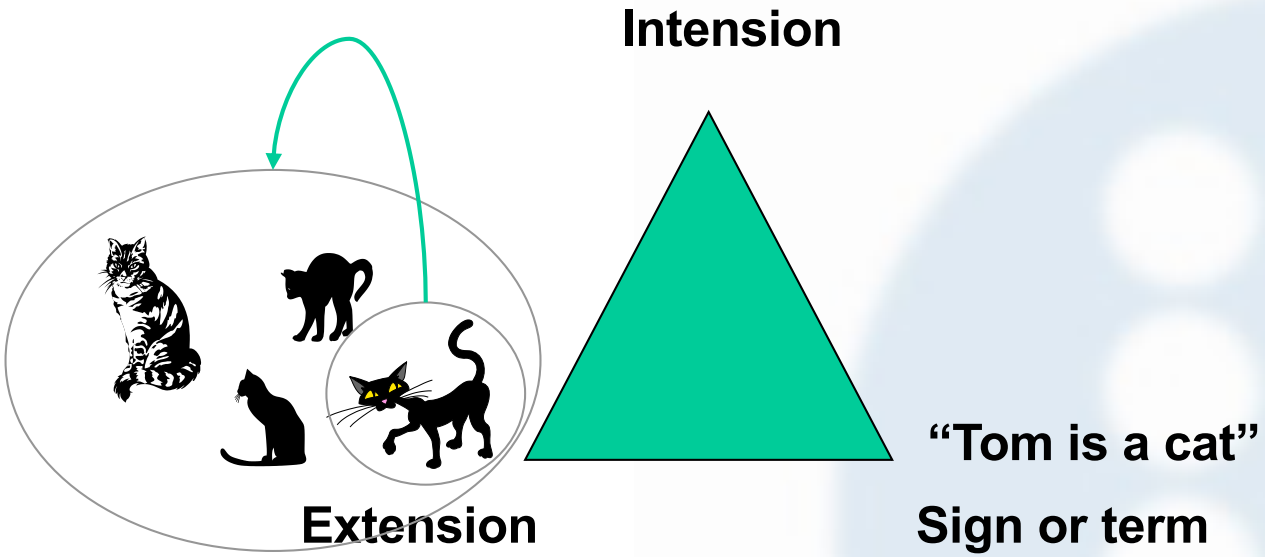
"Semiotic Triangle" (or one version of it)



"Semiotic Triangle" (2)

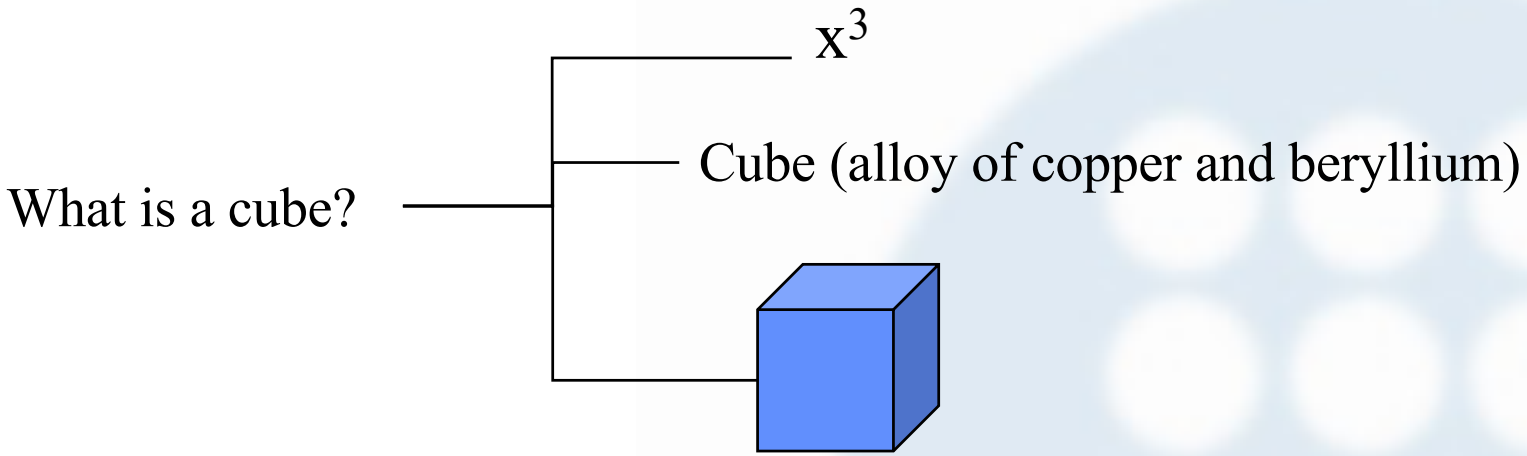


"Semiotic Triangle" (contd.)



Ontological Agreements

Agreements to use the vocabulary in a coherent and consistent way.



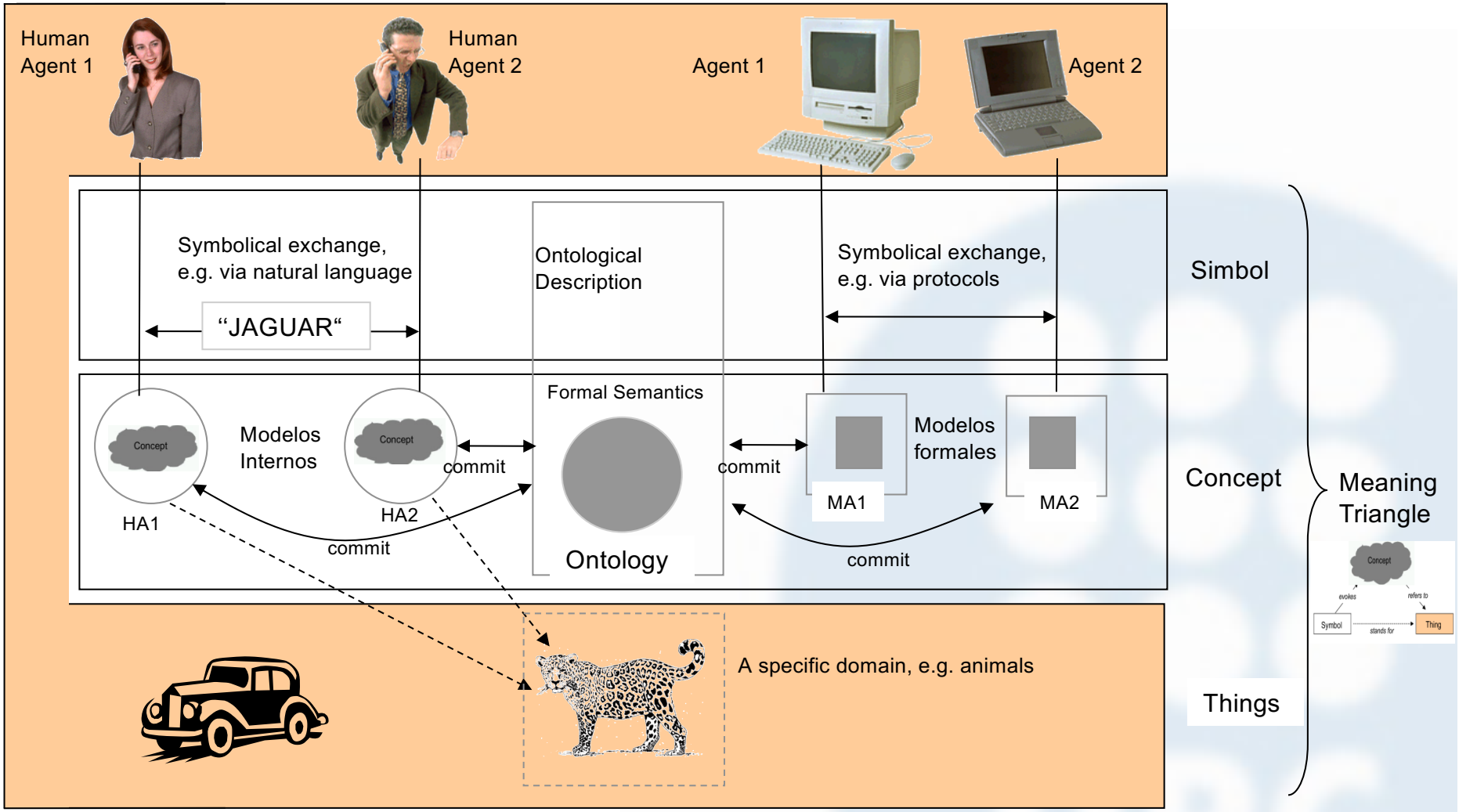
Why Ontologies? (review)

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

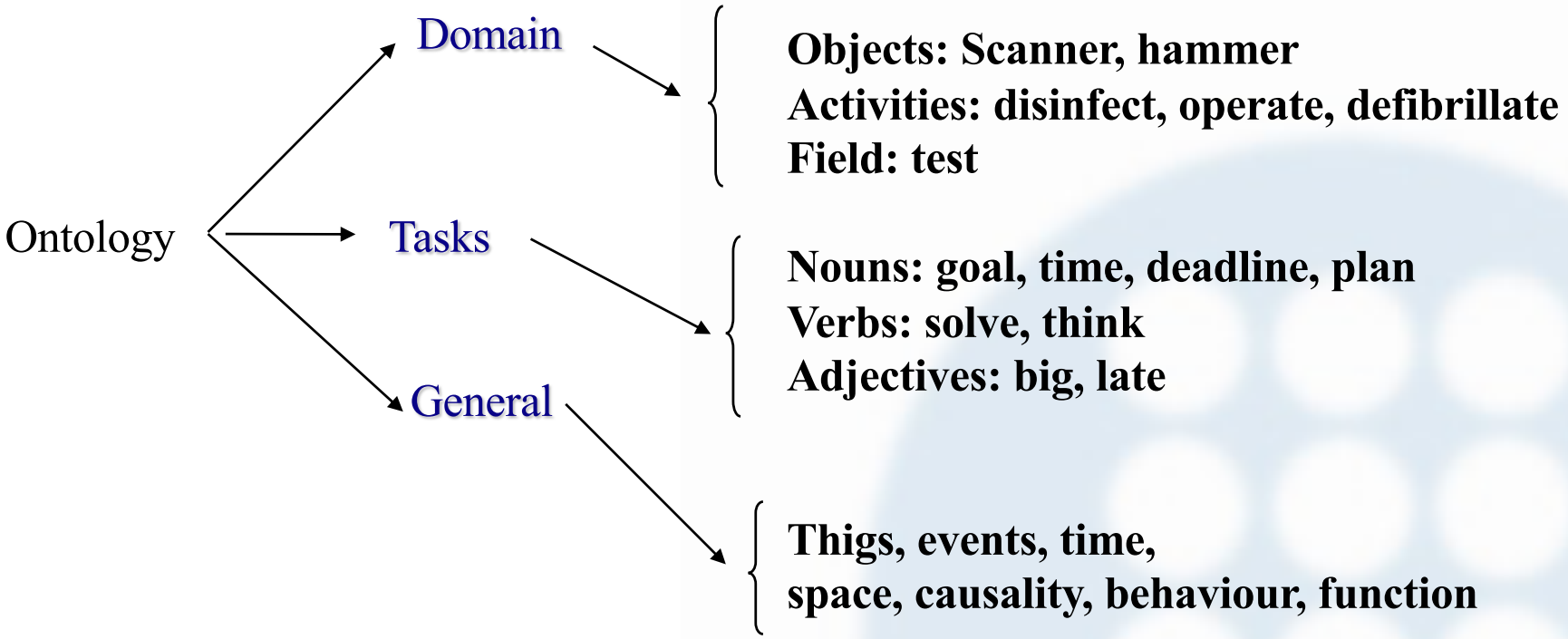
Ontological agreements

Man-machine communication

[Maedche et al., 2002]

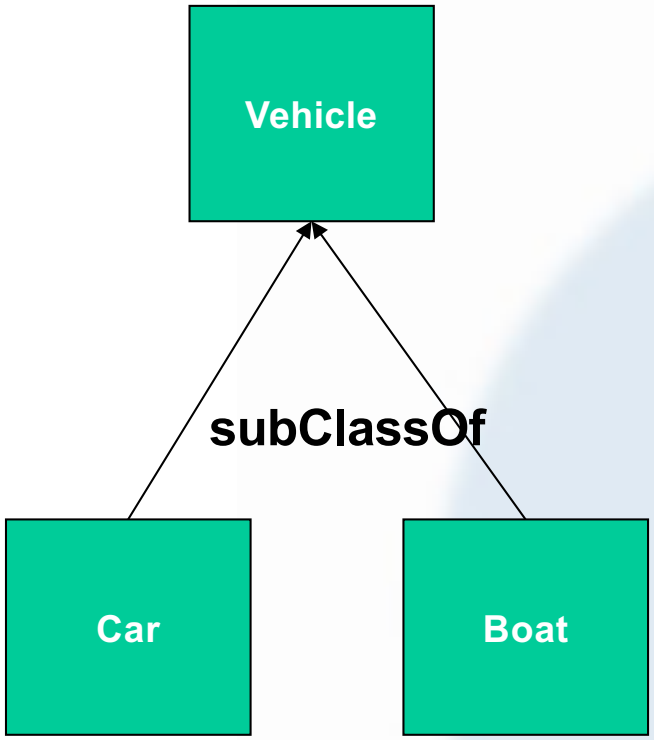


Ontology types

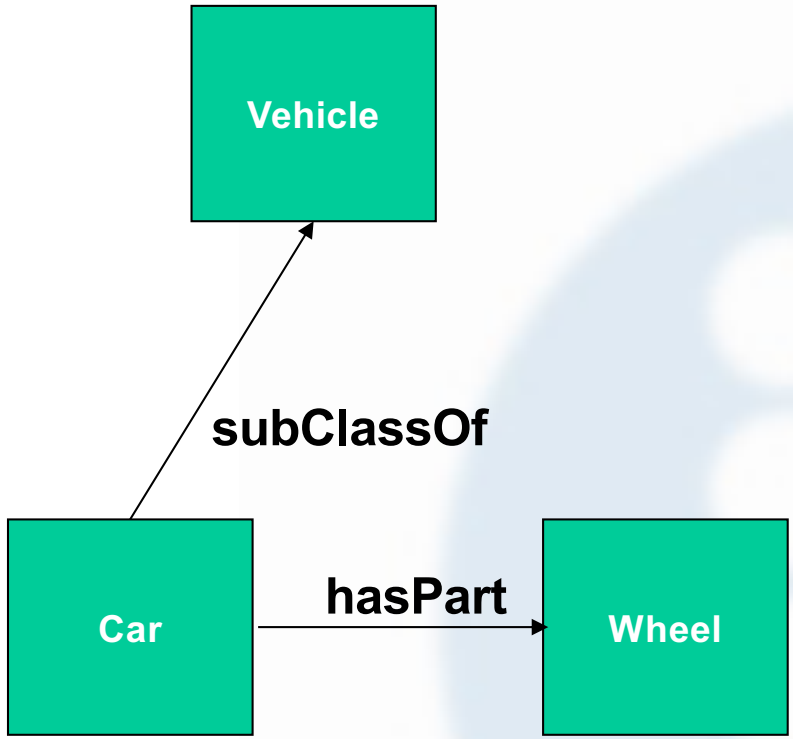


Ask & Tell ontologies : limited subset of what can be queried and answered
Meta-Ontologies

