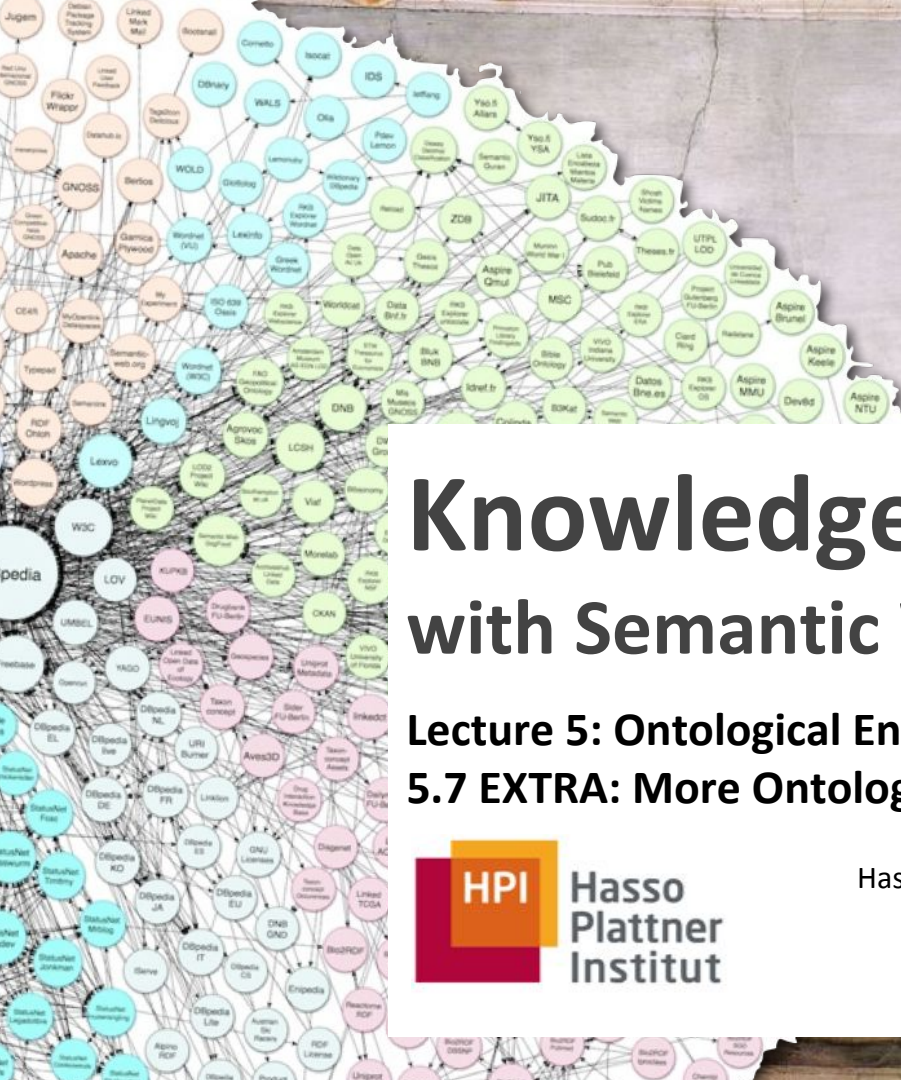




Knowledge Engineering with Semantic Web Technologies

Lecture 5: Ontological Engineering 5.7 EXTRA: More Ontology Evaluation



Dr. Harald Sack
Hasso Plattner Institute for IT Systems Engineering
University of Potsdam
Autumn 2015

