1 Question 2

There are various techniques to debug an ontology. Here follows a discussion of various techniques used to debug ontologies in the field.

As Parsia, Sirin, and Kalyanpur (2005) describes, improving the way the ontology is visually represented and how readable it is, can make it easier to spot mistakes in the ontology. For example, when Swoop is in debug mode, any inferred relationships for an entity definition are italicised. Also named classes that are unsatisfiable are marked with a red icon.

Backjumping is a technique used to determine which clashes between axioms in the ontology is causing an inconsistency in the ontology (Parsia et al., 2005). There are various types of clashes for instance when an individual is a member of a class an that class' complement. This is called an *Atomic* clash. Or a *Cardinality* clash where the individual is related to more individuals than its max cardinality permits. A reasoner can find multiple clashes in an ontology, including clashes that are not to be regarded as errors in the ontology but clashes that help the tableau rules to find the correct model. Backjumping adds additional labels to the various types and property assertions of entities in the ontology. This allows the reasoner, as it follows the branches by utilizing the tableau rules, to track the various branches that created the clashes. The reasoner will stop when a clash that does not depend on a non-deterministic branch arises. This is a popular technique used in many well known reasoners. Additionally reasoners need to keep track from which assertions these problematic axioms are derived. In order for the reasoner to be able to tell the user which assertions have caused the issue (Parsia et al., 2005).

Versioning of the an ontology can also be of benefit. Between each version of the ontology a log is created. This log can help the person who is debugging the ontology to find the error by listing all the areas that were updated in the previous versions. This can help to narrow down where the potential problem or problems are (Parsia et al., 2005).

Another approach, described in Guarino and Welty (2009), is instead of trying to debug or find an error in the ontology after the fact, rather to approach the creation and maintenance of an ontology through a methodology like OntoClean. OntoClean attempts to classify entities to help the creator of the ontology to realise the logical consequences of certain design decisions. The properties of these entities can be classified in various ways. Firstly, is the property of the entity in question essential to that entity. In other words which properties have to hold true in "every possible world" that entity can be in. Those properties that have to hold true are called rigid while those that can lose instances due to other external influences, like time, are called non-rigid. While those that properties that are not essential to their instances at all are called anti-rigid properties.

Other way to classify entity properties in the OntoClean methodology are via their identity or unity criteria. To determine the identity is to determine whether entities are the same or different entities and which properties determine that. These properties that determine identity is then marked for further analysis. The unity criterion is also of importance. The unity criterion describes which parts of an entity form an entity (Guarino & Welty, 2009).

As ontologies become larger and more complex, debugging techniques like these become even more essential.

References

Guarino, N., & Welty, C. A. (2009). An Overview of OntoClean. In *Handbook on ontologies* (pp. 201–220). doi: 10.1007/978-3-540-92673-3_9

Parsia, B., Sirin, E., & Kalyanpur, A. (2005). Debugging OWL Ontologies. In *Proceedings of the 14th international conference on world wide web* (pp. 633–640). New York, NY, USA: ACM. doi: 10.1145/1060745.1060837