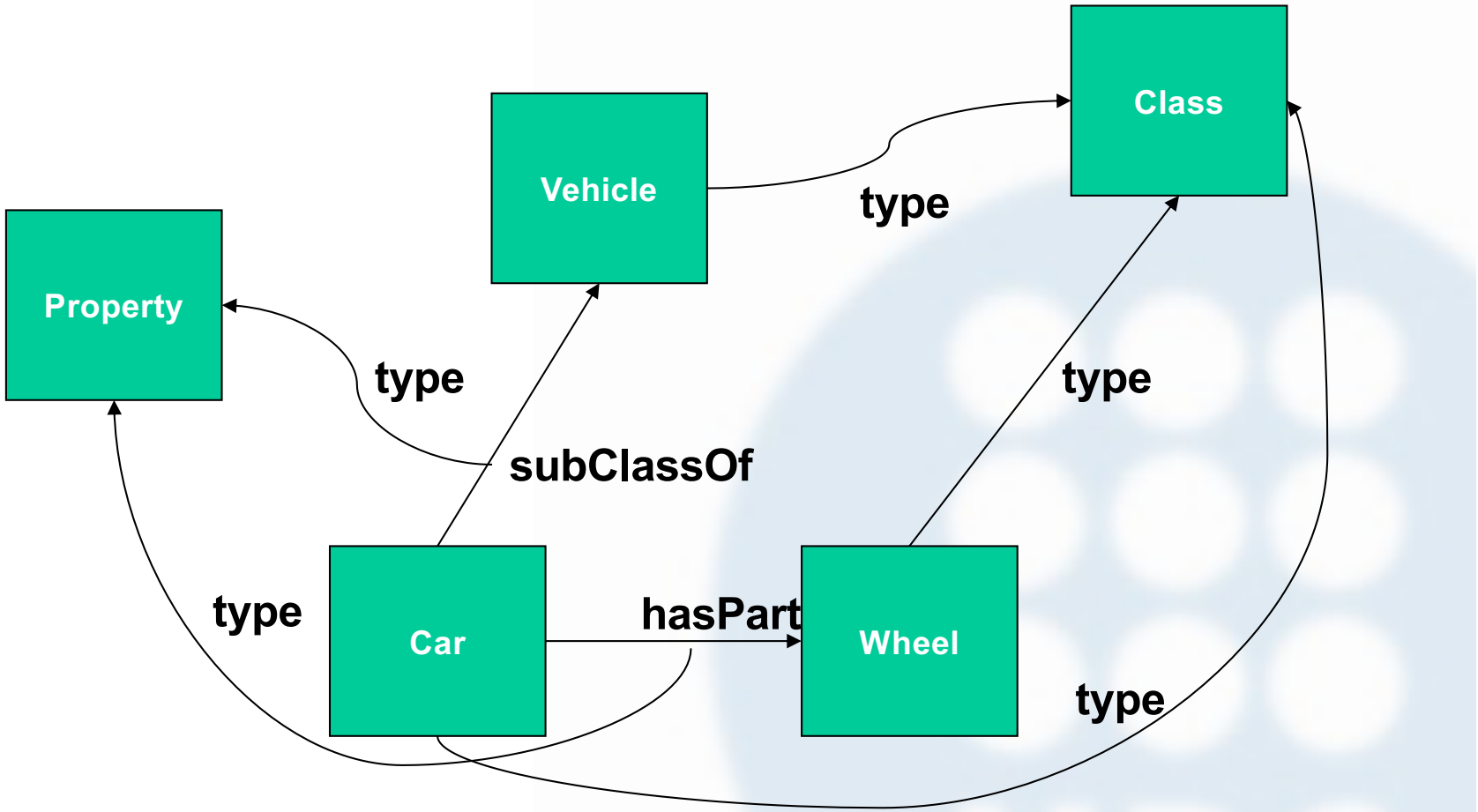
A simple example ontology



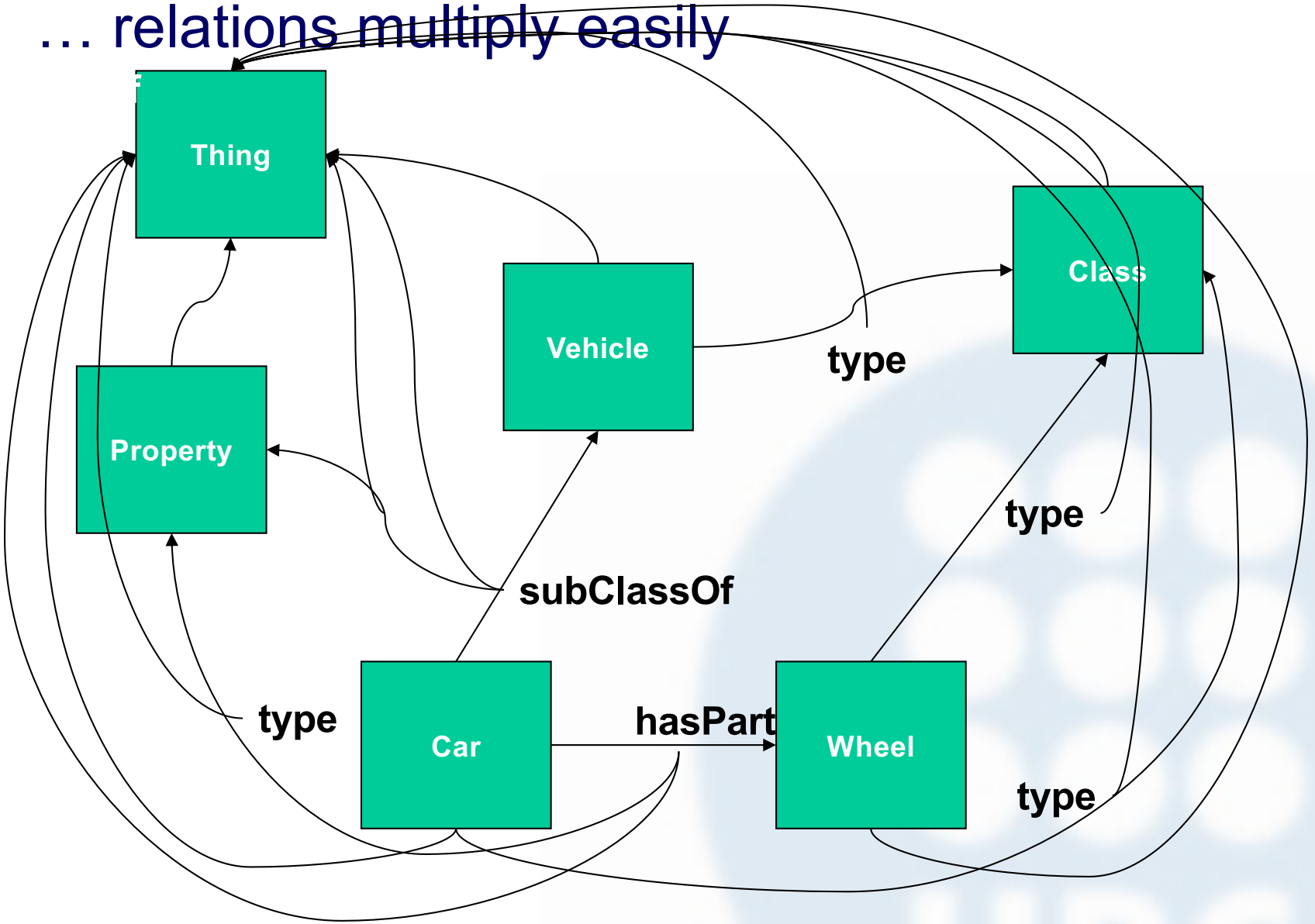
Almost as simple



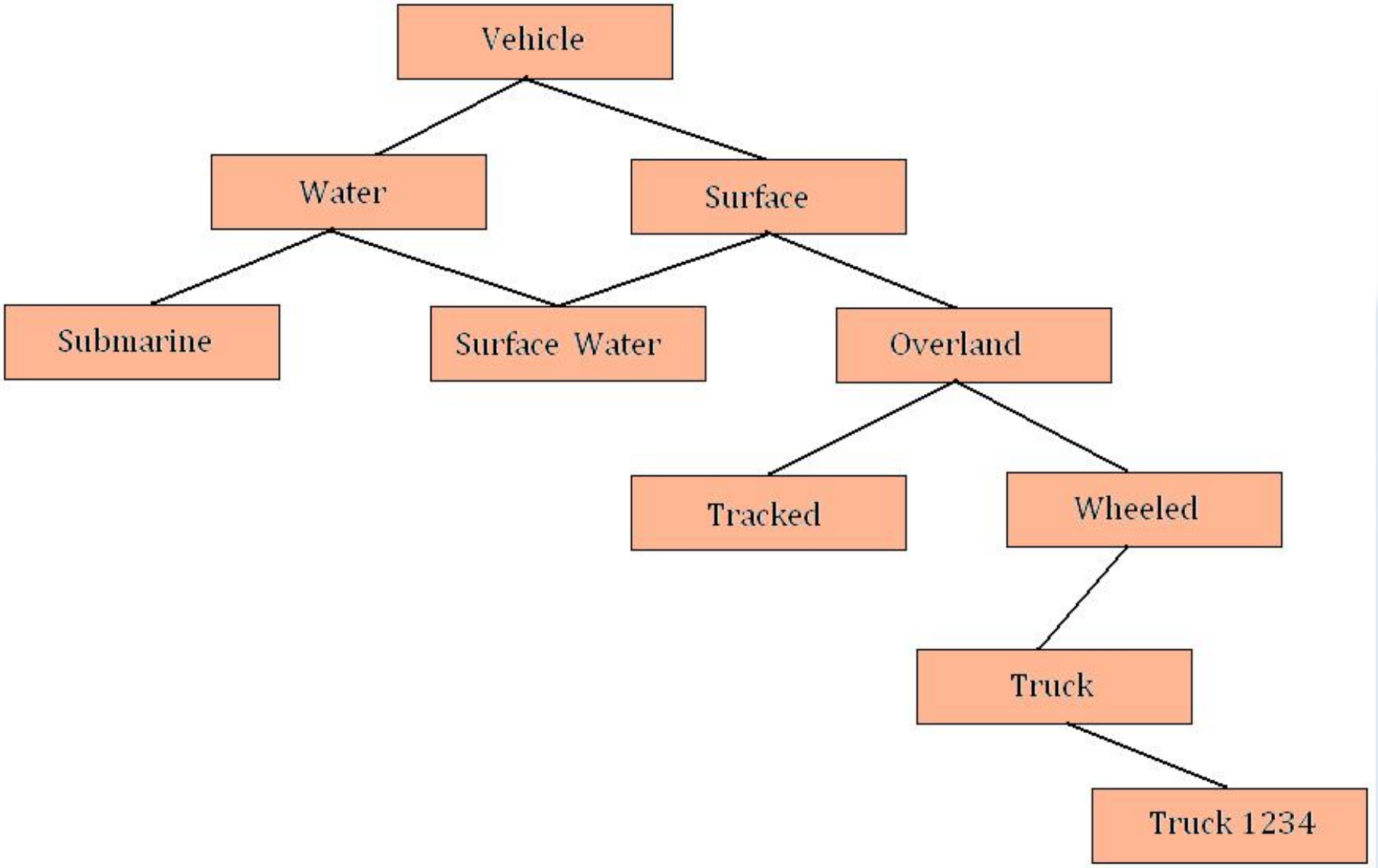
Gets a bit tricky...



… relations multiply easily



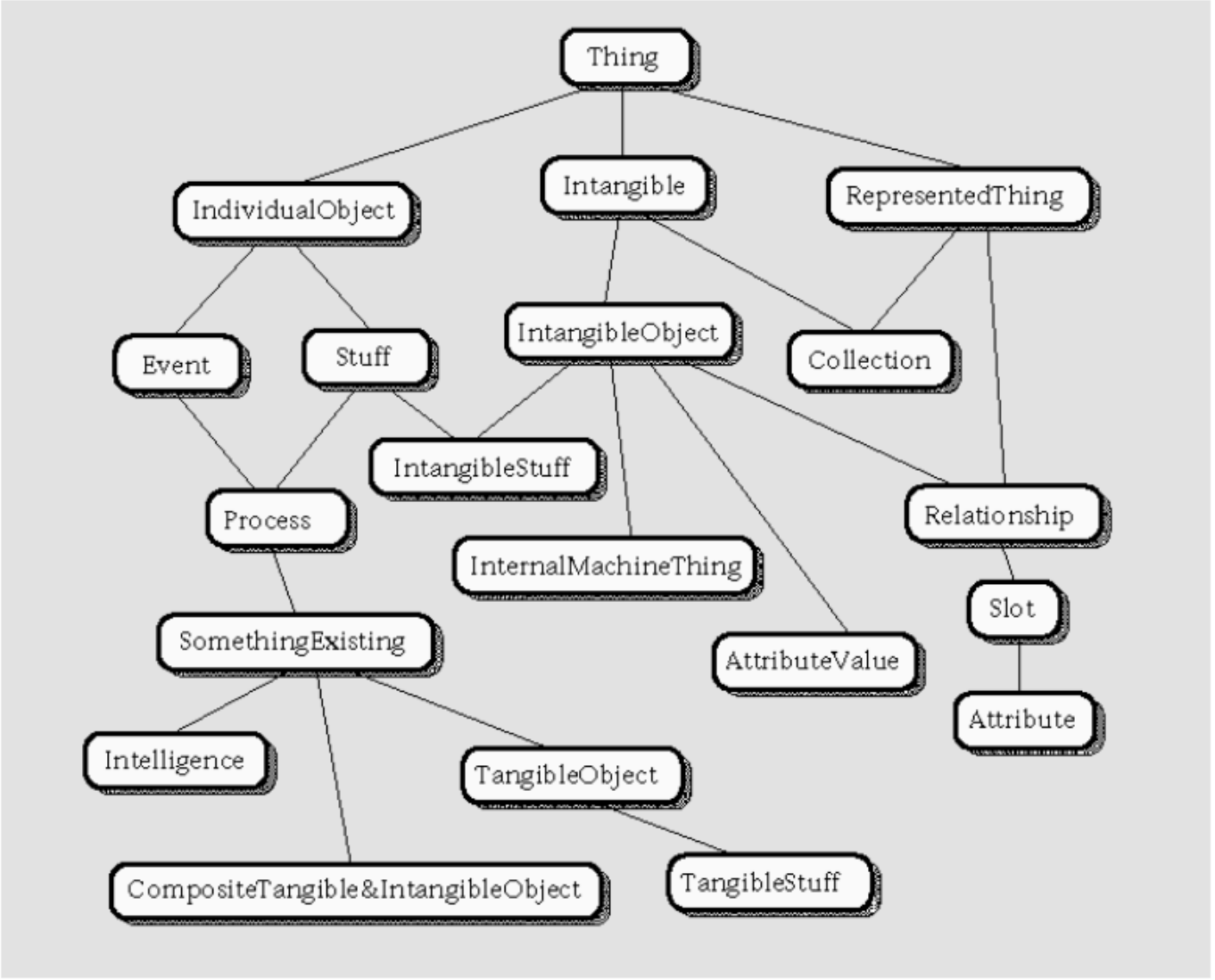
Hierarchy in Ontology



Map of High-Level Cyc Topics

Ontology examples

CYC (1985) / Open CYC (2003)



The case of Cyc

Cycorp: "**It's just common sense**"

In CycL over 1 000 000 hand-entered assertions (or "rules") designed to capture a large portion of what we normally consider consensus knowledge about the world ("river flows downhill" *etc.*)