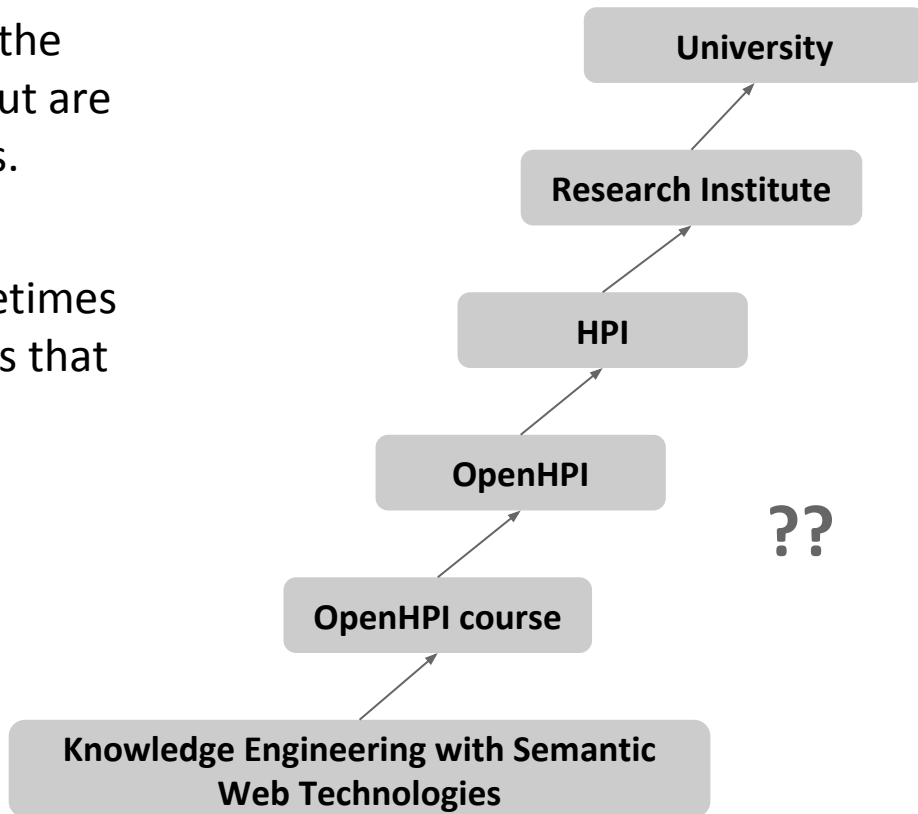




- Theoretical foundations of Ontology Evaluation already covered
- Practical Tool for Ontology Evaluation:
 - OntoClean

