

Week 8-9 Suggested Tutorial Solution: Ontology Quality

Semester 1, 2006

Criticise the ontology in terms of the five principles of Gruber.

- Clarity
 - Regarding to *associations*, to enhance the clarity, we can introduce the concepts from other things: -
 - Multiplicity (Cardinality) constraints can be brought into our ontology from UML (Unified Modelling Language). The *multiplicity* of an association end is the number of possible instances of the class association with a single instance of the other end. For example, on figure 1, the LSH shows the original association on our ontology without the multiplicity, there are no restrictions specified that how many instances of class *lease* will associate with the number of instances of class *building*. On the RHS, it shows one instance of class *lease* is allowed for associating with one particular instance of class *building*. It makes clear to the agents committing to our ontology and regulates their behaviour to the certain extent.



Figure 1

- Property *part of* from Bunge-Wand-Weber (BWW) system, one of formal upper ontologies, can help make implicit structure explicit. For example, the association between class *lease* and *clauses* is simply said that the instances of *clauses* are properties of the instances of *lease* on the solution for week 4 tutorial. Actually, there is a *part of* relationship between those classes with respect to BWW. This means that the domain of property *has* is a complex object, while the range of the *has* is the components of the domain.

- Concept *coupling* introduced by BWW can be employed to clarify the identity of the association between some of classes. For example, the linkage among class *estate agent*, *rent* and *lessee*, that was discussed in week 7 tutorial. Actually, the instances of *estate agent* and *lessee* are coupled in transitive closure through the instance of *rent*.
- How are *owners* created in the accommodation ontology? The introduction of DOLCE upper ontology can help detect the above defect because *endurants* are created, and then destroyed by *perdurants*. In the ontology, the creation of the owner has not been taken into account.
- Regarding to the subsumption, defined subclasses are encouraged due to objectivity, while declared classes are discourage due to subjectivity. Furthermore, *derived property/attribute* could be used in defining subclasses. Derived properties are derivable from stored properties/attributes. For example, class *lessee* can be specialized into two subclasses – *good* and *bad lessee*. A good lessee always pays rent timely and in full amount, but a bad lessee does not. Thus, the properties for such definition derived from the properties – the amount of rent and the date the rent should be paid.

- Coherence

The proposed changes to increase clarity all need to be supported by software that can carry out the reasoning specified: to check the multiplicities, to collect all the parts of a whole, to derive the couplings, to assign individuals to defined subclasses and to check the consistency of the subclass definitions with each other.

For example, the clauses on a lease are governed by Residential Tenancies Authority (RTA) and set by a lessor respectively. Therefore, there could be inconsistency in those clauses. For examples: -

- The bond amount could be more than 4-week rent, which is against RTA.
- No lease required for renting a house could happen if an instance of class *lessee* associates with zero instance of class *lease*,
- The date on a particular lease could be later than the entry date of the house specified by that lease.

Thus, we need to increase the clarity by representing these aspects of the lease in a formal language, and a reasoning tool to detect and fix up such incoherence on the ontology.

- Extendibility

- All the changes meant to increase clarity should be expressed step by step. When we represent the clauses of the lease, we should represent each separate concept distinctly, even if two concepts always occur together. For example of class *lease*, there are instances of class *clauses* as part of the instance of *lease*. As mentioned previously, the clauses in a lease are governed by RTA and set by the lessor respectively. To enhance the extendibility, the clauses are suggested to be specialized into two subclasses – *RTA clauses* that are the clauses specified by RTA in general, and *lessor clauses* that are set by the lessor. The primary advantage for

such specialization is any change on each part can be handled independently rather than going through all clauses on a lease. Furthermore, each clause consists of a number of subclauses, this means that any changes on the subclauses can reflect on its superclause immediately, in turn, on the terms on the lease, with which lessors and lessees comply. For example of clause *Rent in advance*: -

The lessor may require the tenant to pay rent in advance only if the payment is no more than: -

- a) for a periodic agreement - 2 weeks rent; or*
- b) for a fixed term agreement - 1 month's rent.*

Under such circumstance, any change on one of the subclauses will reflect on *Rent in advance* immediately. In addition, any new conditions can be simply appended under clause *Rent in advance* can be effective immediately. The above arrangement for clause *Rent in advance* increases the extendibility of the *lease*, and in turn, of the ontology.

- On the LHS of figure 2, the subsumption from our ontology is not extendable properly because it ignores a fact that the properties are furnished with furniture only. Whereas, the RHS revised subsumption encompasses all likelihoods and becomes more extendable.