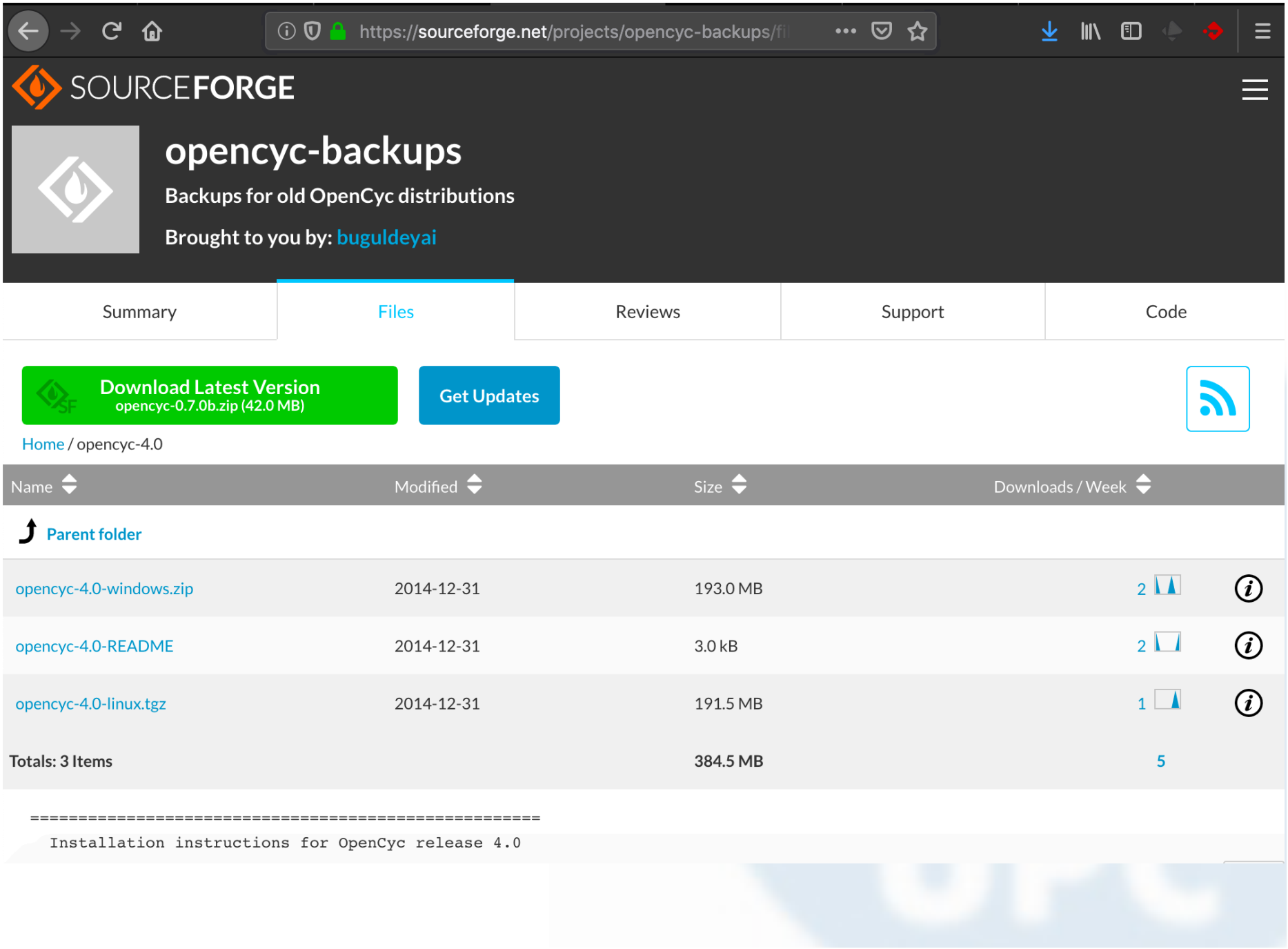
- Sometimes so called ***microtheories*** are however necessary
  - difference with domain ontologies?
- → Nevertheless exemplifies the "One large ontology" approach!

Opencyc: an open source subset of Cyc

- 6 000 concepts: an upper ontology for all of human consensus reality.
- 60 000 assertions about the 6 000 concepts, interrelating them, constraining them, in effect (partially) defining them.
- Opencyc has OWL serialization
  - Note however that Cyc is not a Semantic Web phenomenon but been under development since 1984

There also exists CycL-to-Lisp, CycL-to-C, etc. translators

It uses Google as the search engine in the background.


https://sourceforge.net/projects/opencyc-backups/fi
SOURCEFORGE
opencyc-backups
Backups for old OpenCyc distributions
Brought to you by: buguldeyai
Summary
Files
Reviews
Support
Code
Download Latest Version
opencyc-0.7.0b.zip (42.0 MB)
Get Updates
Home / opencyc-4.0
Name
Modified
Size
Downloads / Week
Parent folder
opencyc-4.0-windows.zip 2014-12-31 193.0 MB 2
opencyc-4.0-README 2014-12-31 3.0 kB 2
opencyc-4.0-linux.tgz 2014-12-31 191.5 MB 1
Totals: 3 Items 384.5 MB 5
==========
Installation instructions for OpenCyc release 4.0

#$Person Search Clear

Assert Compose KE Text Create Documentation History Query

You are: Guest [Logout]

Preferences Tools

Person

order by : predicate **ontology**
filters : predicate ontology
index view : **inferred** legacy

▶ Browsing ▶ Editing ▶ Advanced

All Assertions *(6265)* ! ▶ open all
▼ via Person *(5817)* !
- isa *(12)*
- isa arg2 *(2726)* !
- quotedIsa *(4)*
- genls *(10)*
- genls arg2 *(1174)* !
- disjointWith *(20)*
- disjointWith arg2
- comment
- arg1Genl arg2 *(36)*
- arg1Isa arg2 *(248)* !
- arg2Genl arg2 *(37)*
- arg2Isa arg2 *(343)* !
- arg3Genl arg2 *(7)*
- arg3Isa arg2 *(6)*
- arg4Genl arg2 *(4)*
- argGenl arg3 *(84)*
- argIsa arg3 *(592)* !
- argSometimesIsa arg3 *(26)*
- broaderTerm *(2)*

Collection : Person [+]

on the term

isa : Analyst-PertinentConcept TerroristAttackTargetTypeType ExistingObjectType
not isa : AtemporalNecessarilyEssentialCollectionType
isa : ClarifyingCollectionType
isa : OrganismClassificationType
isa : KEClarifyingCollectionType
isa : CandidateKBCompletenessNode
isa : CalaisOntologyEntityType
isa : KEClarifyingCollectionType
isa : (SampleInstanceOfTypeForProgramFn
(SpecsFn PartiallyTangible) CycAnalyticEnvironment-TheProgram)
isa : Agents-Topic
(isa MohammedNourAbdelkerim Person)
(isa AleemNasir Person)
(isa VPOfProduct Person)
(isa Person-DummyNamedEntity Person)
(isa Pink-Songwriter Person)
(isa CycAdministrator Person)
(isa MarthaStewart Person)
(isa BlazFortuna Person)
(isa LukaBradesko Person)
(isa
(InstanceNamedFn-Ternary "Israeli soldiers" Person "d94e27c6-17bb-41d9-82d8-

Update Comm: Storing Only Agenda: Idle **KB:** 5022 **Local:** (6) System: 10.140388 Learn about ResearchCyc

Predicate Calculus Representation

Parents love their children

This can be represented as

```
(ForAll ?P (ForAll ?C
   (implies
     (and
        (isa ?P Person)
        (child ?P ?C))
     (loves ?P ?C)))))
```

For all P, For all C, P is a person AND C is a child of P implies P loves C

Reasoning Using Logic

Examples:

Simple:

(isa Socrates Man)
(ForAll ?x (implies (isa ?x Man) (isa ?x Mortal)))

(isa Socrates Mortal) => *Yes*

Harder:

Using general and specific knowledge

Can a can can-can? => No

CycL is Cyc's language

"**Bill Clinton belongs to the collection of U.S. presidents**" and

(#$isa #$BillClinton #$UnitedStatesPresident)

"**All trees are plants**".

(#$genls #$Tree-ThePlant #$Plant)

"**Paris is the capital of France**."

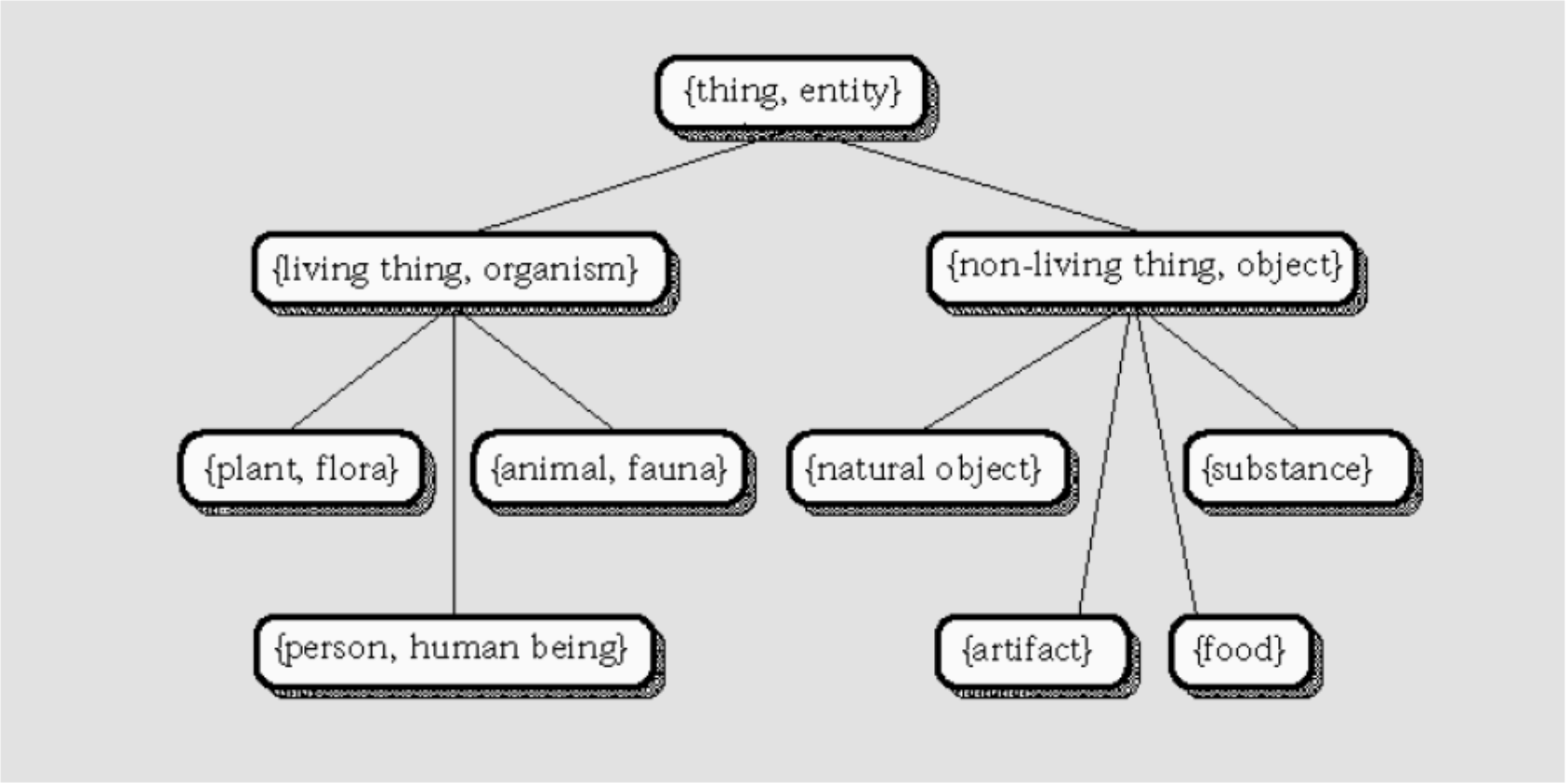
(#$capitalCity #$France #$Paris)

"**a fact about sets**"

```
(#$implies
   (#$and
     (#$isa ?OBJ ?SUBSET)
     (#$genls ?SUBSET ?SUPERSET))
   (#$isa ?OBJ ?SUPERSET))
```

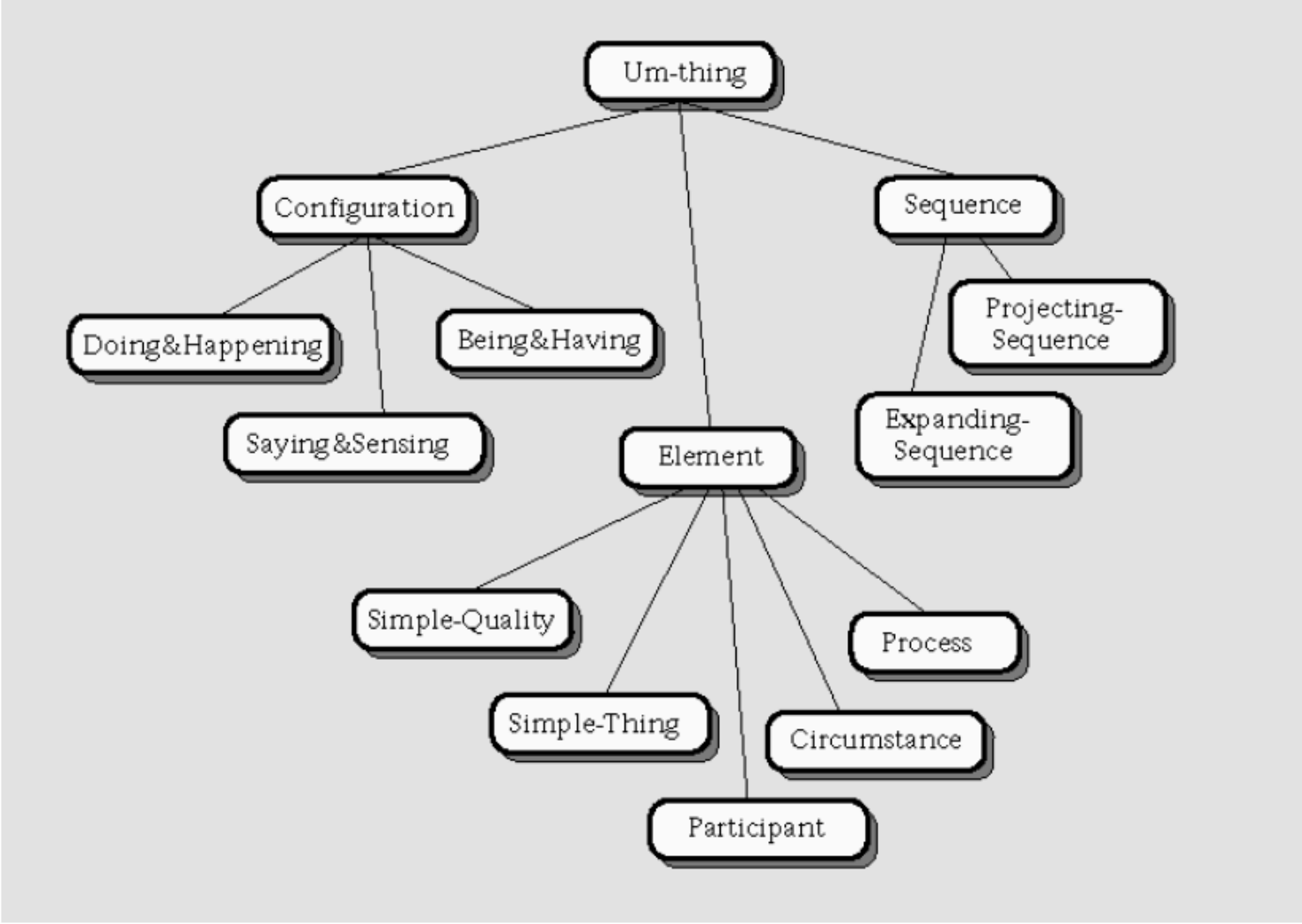
Ontology examples

WORDNET(1990)



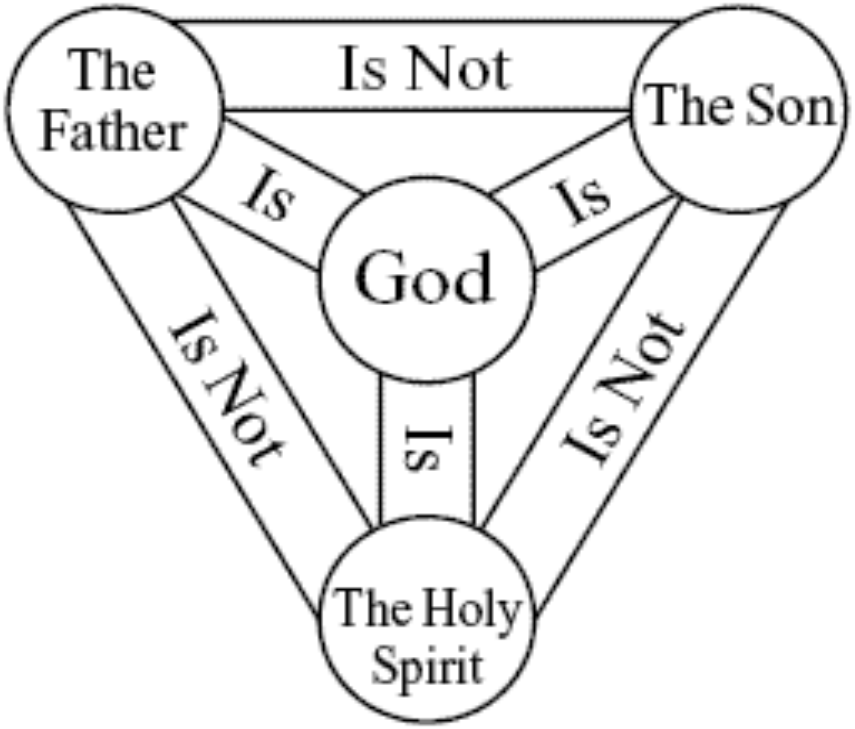
Ontology examples

Generalized Upper Model (1994)