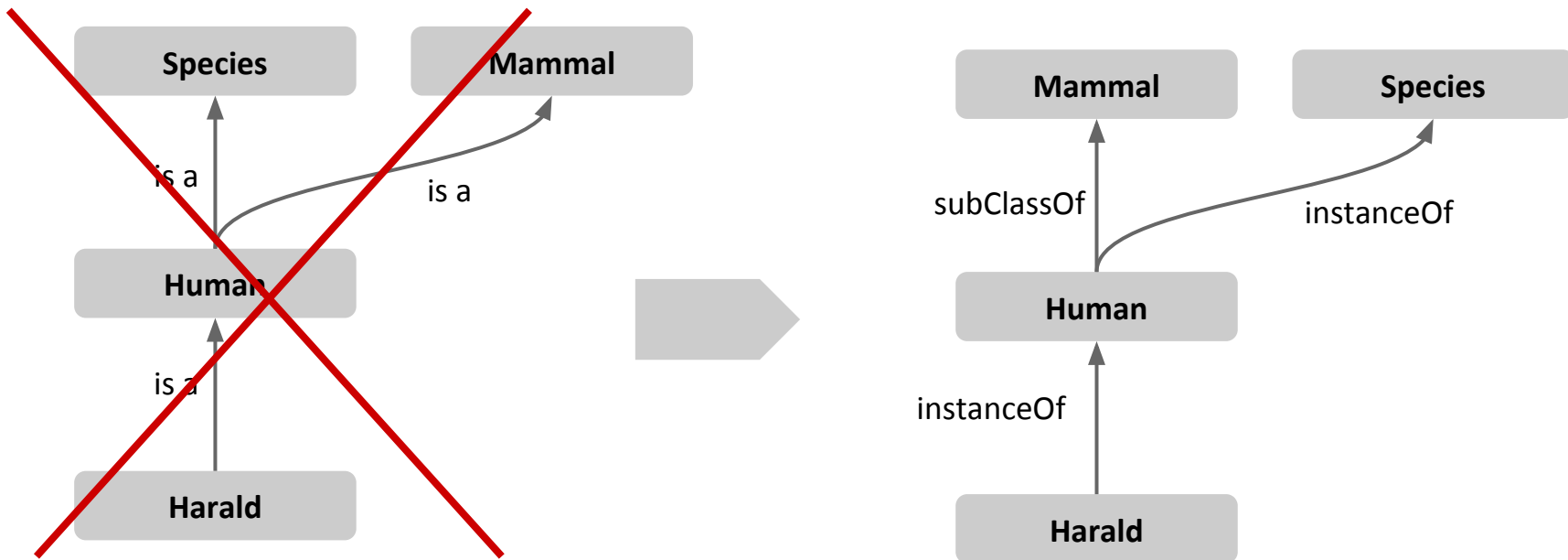
Correct Taxonomical Structures

- Experienced domain modellers can see the correct way to **structure a taxonomy**, but are typically unable to justify their decisions.
- **Problem:**
 - **Subsumption hierarchies** are sometimes misused, representing relationships that aren't really subsumption

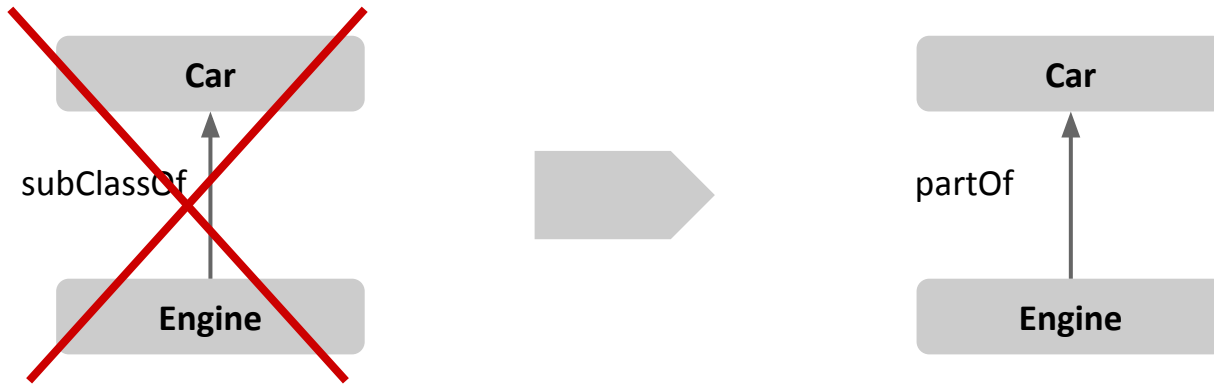


- **Subsumption** (also is-a relationship) is used to construct concept hierarchies
- Formal interpretation:
 - **A subsumes B ($B \sqsubseteq A$)**
if all instances of B are necessarily also instances of A
 - attributes, properties, characteristics of a superclass are inherited along the hierarchy to all subclasses
- Unfortunately, the subsumption relation is often misused or confused with other types of relationships