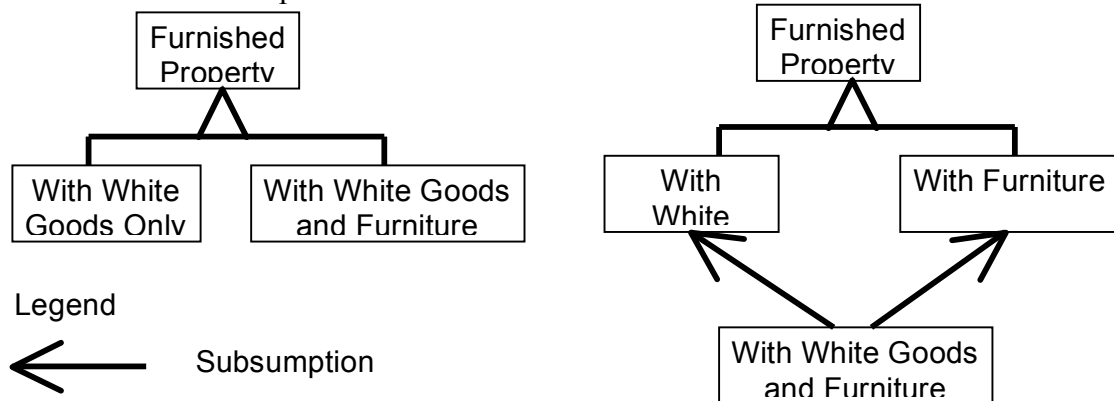


Figure 2

- Encoding bias

Obviously, our ontology is free from implementation details. However, we need to think about encoding bias as we implement the ontology. For example of encoding schemes, if we adopted Unicode alone as our encoding scheme for implementation, the agents that know ASCII would be in trouble for exchange. Choice of Unicode is an encoding. Whether it is bias depends on whether the ontology is intended to support a particular exchange or whether it is intended as a general template to be used as a component in the design of many exchanges. In the first case the encoding is necessary, in the second case it is bias.

Furthermore, built-in data-types for different programming languages could be issues for encoding bias. For instance, if our ontology were implemented by Java programming language, the agents written in C programming language could have trouble in interoperation. One of reasons behind that is, for example, ArrayList is a Java class to stand for a particular type of data structure, but C does not support such type, hence such interoperation would

fail. However, if the ontology were implemented by an XML serialization of the conceptual model, then each agent could implement the high-level data structures in the way most convenient to them, removing the encoding bias.

- Ontological commitment

There are two perspectives we can consider about an ontological commitment:

-

1. From the perspective of reusability, currently, the ontology is restricted to Brisbane, we can extend the subclasses of *amenity* to include class *tram stop*, so the ontology can be used in Melbourne. Likewise, there is a subclass *beachside suburb* under class *suburb* as the ontology is extended to Perth and Sydney. So for reusing the ontology elsewhere in Australia, the ontological commitment is fairly low.
2. If we wanted to implement the ontology in Hong Kong, there are other amenities like *MTR (Mass Transit Railway) station* that could be added. But in Hong Kong, the tenant pays a commission to the agent, while in Australia the commission is paid by the owner. If our ontology included payment of commission, the responsibility of who pays would have to be changed. The ontological commitment is higher.

If we wanted to use the ontology to support an exchange of renting market stalls or parking spaces, the owner and lease parts could be re-used but the parts of the ontology describing residential property are irrelevant. The ontological commitment is much higher for changes in the kind of domain. Modularization would be a way to go for decomposing and packaging the ontology into relatively small components. If we developed modules for owner, property, tenant and lease, it would be easier for an exchange supporting market stalls. They could choose only the modules for owner, tenant and lease, replacing the property module with their own, or a component from another source. Ontological commitment would be much lower.

Propose an improvement to the ontology. Argue why this improvement is a good idea in terms of at least one of Gruber's principles. Examine the cost and benefits of the improvement, taking into account the generality of the ontology and the number of implementations one might expect it to have. On balance, is the improvement a good idea? Take a position and justify it.

Any of the proposed changes to increase clarity, coherence, extendibility or to reduce encoding bias or ontological commitment qualify as proposed improvements.

Each of these has associated costs. One aspect of cost is the effort needed to make the change. For examples, Introducing cardinality constraints or defined subclasses is fairly easy, while representing the clauses of the lease in a formal language takes much more effort. A second aspect of cost is the cost of software to do the checks in the coherence principle. Cardinality constraints or defined subclasses can be implemented in SQL, which would likely be available to implementers, while the reasoning needed to do the checks on the formal representation of the lease would require more powerful and less commonly available software, so a greater cost.

The benefit of these changes comes from reduced cost of implementation and reduced cost of commitment (less likelihood of misunderstanding). These benefits are much greater for an ontology intended to be reused many times in a wide range of applications.

The choice therefore depends on how you see the benefit relative to the cost.