Ontology examples

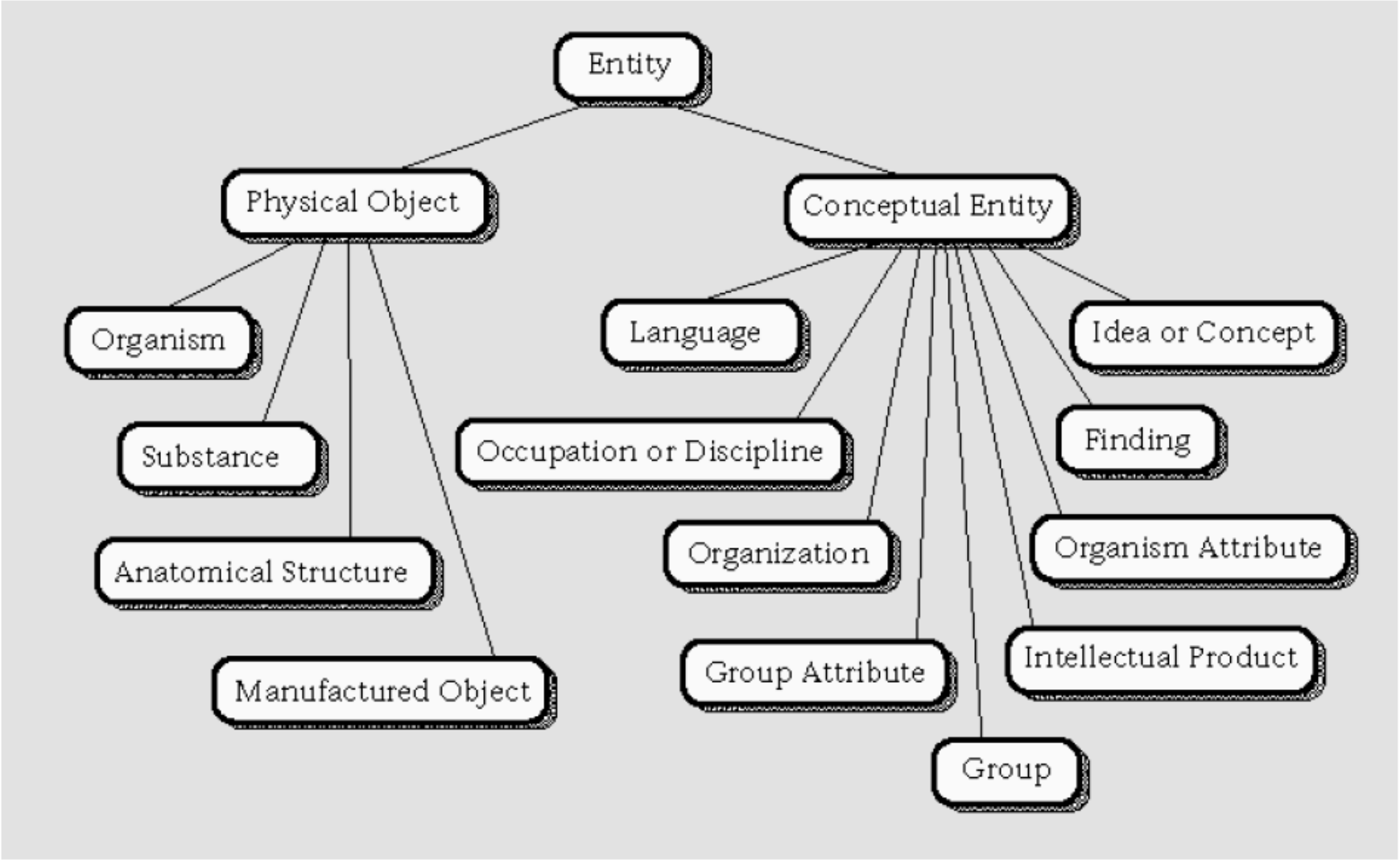
Domain-specific ontologies: Ontology of GOD



[Shield of the Trinity, circa 12th Century aD]

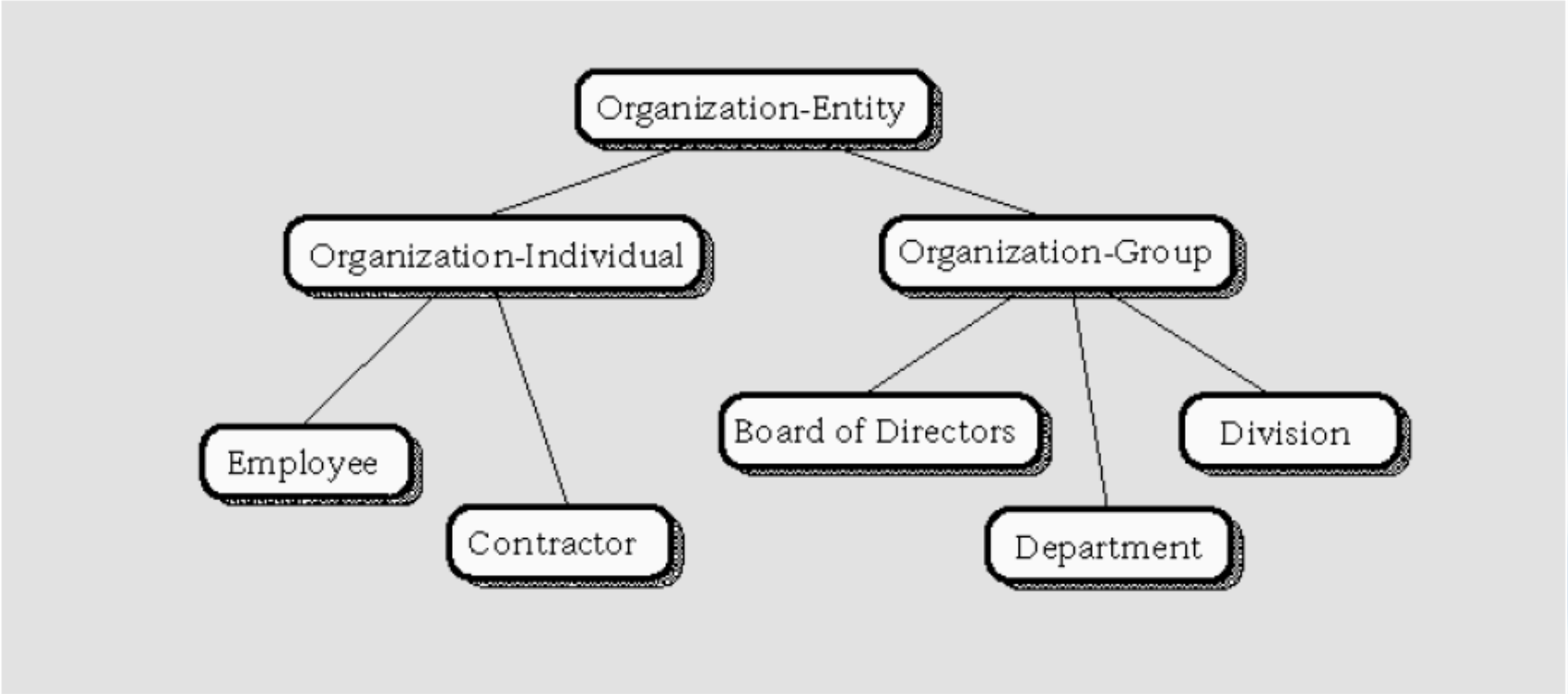
Ontology examples

Domain-specific ontologies: Unified Medical Language System (1993)



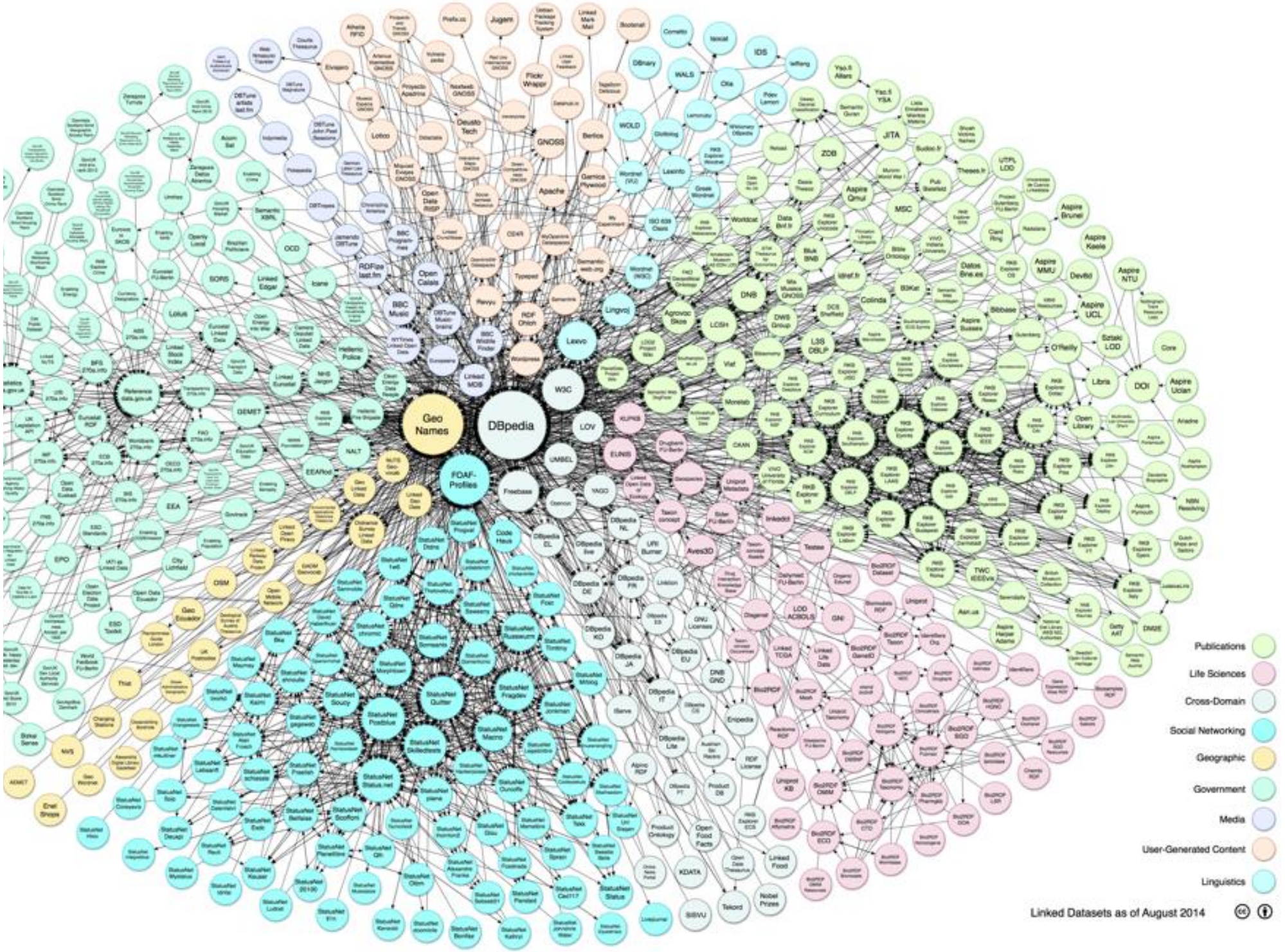
Ontology examples

Domain-specific ontologies: TOVE (1995) – Enterprise ontology



Ontology examples

- Traffic & mobility
  - DATEX II (https://datex2.eu/)
- Healthcare
  - Semantic DICOM (https://bioportal.bioontology.org/ontologies/SEDI)
  - Disease Ontology (http://disease-ontology.org/)
  - HL7 RDF (http://build.fhir.org/rdf)
- Music
  - The Music Ontology (http://musicontology.com/)
- Encyclopedic information
  - DBPedia (https://wiki.dbpedia.org/)
- Generic metadata
  - Dublin Core (http://dublincore.org/)


Geo Names
DBpedia
FOAF-Profiles
W3C
LOV
Freebase
Publications
Life Sciences
Cross-Domain
Social Networking
Geographic
Government
Media
User-Generated Content
Linguistics
Linked Datasets as of August 2014

Ontologies

elements

- Concepts: they serve to model several items including tasks, functions, actions, strategies, plans, etc.
  - Some ontology languages refer to them as classes composed by properties.
- Relations: they model a type of interaction between domain concepts.
- Functions: a special kind of relation.
- Restrictions: rules that may constrain the domain of values for some concepts and relations
- Instances: constitute the concrete items/individuals represented by the ontology
- Axioms

Ontologies
elements

- **Concepts**
- **Relations** **R: C_1 x C_2 x …x C_n**

Subclass-of: $Concept_1$ x $Concept_2$
Connected to: Component1 x Component2

- **Functions** **F: C_1 x C_2 x …x C_{n-1}---> C_n**

Cousins: Person x Person → {TRUE,FALSE}
Object-price: Value+Gain+TAX →Price

- **Restrictions** ***e.g.*., data type, cardinality…**
- **Instances** **concrete items/individuals**
- **Axioms** **rules that are always true**

Ontologies
Design and development

- **Creating an ontology requires**
  - To define the classes in the domain
  - To organize the classes in a taxonomic hierarchy
  - To define each class' properties and include any restriction on their values
  - To assign values for each property to create instances.

Ontologies
Design and development

- **It is important to note that:**
  - There is no single standard methodology to develop ontologies
  - There is no single correct method to model a domain. Best solution depends on given application/domain.
  - The development process tends to be iterative
  - The objects in the ontology should be close to the objects and relations used to describe the domain
    - Usually they correspond to substantives and verbs which appear in sentences describing the domain

Phases in Ontology Development

- **In most methodologies the following 7 phases are present**
  - Phase 1: Determine the domain and coverage for the ontology
  - Phase 2: Consider to re-use existing ontologies
  - Phase 3: Enumerate the important terms in the ontology
  - Phase 4: Define the classes/concepts and their hierarchy
  - Phase 5: Define the attributes and relations for each class
  - Phase 6: Define the attributes' /relations characteristics
  - Phase 7: Create instances (optional)

Phases in Ontology Development

- **Phase 1: Determine the domain and coverage for the ontology**
  - Which is the domain to be covered by the ontology?
  - What will we use the ontology for?
  - Which kind of questions the ontology should answer?
  - Who will use and maintain the ontology?

- **Phase 2: Consider to re-use existing ontologies**

  - Ontologies are built to communicate Knowledge in a given domain. Therefore they are built to be shared.
  - It is unnecessary to re-do work already done by someone else: if there is already a (good) ontology for our domain we can use it.

Phases in Ontology Development

- **Phase 3: Enumerate the important terms in the ontology**
  - Write down a list of the terms useful to refer to our domain, by creating sentences we could use to ask things about it or to explain/describe the domain to others
    - Which properties do these things/terms have?
    - What would we like to say about them?

Phases in Ontology Development

- **Phase 4: Define the classes/concepts and their hierarchy**

- There are several approaches

  - *Top-down*: we define the most general concepts and then we refine them by a specialization process

  - *Bottom-up*: we defin the most specific cconcepts and then we group them according to comon properties, in a generalization process

  - *Bi-directional*: we define the most relevant cocepts and then we generalize / specialize to complete the ontology

- None of these methods is the best one, it depends on the domain and the target granularity of our model.