Subsumption confused with Instantiation

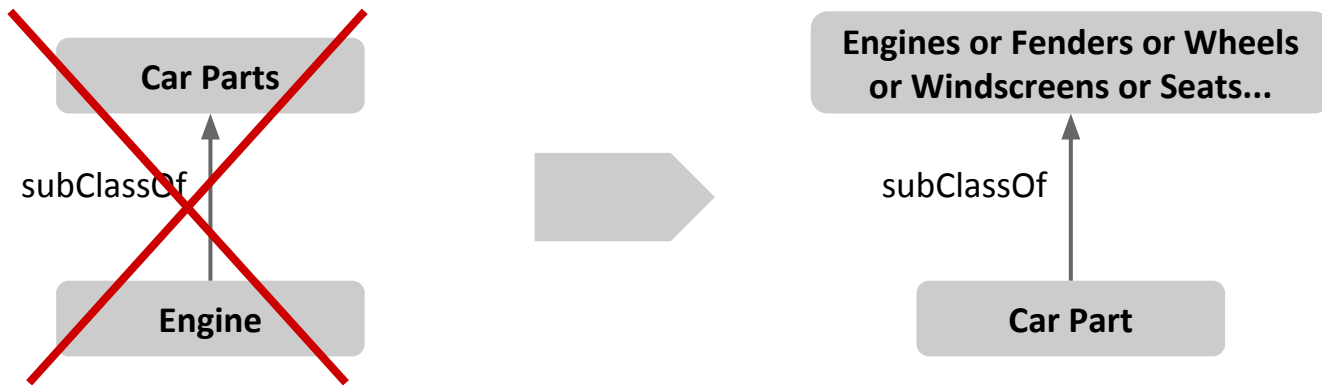


Subsumption confused with Part/Whole



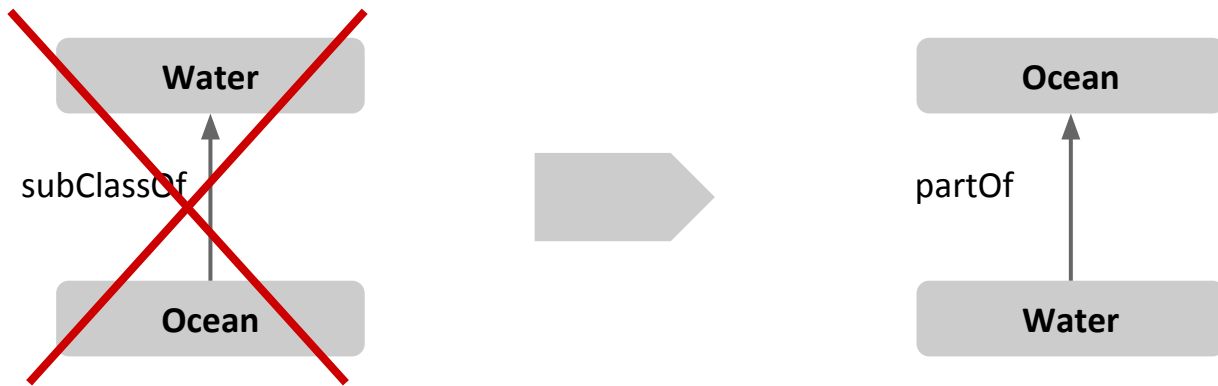
- some essential properties of **Car** are functional properties, as e.g. *being able to accommodate people*.
- **Engine** also has functional properties as essential properties, as e.g. *being able to crank and generate a rotational force*.
- But: **essential properties of cars do not apply to engines**
- Therefore: **Car cannot subsume Engine**

Subsumption confused with Disjunction



- not all **Engines** are really **Car Parts**, as e.g. a boat engine
- Vice versa, Car Parts are a subclass of the class of all Engines, Fenders, Windcreens, Doors, Seats, etc.

Subsumption confused with Constitution



- an instance of **Water** is an amount of water
- an instance of **Ocean** is, e.g., the Pacific Ocean
- **Oceans** are made up of amounts of water

- **OntoClean** for Ontology Validation
 - helps to expose common misuses of subsumption and
 - provides a formal basis for why they're wrong.
- **OntoClean** helps to expose
 - inappropriate and inconsistent modelling choices
 - implicit assumptions about the way in which the atomic properties or classes are being interpreted
 - ontological commitments being made by the modeller.
- **OntoClean** focuses on **taxonomy** and **subsumption**:
 - *A subsumes B iff for all x, x instance of B implies x instance of A*

- **OntoClean** applies a slightly different terminology
 - **General Properties** (corresponds with unary predicates in FOL), as e.g.
 - being an apple
 - being a table
 - being a person
 - ...
 - For each property we can associate a **Class**,
i.e. a set of entities which expose this property in a possible world
 - Members of the Class are **Instances** of the property.
 - NOT to be confused with OWL properties!

- **OntoClean** applies a slightly different terminology (cont.)
 - **Metaproperties** describe particular characteristics of the properties
 - Essence
 - Rigidity
 - Identity
 - Unity
 - Dependency
 - Associating metaproperties with properties (i.e. to ontology classes) helps characterize aspects of the properties.
 - **Constraints** on the metaproperties enforce restrictions on the taxonomy and help to highlight and evaluate the choices made