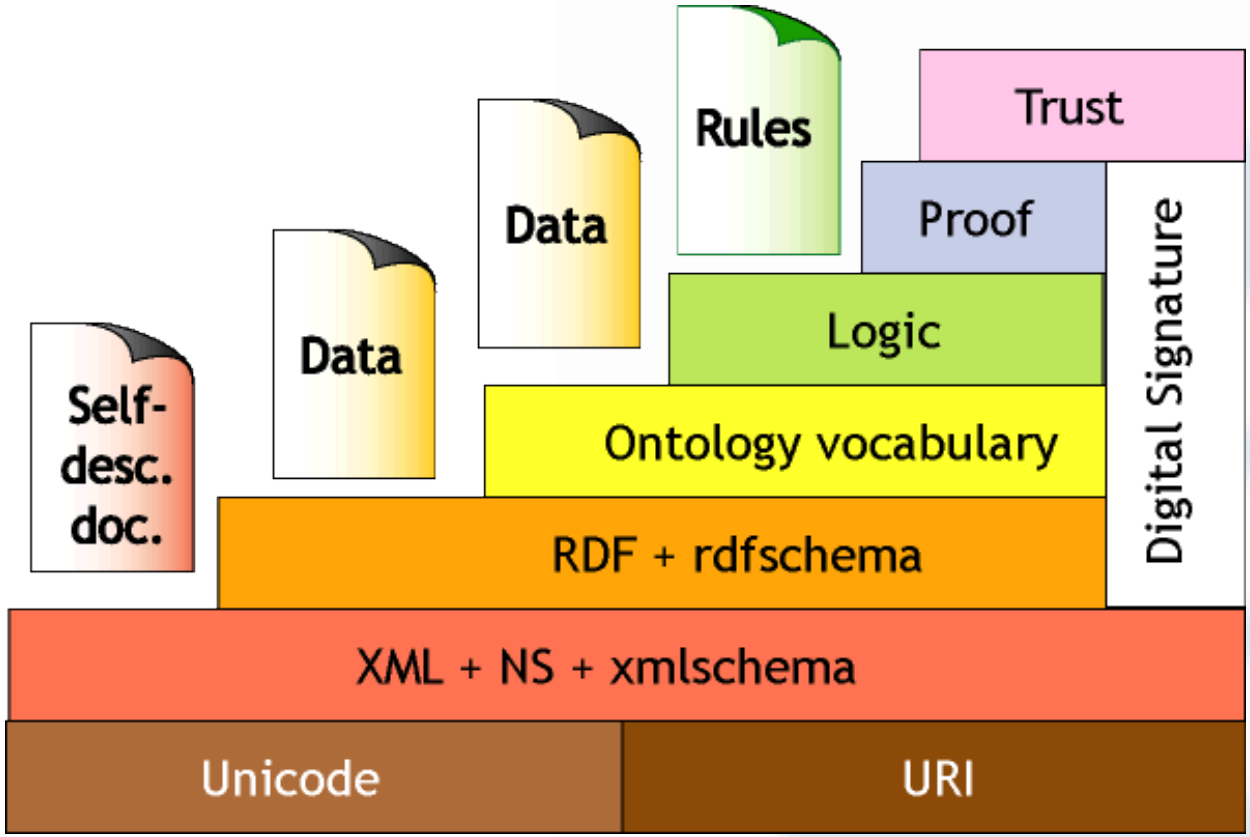
Phases in Ontology Development

- **Phase 5: Define the attributes and relations for each class**
  - Should describe the internal structure of our classes
  - Define a list of characteristics and to which class do they correspond
  - We may have different property types
    - Descriptive properties, *e.g.* quality
    - Identifying properties, *e.g.* names
    - Parts
    - Relations with other class instances
  - Properties should be assigned to the most general class, the subclasses will then inherit these properties

Phases in Ontology Development

- **Phase 6: Define the attributes'/relations' characteristics**
  - For attributes:
    - Cardinality (number of allowed values)
    - Type, value domains
    - Default values
    - Mandatory/optional
  - For relations
    - Cardinality
    - Range
- **Phase 7: Create instances**
  - If needed, create instances which will be part of the ontology at all times.

Semantic Web Wedding Cake



Ontology development principles

- **Clarity and Objectivity.**
  - An ontology must provide the user with the meaning of the term defined objectively and in a close-to natural language representation
- **Completeness.**
  - The definitions must be expressed in terms of necessary and sufficient.
- **Consistency.**
  - To ensure (allow) the inferences derived from the ontology to be consistent with definitions

Ontology development principles

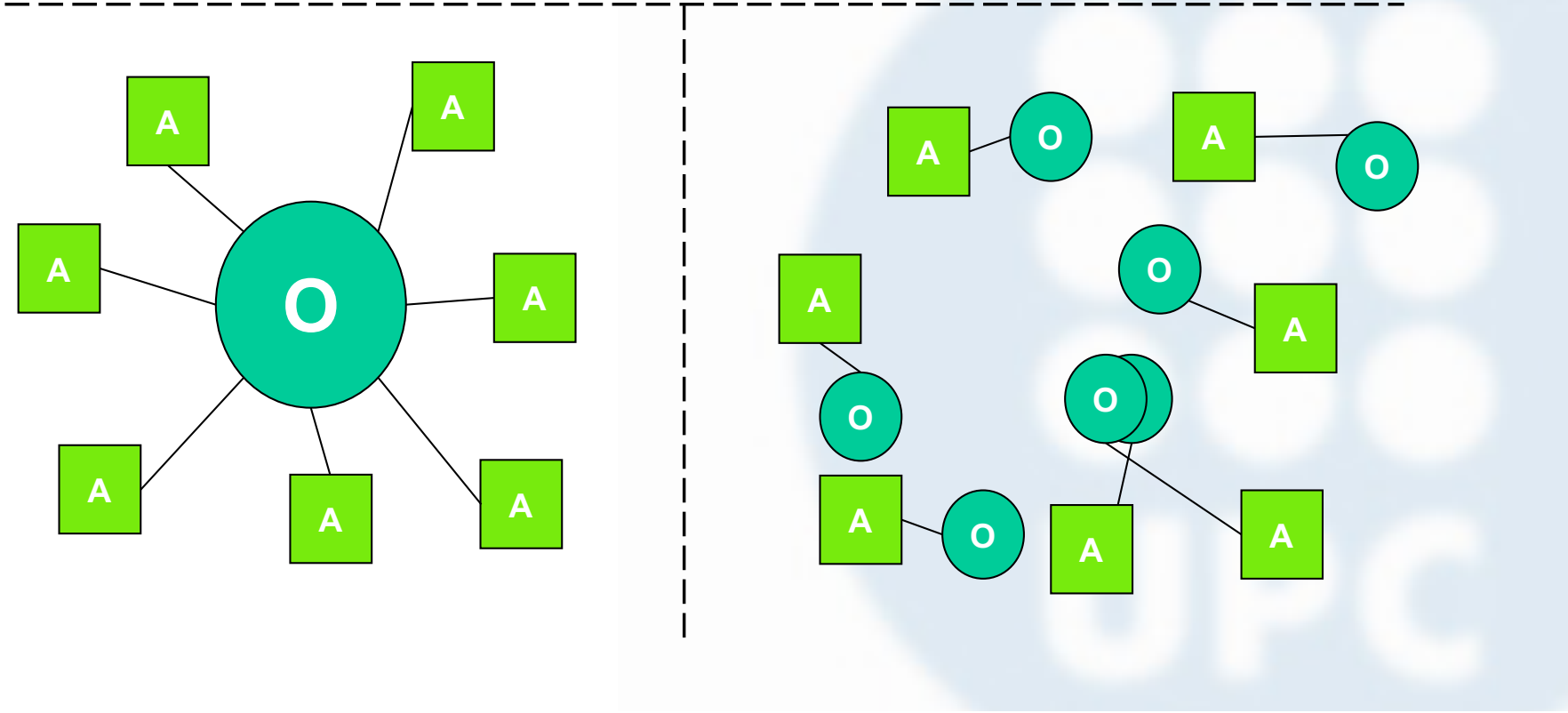
- **Maximum monotone extensibility.**
  - New general or specialized terms can be included in the ontology in such a way that it does not require the revision of existing definitions.
- **Principle of ontological distinction.**
  - The classes in an ontology must be disjoint.
- **Diversification**
  - Diversification of the included hierarchies to increase the power of multiple inheritance mechanisms.

Ontology development principles

- **Modularity**
  - To reduce the coupling between modules.
- **Name standardization**
  - Standardization of names where possible
- **Minimizing semantic distance**
  - Minimizing the semantic distance between closely related concepts. Similar concepts will be grouped together and represented using the same primitives.

Size and scope of an ontology

- Two extremes (the reality something in between):
  - One huge ontology O that captures *everything*
  - One (small) ontology for each specific application A



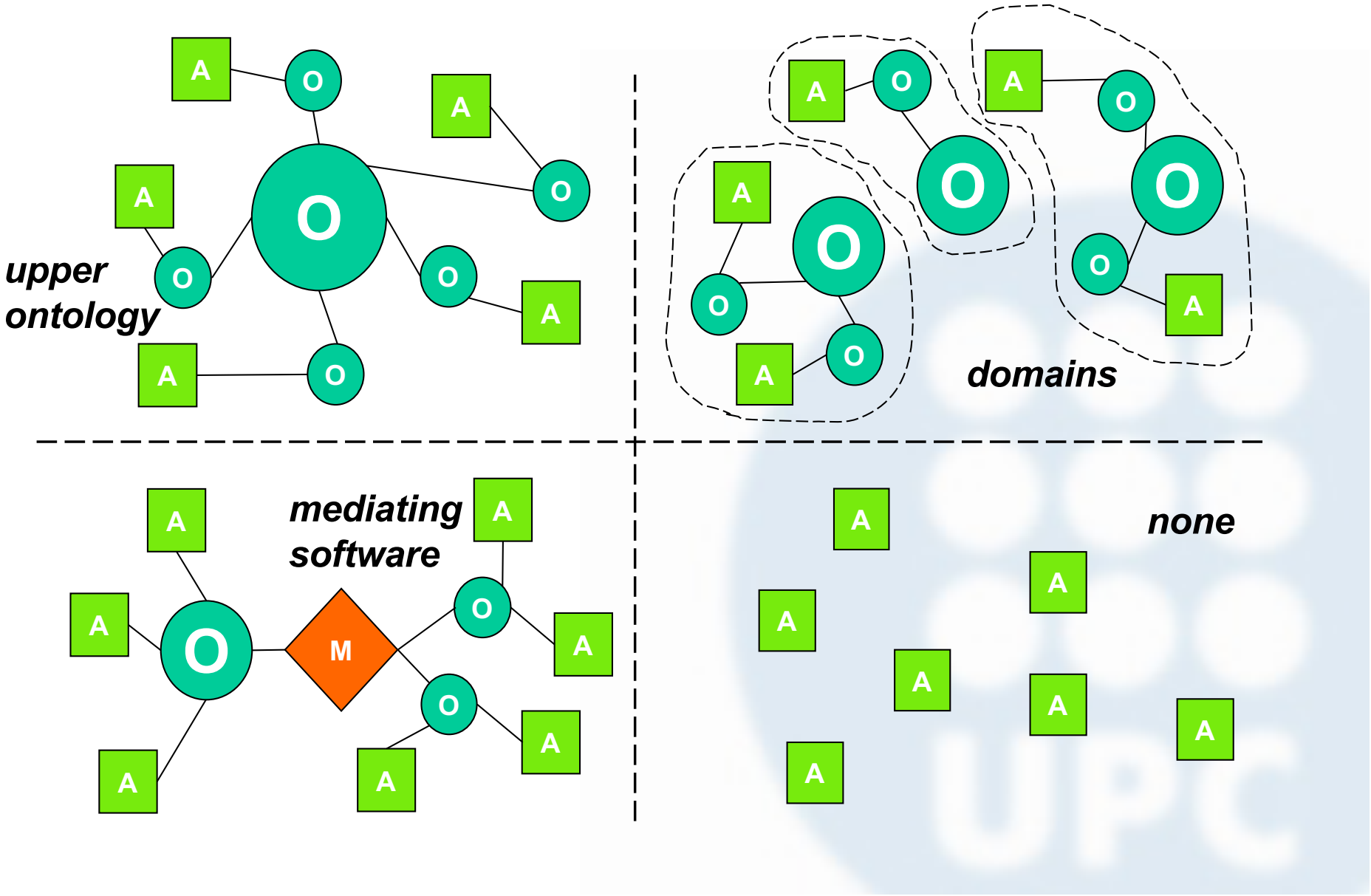
"One large ontology" approach

- Benefits
  - few or no internal inconsistencies
  - for an application developer easier to find
  - uniform documentation
  - less overlapping work!
- Drawbacks
  - who maintains it?
  - who is responsible?
  - heavy and slow to use (both for human user and for application)
  - difficult to take into account everybody's opinions and wishes at design time and when updating
- Example: Cyc

"Several small ontologies" approach

- Benefits
  - ontologies fit well the application demands
  - faster to use
  - easier to form complete picture of an ontology (fewer concepts and interrelations)
- Drawbacks
  - different ontologies do not fit together without either
    - central coordination body or
    - ontology alignment software 
    - coincidence
  - overlapping work - same concepts defined in multiple ontologies, either in the same way or (even worse!) differently

Some mixed / other approaches



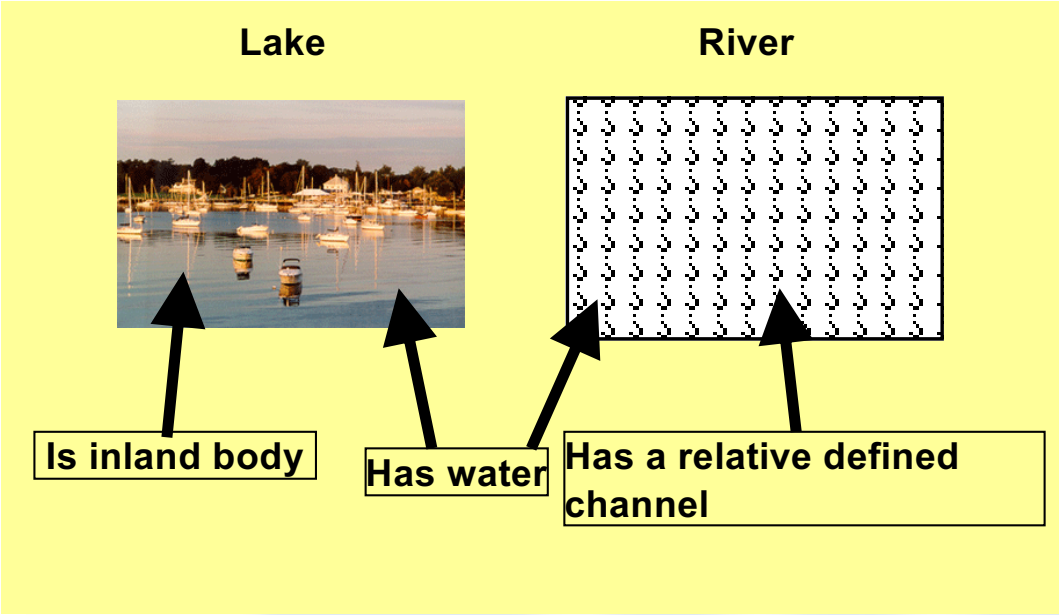
Ontologies

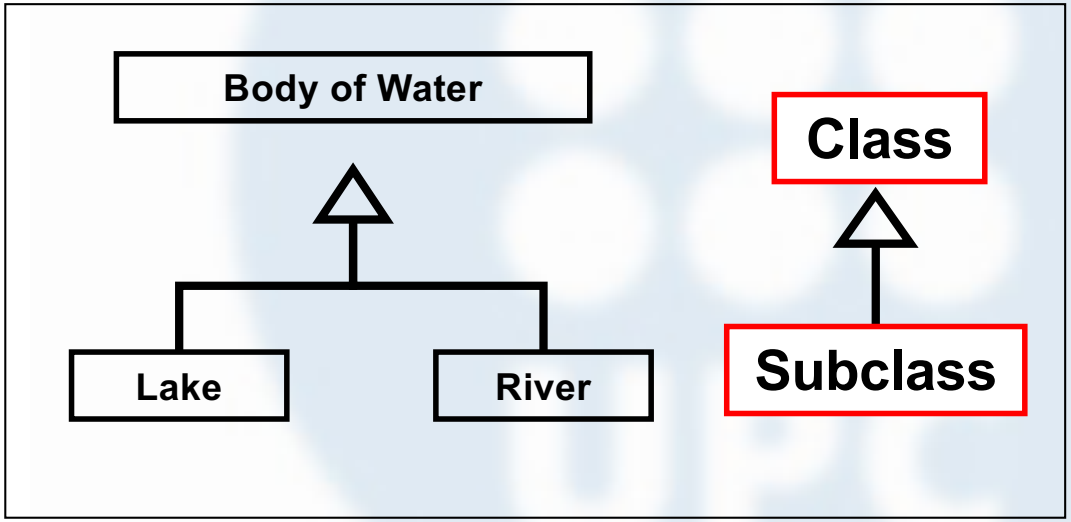
- -substrate
- -entity
  - -physical thing
    - -object(Continuant)
      - -Topological whole
      - -morphological thing
      - -functional
        - -biological
        - -artificial
      - -agent
        - -single
          - -animal
            - -human
          - -robot
        - -complex
          - -organization
            - -corporation
      - -society
    - -process(Occurrent)
      - -active
        - -Action
          - -physical action
            - -achievement act(sit)
            - -continuing act(sing)
            - -meta-act(begin)
          - -cognitive act
        - -phenomena
          - -burning, rain fall
      - -stative
        - -state
          - -internal state(hunger)
          - -external state(singing)
        - -state of affairs
  - -abstract thing
    - -belief
    - -knowledge
    - -purpose
    - -value
      - -function
      - -number
        - -real
        - -proportional
        - -ordinal
        - -categorical
- -quality
  - -intrinsic attribute
    - -mass
    - -length
    - -color, etc.
  - -accidental attribute
    - -possessions
  - -property
    - -red
  - -unit
    - -length unit
      - -cm, m, km, etc.
    - -weight unit, etc.
- -quantity
- -role
  - -intrinsic role
    - -part-whole role
      - -husband
    - -relation-role
      - -husband
      - -friend
    - -action role
      - -agent
      - -object
      - -means role
    - -vocational role
      - -nurse, etc.
    - -others
      - -material
      - -input, etc.
  - -pseudo role
    - -Agent
      - -rotating object
      - -flying object, etc.
- -relation
  - -topological relation
    - -structure
    - -right and left
  - -basic relation
    - -is-a
    - -part-of
  - -temporal relation, etc.
- -proposition
  - -product proposition
    - -novel
    - -poem
    - -etc.
  - -design proposition
    - -procedure
    - -music
    - -drama
    - -specification
- -representation
  - p/o “form”:rep. form
  - p/o “content”:proposition
  - -representation by symbol
    - -musical score
    - -sentence
  - -representation by image
- -representation form
  - -language
    - -spoken language
      - -utterance
    - -written language
      - -hand-written
  - -still image
    - -photo
    - -hand-written image
    - -figure, etc.
  - -motion image

Ontologies - Computer Science
Specification of conceptualizations

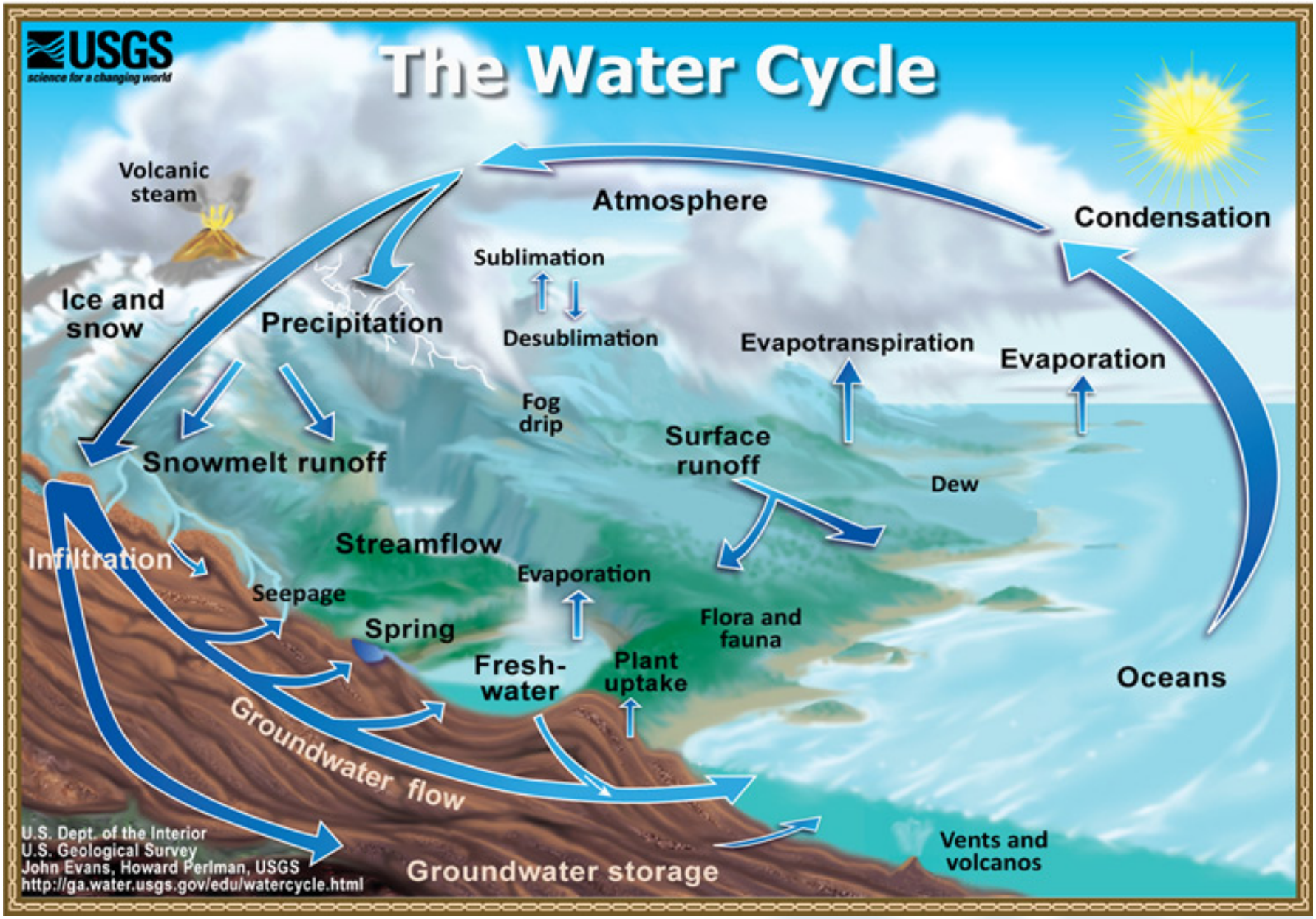
Example:

1. Properties of real world objects are identified.
2. Similarities are identified.
3. Concepts are created
4. and are expressed as a class.
5. Classes are related.





Hydro Cycle



Ontologies

Ontology:
What *exactly* **is** the “lake”?

- Is a lake a large body of water?
- **SDTS:** Lake: “***Any standing body of inland water***”
- Or is it a (natural) basin that can hold a large amount of water?
- Or is it a large area covered by water?
- If the water goes away, is the lake dry, or does the lake cease to exist?
- If the water rises, what is lake and what is flood?
- As water flows through it, where does the lake begin and end?

What *exactly* **is** the "river"?

- Is a river a large stream of water?
- **SDTS:** Watercourse: "***A way or course through which water may or does flow***"
- If the water goes away, is the river dry, or does the river cease to exist?
- If the water rises, what is river and what is flood?
- Where does surface flow stop and river begin?
- Where does river stop / start and lake start / stop
- How many streams is a braided stream?

Water and Voids

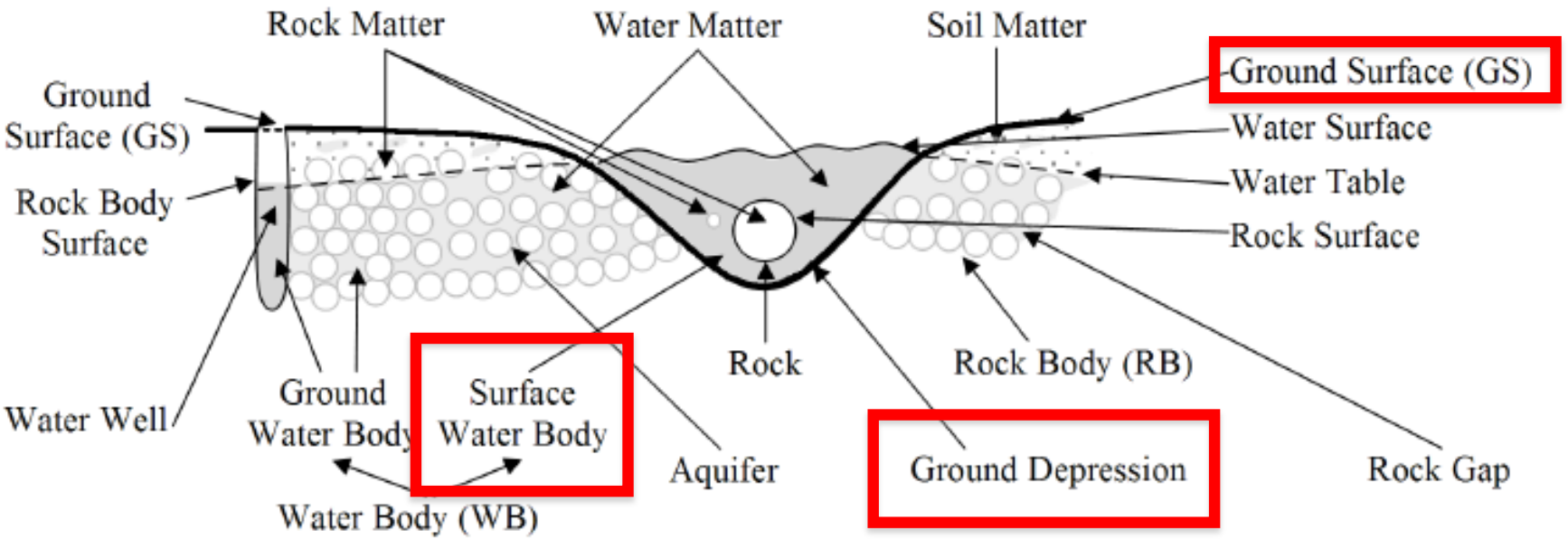
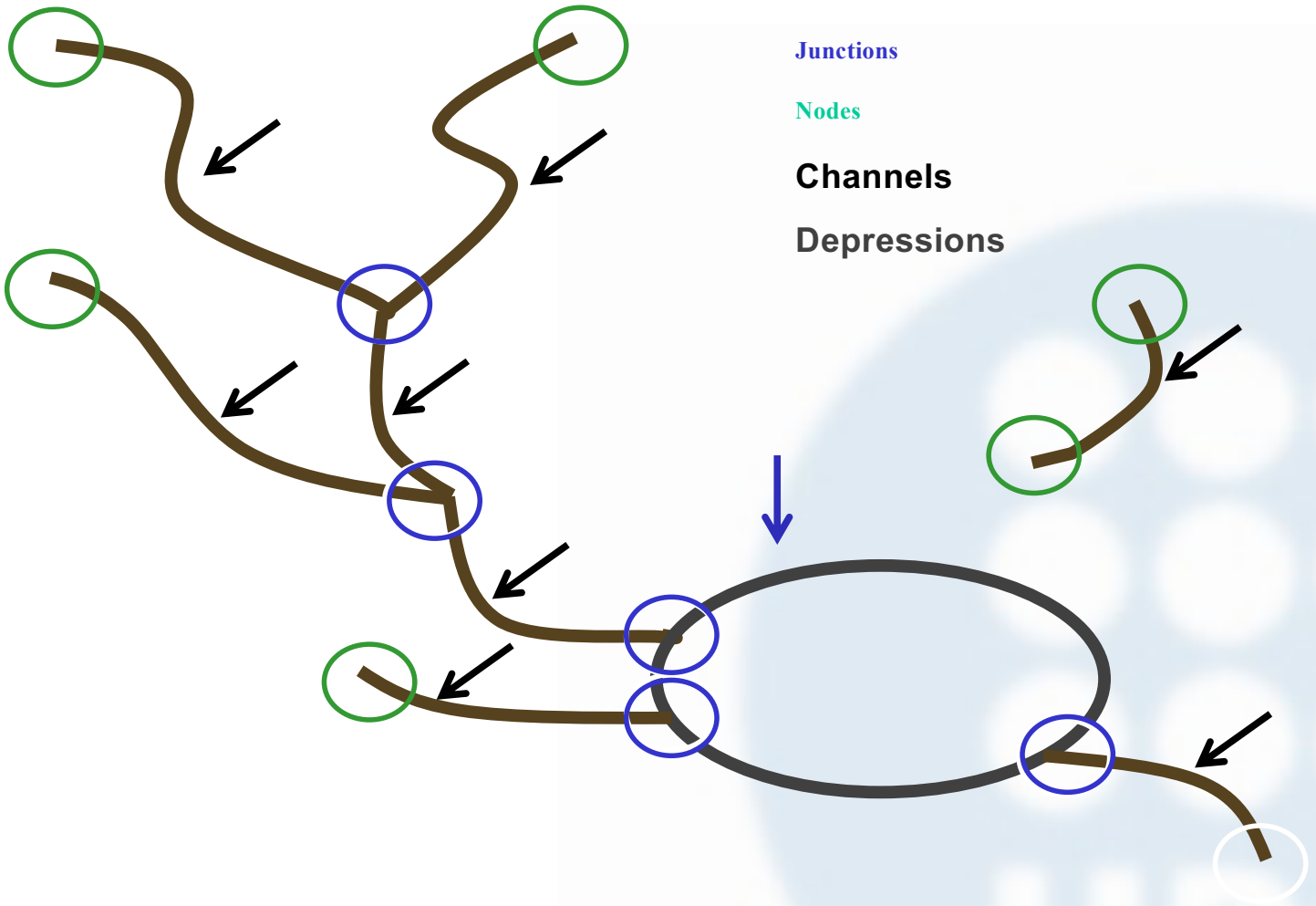


Figure 1. Examples of physical objects (rock body, water body, aquifer, rock) and physical voids (water well, ground depression, gaps between rocks) in hydrogeology. Different kinds of matter are shown at the top, and various features are illustrated on the right (ground surface, water surface, rock surface, water table).