OntoClean - Essence and Rigidity

- A property is **essential** to an entity if it must hold for it.
 - Not just things that (accidentally) happen to be true all the time.
- A **rigid** property is a property that is essential to all of its instances.
 - Every entity that can exhibit the property must do so.
 - Every entity that is a *person* must be a *person*
 - There are no entities that can be a *person* but aren't.
- An **anti-rigid** property is one that is never essential
 - For example every instance of *student* isn't necessarily a *student* - *students* may cease to be *students* at some point without ceasing to exist or changing their identity
- A **semi-rigid** property is one that is essential to some instances but not to others

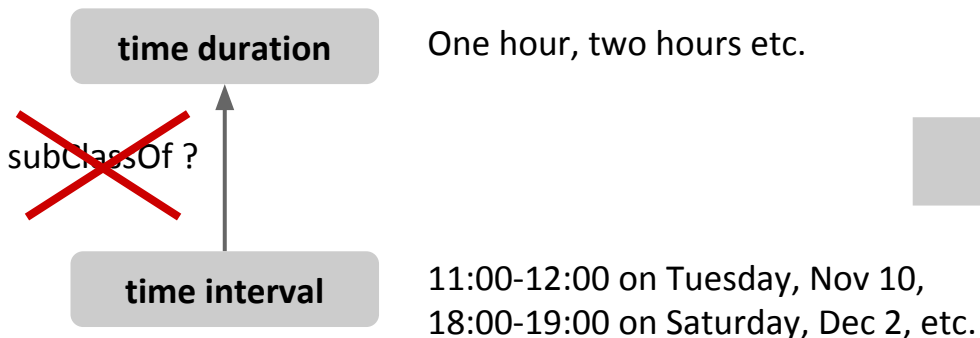
non
rigid

OntoClean - Rigid Properties

- **Human** is **rigid**
 - Every instance of human is necessarily a human.
 - An entity can't stop being a human and still be recognisably the same entity.
 - There are no entities that can be a human but aren't.
- **Student** is **not rigid**
 - Any student can cease to be a student while the entity is still the same.
- For **ontology evaluation**, all classes in the ontology should be labeled accordingly
 - **+R** for rigid, **~R** for anti-rigid, and **-R** for semi-rigid (not rigid)
 - meta properties pose constraints on subsumption relation, as e.g. a rigid class may never be subsumed by an anti-rigid class

OntoClean - Identity

- **Identity** criteria allow us to recognize individual entities in the world.
- **Identity** refers to the problem of being able to recognize individual entities in the world as being the same (or different).
- Consider the following example:



Identity

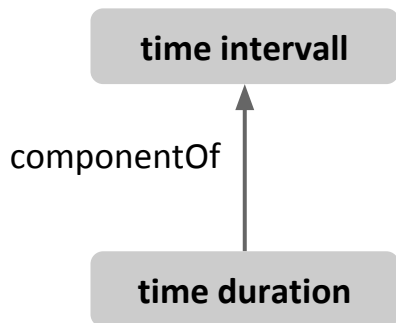
two durations of the same length are the same duration

But many **time intervals** may have the same **time duration**

two intervals at the same time and of same length are the same interval

OntoClean - Identity

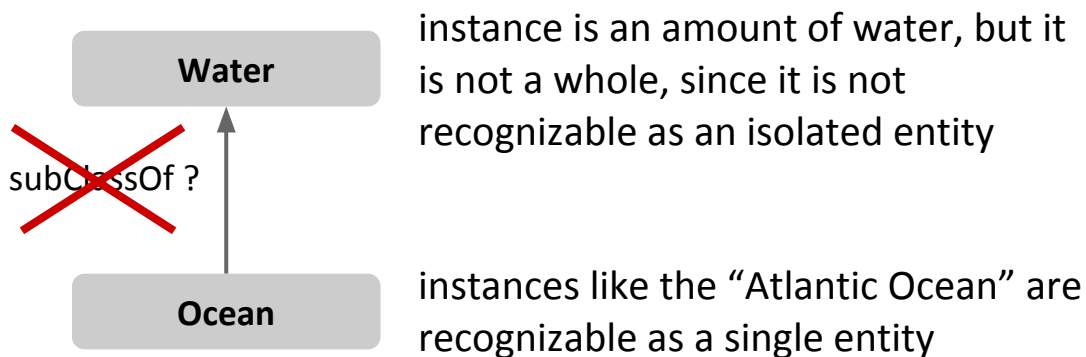
- “all time intervals are time durations” really means
“all time intervals have a time duration”
- duration is a component of an interval, but it is not the interval itself.