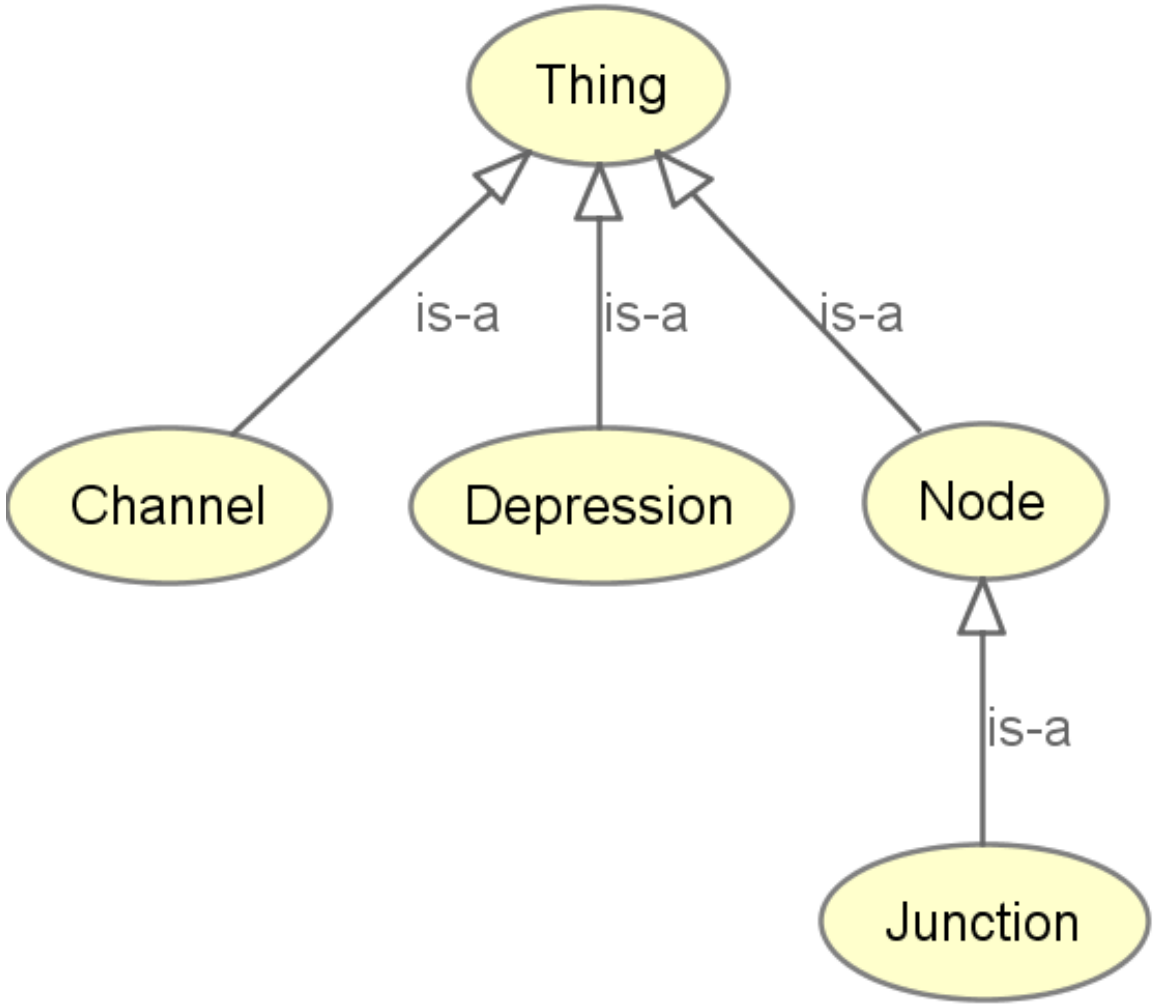
Wet vs Dry

- We divided into two models:
  - "*Dry*" model is terrain features that **may** contain water
  - "*Wet*" model is where water is currently flowing and standing
- Initial models are temporal snapshots
- Ignored erosional causation, groundwater leakance, *etc*.
- Do not necessarily correspond directly to named or topographic features

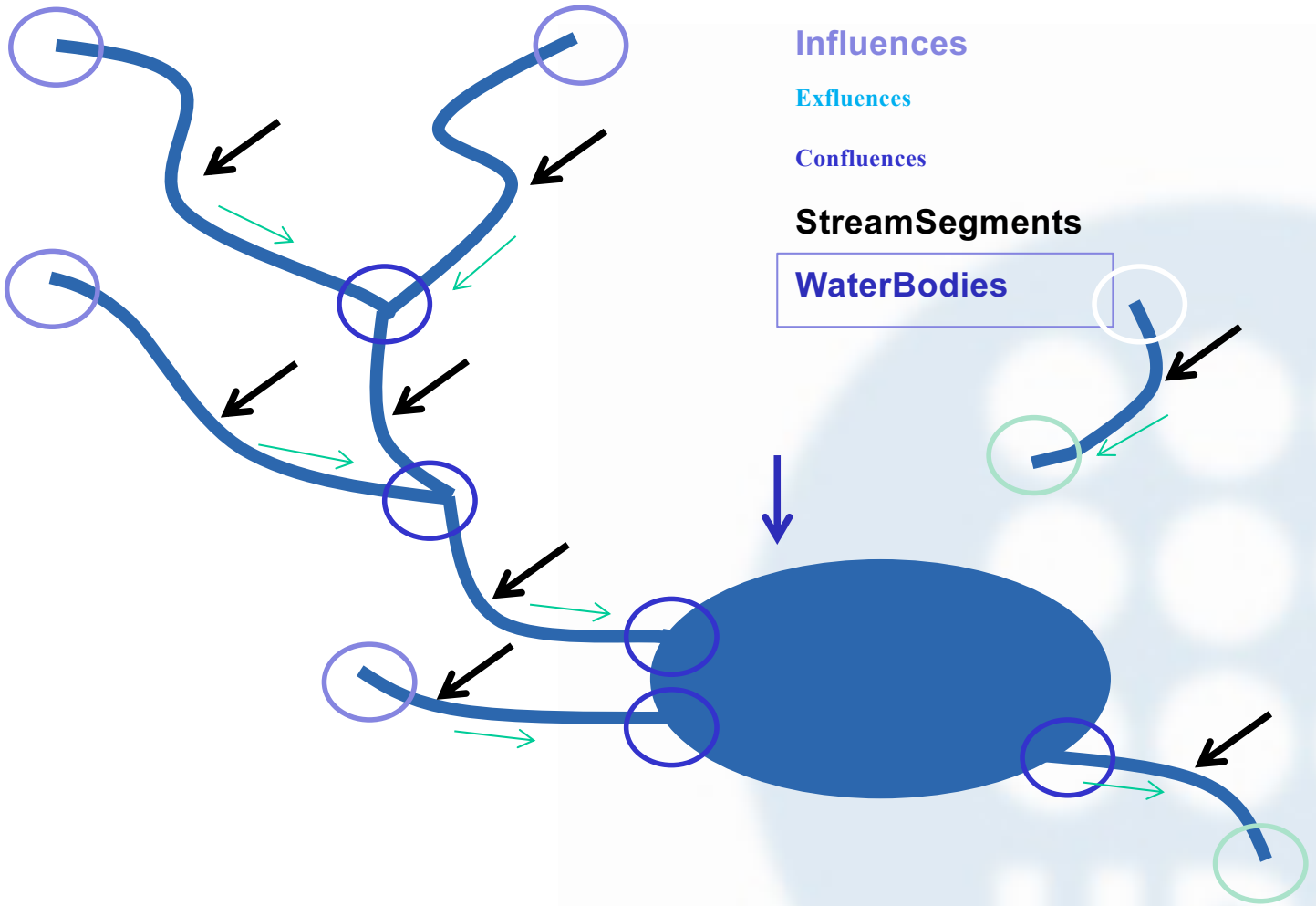
Dry Model



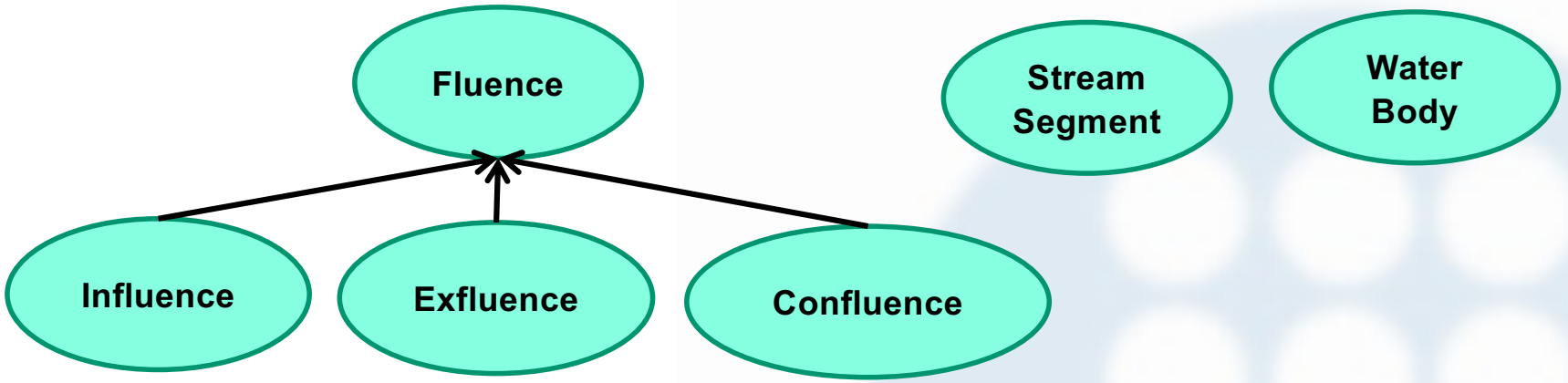
Dry Model Classes



Wet Model

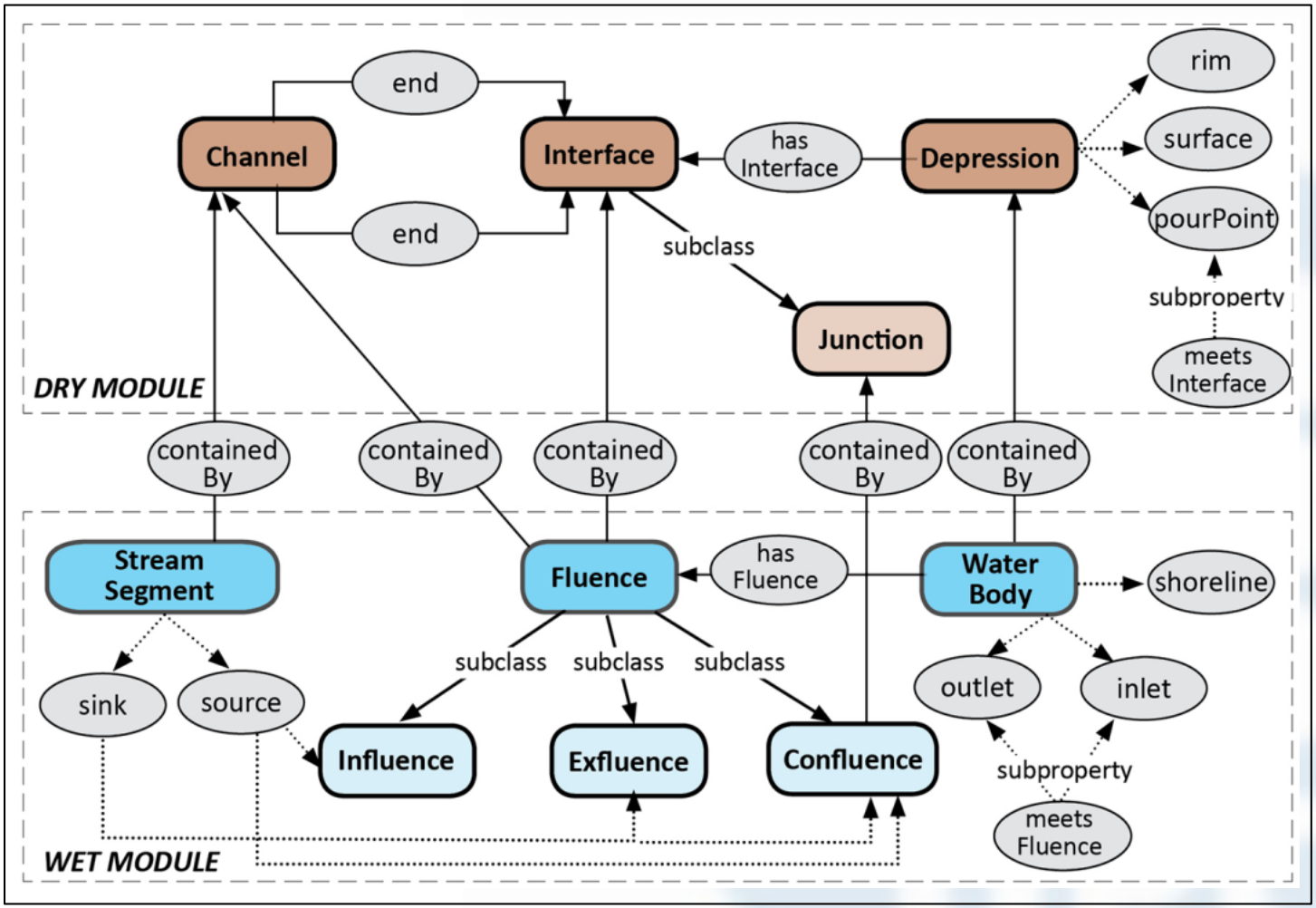

Influences
Exfluences
Confluences
StreamSegments
WaterBodies

Wet Model Classes



Surface Water Patterns

Fig. 1. Surface *Water* pattern's *Dry* and *Wet* module classes (brown/blue) and properties (grey).



Which of the approaches to use with agent systems (in the long run)?

Agent-systems are typically distributed systems

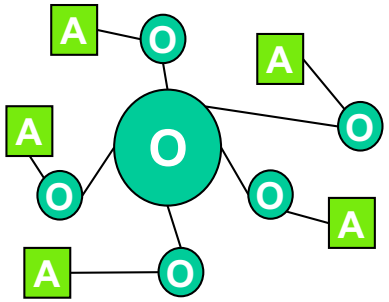
- agent-systems consisting of but one agent are rare and often not useful

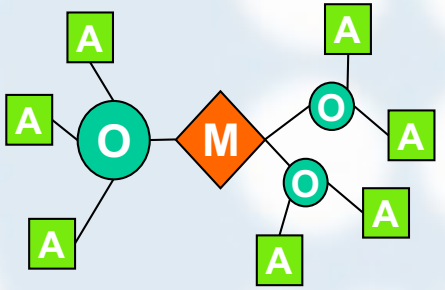
→ agents typically need to communicate with each other

→ agents should understand each other

People often design and implement agents independently, unaware of the other developers

- agents' understanding each other is (partly) based on ontologies





Some advices on ontology development

- Do not include singular and plural versions of a word (the best policy is to only use names in singular or plural).
- Names are not classes
  - We must distinguish the class name which we give, we can have synonyms, but all represent the same class.
- Make sure that the hierarchy is successfully built.
- See transitivity relations and see if they are correct.
- Avoid cycles in the hierarchy.

Some advices on ontology development

- All subclasses of a class should be at the same level of generality.
- There is no criteria regarding the number of classes, experience is that a number between two and twelve, but you would suggest that we need to structure classes by adding more levels.
  - When to introduce new classes?
- Often uncomfortable to browse hierarchies or too flat or very deep, you should choose a midway point, some indications would be.
  - New classes have additional properties that do not have the superclass.
  - They have different restrictions. Participate in different relations.

Some advices on ontology development

- Decide if we have to use a property or create a class.
  - Sometimes an attribute is important enough to consider that its different values correspond to different objects
- Decide where is the level for instances.
- Think what minimum level of granularity you need.
- Limit the scope of the ontology.
  - The ontology need not include every possible domain class, only the ones needed for the application being developed.
  - We also do not need to include all the attributes/restrictions/possible relationships

Formal foundations of Ontologies

Mereology

- Mereology is a collection of axiomatic first-order theories dealing with parts and their respective wholes.
- Its core notion is meronomic, which means based on part–whole relationships.
- It has been the formal basis of ontologies since Plato until beginning of 20th century

- *Ground mereology*: parthood is a reflexive partial order

x is_a part_of x

If x is_a part_of y and y is_a part_of x then x=y

If x is_a part_of y and y is_a part_of z then x is_a part_of z

Formal foundations of Ontologies

Mereology

- x is proper part of y: x is part of y and y is not part of x
- x overlaps y: there is a part of x that is also a part of y
- x and y are disjoint: x and y do not overlap
- *Binary Product*: x . y: individual that is part of both x and y and any common part of x and y is a part of the product.
- *Binary Sum*: x + y: individual that overlaps something iff it overlaps at least x or y.
- *Difference*: x – y: largest individual contained in x that has no part in common with y.

member-of ≠ instance-of

In other words, **set ≠ class**

A set which does not have its intensional description cannot be a class.

{car, 3, set, sushi} is not a concept, and hence not a class
{human-1, human-2, …(all humans)} can have two interpretations:
a **class** Human when human-i is interpreted as an instance of Human.
an **instance** of a class "human-set" whose **members** are human-i.
{tree-1, tree-2, …, tree-n} → a forest*(member-of)*.
{tree-1, tree-2, …, tree-i, …(all trees)} → a class Tree*(instance-of)*.

The ontological definition of a class and instance

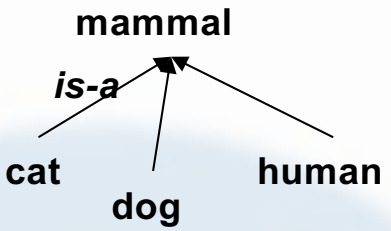
A thing which is a conceptualization of a set X can be a class if and only if each element x of X belongs to the class X if and only if the "intrinsic property" of x satisfies the intensional condition of X. And, then and only then, <x *instance-of* X> holds

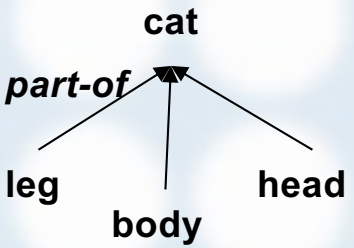
Is-a *vs.* part-of

The major difference between *is-a* and *part-of:*
when making an instance of a class:

***is-a* case**:
Only one instance of one of the subclasses are generated

***part-of* case**:
Instances from each of its subclasses are generated





cat *part-of* mammal? dog *part-of* mammal?
If we interpret them as ***species***, it is correct.

Formal foundations of Ontologies

Description Logics

- Need to find a computable representation
- First Order Logic (FOL) is too expressive and computationally expensive
- Description Logics is a valid option
  - It allows to reason about classes, subclasses, instances and definitions
  - Is a restriction from FOL based on Set Theory
  - Is also the foundation of Object Oriented formal semantics
  - There are efficient algorithms based on analytic tableaux

Formal foundations of Ontologies

Description Logics

- FOL + new operators and symbols
  - $\doteq$ (if and only if), $\sqsubseteq$ (if)
  - $\sqcup$ union, $\sqcap$ intersection
  - $\top$ (universal set, theorem), $\bot$ (empty set, contradiction)
- Distinction between two kinds of predicates
  - Concepts (C)
  - Relations (R)
- Quantified formulae are rewritten:

$$\exists R.C \equiv \exists y(R(x,y) \wedge C(y))$$
$$\forall R.C \equiv \forall y(R(x,y) \rightarrow C(y))$$

Ontologies
Languages

- Need to express ontologies in a machine-computable language (usable by computational entities in their messages and in their reasoning)
  - A language simple enough to make ontology development easier
  - A language with formal semantics
    - Formal semantics are needed in order to obtain deductions from the information in the ontology
  - A language allowing agents and other intelligent systems to reason with it
  - The computational cost should be reasonable

Advantages of Ontologies (1)

Formal Community View

Make it possible to formalise a shared viewpoint over a certain universe of discourse

e.g., agreement on how to model time

Interoperability

Can support communication and cooperation between systems developed at different sites

The ontological commitments made by a system are made explicit

e.g., diagnostic and therapy-control medical systems may share the same underlying generic medical ontology

e.g., notion of pathological state, therapeutic procedure

Advantages of Ontologies (2)

Model-based knowledge acquisition

use the medical guideline ontology to acquire knowledge about particular medical guidelines in a structured way

Knowledge-level validation and verification , use the medical guideline ontology to check guideline documents

KIF

Knowledge Interchange Format

- Developed at Stanford University (1992)
- Idea: to have an exchange format between applications, independent from their internal representations.
- Based in First Order Logic (FOL)
  - Prefix notation + definitions
- Semantics: Description Logics (Definitions + needed conditions)

KIF

- KIF has FOL's operators
  - Boolean values: `true`, `false`
  - Connectives:
    - `and`, `or`, `not`,
    - => (if) <= (only if), <=> (definition)
  - Quantifiers: `forall`, `exists`
  - Vars: `?x` (individual var) `@x` (var group, as in PROLOG)
- *e.g.*, `(forall (?x)(> ?x 3))`
- Lists can be built and used as basic data types (as LISP)

```
(listof 3 ‘‘HOLA’’ 2.34)
```

KIF

- **Functions** can be defined

```
(deffunction abs (?x) := ((if (>= ?x 0) ?x (- ?x))))
```

- **Relations** can also be defined

```
(defrelation number (?x) :=
   (or (integer ?x) (real ?x) (complex ?x)))
≡
(<=> (number ?x)
   (or (integer ?x) (real ?x) (complex ?x)))
```

- **Metaknowledge** expressions

```
(believes john '(exists (?x) (> ?x 3)))
```

KIF
example

- **Class person**

```
(defrelation name (?x) := (string ?x))

(defrelation age (?x) := (integer ?x))

(defrelation person (?x ?y) :=
   (listof (name ?x) (age ?y))

(defobject juan:= (person "Juan" 25))

(defrelation adult (?x) :=
   (and (= ?x (person ?x ?y))
        (> ?y 18)))
```

Markup Languages: XML

- Idea of a Semantic Web:
  - Information semantically annotated in a machine-parseable language
- HTML is not enough
  - Language oriented to presentation
- Idea: to use XML (derived from SGML)
- Advantages
  - allows to describe attributes in information
  - already used by industrial initiatives
  - allows integration from different data sources (by means of XSLT translation rules)
  - Non-proprietary language

XML

- An XML document can contain Data Type Definitions inside or can refer to a DTD file
- One can create repositories of reusable definitions (namespaces)
- *e.g.*:

```
<!Element direction (name, place)>
<!Element place (street, city)>
<!Element name (#PCDATA)>
...
<direction>
  <name> John </name>
  <place>
    <street> Oxford St. </street>
    <city> London </city>
  </place>
</direction>
```

From XML to DAML+OIL

- **Problems:**
  - XML is too rigid (tree-like structures)
  - Difficult to include relationships to the structures defined
  - Difficult to assert predicates
- **Extension: RDF + RDFS**
  - RDF allows to assert statements
  - RDFS declares classes, attributes and relations
  - RDFS definitions can be instantiated
- **Even more powerful extension: DAML+OIL**
  - DARPA agent markup language
  - Ontology Inference Layer

DAML+OIL

example

```
<daml:Ontology>
<daml:Class rdf:ID="Person"/>

<daml:ObjectProperty rdf: ID="Name">
 <daml:domain rdf:Resource="http://..." />
</daml:ObjectProperty>

<daml:ObjectProperty rdf: ID="Parent_of">
 <daml:domain rdf:Resource="#Person" />
 <daml:range rdf:Resource="#Person" />
 <daml:Restriction daml:Cardinality="2" />
</daml: ObjectProperty>

<daml:ObjectProperty rdf: ID="Son_of">
 <daml:InverseOf rdf:Resource="#Parent_of" />
</daml: ObjectProperty>

</daml:Class>
</daml:Ontology>
```

DAML+OIL

example

```
<Person rdf:ID="John"/>
<rdfs:comment> John is Peter's father </rdfs:comment>
<Age> 38 </Age>
<Parent_of:Resource="#Peter" />
</Person>

<Person rdf:ID="Peter"/>
<Age> 12 </Age>
<Son_of:Resource="#John" />
</Person>
```

OWL

- Further extension: OWL (Ontology Web Language)
  - Fusion of DAML+OIL, standard of W3C
  - Has also become standard *de-facto* for most of Semantic Web applications and Semantic Web Services
  - 3 levels:
    - *OWL lite*: defines taxonomies and simple restrictions
    - *OWL DL*: provides expressiveness as Description Logic
    - *OWL full*: maximum expressiveness (but not available reasoners)
  - Specification available at
    - http://www.w3.org/TR/2004/REC-owl-features-20040210/#s2.1
  - There is a new specification being defined, OWL 2 (Oct 09).

- There is yet another extension, called OWL-S, used for Web services and the Semantic Web

OWL
Characteristics

- It integrates all elements in DAML+OIL, creating a new namespace: `owl`
  - All properties that DAML+OIL indlude have been integrated into the `owl` namespace
  - Sintax is extended to specify instances, and a new tag, `owl:Thing` , is added
    - (all instances belong to `owl:Thing`)
  - It allows the definition of all the primitive types included in the XML Schema Datatypes by using the `xsd` namespace.

OWL
Characteristics

- OWL adds new restrictions to classes, properties and individuals
  - `owl:allValuesFrom`: the property values should belong to certain class
  - `owl:equivalentClass`, `owl:equivalentProperty`: Equivalence between classes and properties
  - `owl:sameAs`, `owl:differentFrom`: equal/different Individuals
  - `owl:SymmetricProperty`: a symmetric property to another one

Public ontology repositories

- http://www.daml.org/ontologies/
  - 282 public Ontologies written in DAML+OIL/OWL
  - Wide variety of topics:
    - academic department, Actors, address book, airport, Bibliography, Biology, Chemistry, Clothing, Weather
- http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library
  - Is not a repository, but a list to some interesting ontologies (mostly OWL)

Ontologies and Knowledge Representation

- **In** Distributed AI, ontologies are the basis of knowledge exchange.
- But are Ontologies a Knowledge Representation?

- Knowledge representation is a multidisciplinary subject that applies theories and techniques from three other fields:

1. *Logic* provides the formal structure and rules of inference.
2. *Ontology* defines the kinds of things that exist in the application domain.
3. *Computation* supports the applications that distinguish knowledge representation from pure philosophy…

The role of ontologies in agent communication

Used by agents when trying to make sense of each other

Case of FIPA: Ontology is denoted by one parameter in an ACL message (along with other parameters: sender, receiver, content, reply-with, reply-by, in-reply-to, envelope, language, protocol, and conversation-id)

Example:

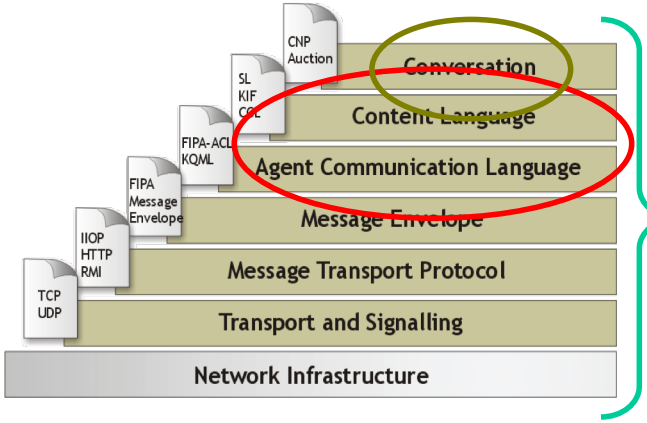
```
(cfp
   :sender (agent-identifier :name j)
   :receiver (set (agent-identifier :name i))
   :content
     "((action (agent-identifier :name i)
        (sell plum 50))
      (any ?x (and (= (price plum) ?x) (< ?x 10))))"
    ontology fruit-market
   :language fipa-sl)
```

http://www.fipa.org/specs/fipa00061/

http://www.fipa.org/specs/fipa00037/

The role of ontologies in agent communication (2.)

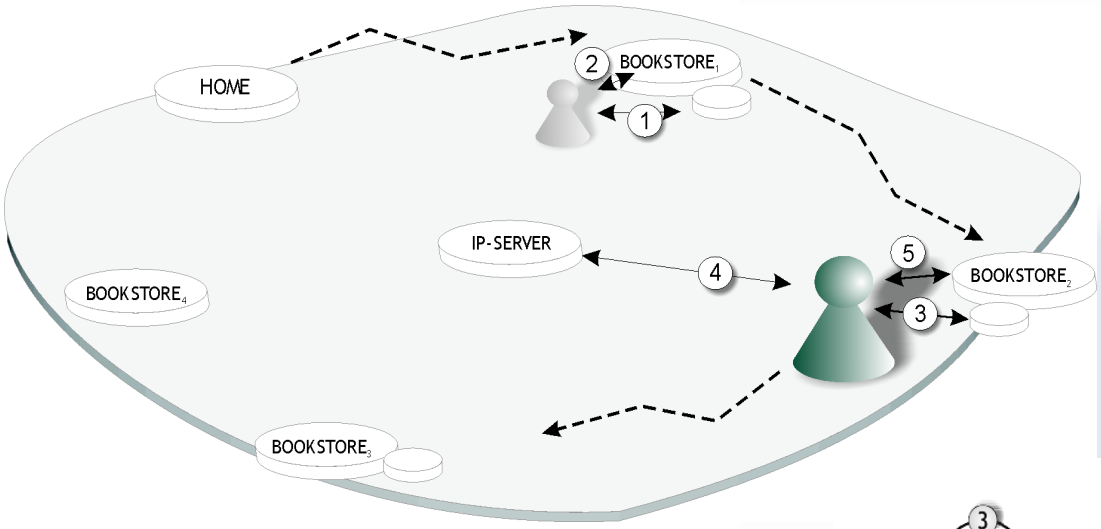
Ontology is referenced from within *an ACL message* and it determines the semantics of the concepts used in the *content language*

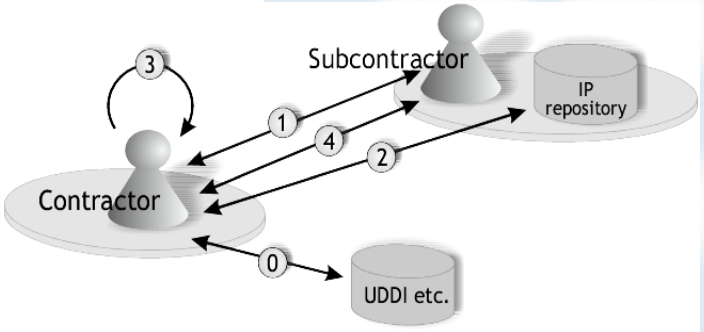


- However, ontologies themselves might concern whatever topics
- One interesting case would be to *ontologize* agent communication
- For example interaction protocols (Conversation layer) might be described in an ontology external to the agents

- The agents could download the interaction protocol descriptions and modify their behaviour accordingly
  - Ontology of actions or tasks vs. ontology of facts
  - ***Know-how*** vs. ***know-that***

The role of ontologies in agent communication (contd.)





Ontologies and Knowledge Representation

[…] Without logic, a knowledge representation is vague, with no criteria for determining whether statements are redundant or contradictory. Without ontology, the terms and symbols are ill-defined, confused, and confusing. And without computable models, the logic and ontology cannot be implemented in computer programs. Knowledge representation is the application of logic and ontology to the task of constructing computable models for some domain.

(Sowa (2000), *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks Cole Publishing Co., Pacific Grove, CA.)

Some ontology tools

Protégé (Ontology editor)

Lots of plugins: OKBC, RDF(S), OWL, UML, Prolog, Jess, OWL-S, etc.
http://protege.stanford.edu/, http://protege.stanford.edu/plugins/owl/

Jena

Java API for manipulating RDF, RDF(S), and OWL
http://www.hpl.hp.com/semweb/jena.htm, http://jena.sourceforge.net/

Pellet

Open-source Java-based OWL DL reasoner
http://www.mindswap.org/2003/pellet/

RDF Validator

For testing ontology code on RDF level:
http://www.w3.org/RDF/Validator/

Ontolingua

collaborative environment to browse, create, edit, modify, and use ontologies
http://www.ksl.stanford.edu/software/ontolingua/

References

[1.] **Luck, M., McBurney, P., Shehory, Onn, Willmott, S. "Agent Technology: Computing as interaction. A Roadmap to Agent Based Computing". Agentlink, 2005. ISBN 085432 845 9**

[2.] **D. Nardi, R. J. Brachman. "An Introduction to Description Logics". In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 5-44.**

[3.] **Haddadi, A. "Communication and Cooperation in Agent Systems: A Pragmatic Theory" Lecture Notes in Artificial Intelligence #1056. Springer-Verlag. 1996. ISBN 3-540-61044-8**

[4.] **Rosenschein, J. & Zlotkin, G. "Rules of Encounter. Designing Conventions for Automated Negotiation among Computers". MIT Press. 1994 ISBN 0-262-18159-2**

[5.] **N. F. Noy, D. L. McGuinness. "Ontology Development 101: A Guide to Creating Your First Ontology" Stanford University.**

6. **D. B Lenat, G. Miller and T. Yokoi, *CYC, WordNet, and EDR: critiques and responses*, Communications of the ACM, 1995.**

References (2)

[1.] **Neches, R; Fikes, R; *et al* "Enabling Technology for Knowledge Sharing" AI Magazine. Winter 1991. pp36-56**

[1.] **Gruber, T. "A translation approach to portable ontology specifications" Knowledge Acquisition. Vol. 5. 1993. 199-220.**

[1.] **Sowa, J. Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA. (2000).**

[1.] **Uschold, M & Jasper, R.. "A Framework for Understanding and Classifying Ontology Applications" Procc. Of the IJCAI-99 Workshop onOntologies and Problem-Solving Methods (KRR5)**

These slides are based mainly in W3C documents, [2], [3] and materials from J. Béjar.