- For **ontology evaluation**, all classes in the ontology should be labeled accordingly
 - **+I** for carrying identity, **-I** for not carrying identity, and **+O** for carrying own identity (not inherited)
 - a property will inherit identity criteria from its parents (**+I**)
 - a property may also supply its own additional identity criteria (**+O**)

OntoClean - Unity

- **Unity** refers to the problem of describing the way the parts of an object are bound together.
- **Unity criteria** identify whether or not a property is intended to describe whole objects
- For some classes, all their instances are wholes (as e.g. Car), for others none of their instances are wholes (as e.g. Water).
- Consider the following example:

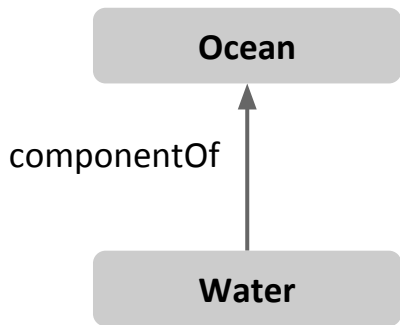


If we claim that instances of the latter are not wholes, and instances of the former always are, then this results in a **contradiction**.

- A property carries **common unity (+U)**
if all its instances exhibit a common unity criterion (e.g., Ocean)
- A property carries **no common unity (-U)**
if its instances are all wholes, but with different unity criteria
(e.g. Legal Entity, as it includes companies and people)
- A property carries **anti-unity (~U)**
if all of its instances are not necessarily whole

OntoClean - Unity

- “all Oceans are Water” really means
“Water is a component of Ocean”
- Problem arises from ambiguity of natural language



- For **ontology evaluation**, all classes in the ontology should be labeled accordingly
 - **+U** for carrying common unity, **-U** for not carrying common unity, and **~U** for carrying anti-unity
 - Thinking about concepts' identities, while building an ontology helps to avoid and to discover mistakes
 - Whole classes should never be subsumed by 'not-whole' classes

- A property X is **externally dependent** on a property Y if for all its instances, necessarily some Y must exist, which is neither a part nor a constituent of it.
 - e.g. Parent is externally dependent **(+D)** on Child - *one cannot be a parent without having a child.*
 - Person is not externally dependent **(-D)** on Heart *as any Person has a Heart as a part.*

Applying OntoClean

- Properties are annotated according to their characteristics

Not Externally Dependent	-D	Not Rigid	-R
Externally Dependent	+D	Anti-Rigid	~R
Carrying Identity	+I	Carrying Common Unity	+U
Not Carrying Identity	-I	Not Carrying Common Unity	-U
Carrying Own Identity	+O	Carrying Anti-Unity	~U
Rigid	+R		

Applying OntoClean

- **The Rules:**

- Given two properties **p** and **q**, where **q** **subsumes** **p** ($p \sqsubseteq q$), the following constraints must hold:

1. If **q** is **anti-rigid** ($\sim R$), then **p** must be anti-rigid ($\sim R$)
2. If **q** carries an **identity** criterion, then **p** must carry the same criterion
3. If **q** carries a **unity** criterion, then **p** must carry the same criterion
4. If **q** has **anti-unity** ($\sim U$), then **p** must also have anti-unity ($\sim U$)
5. If **q** is **externally dependent** (+D), then **p** must be (+D)

- Analysis of a hierarchy using the rules can help identify problematic modelling choices.
- Conceptual backbone of rigid properties

- **Additional Rules for Identity:**
 - **Sortal Individuation**
 - Every element of the domain must instantiate some property carrying an Identity Criterion (+I)
 - “No entity without identity”
 - **Sortal Expandability**
 - If something is an instance of different properties (for example at different times), then it must also be an instance of a more general property carrying a criterion for its own identity
- ▶ Every entity must instantiate a unique most general property carrying a criterion for its identity



08: EXTRA - More Ontology Evaluation (Part 2)

OpenHPI - Course Knowledge Engineering with Semantic Web Technologies

Lecture 5: Ontological Engineering