RepOSE : A System for Debugging is-a Structure in Networked Taxonomies

Patrick Lambrix and Qiang Liu

Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden

Introduction. Developing ontologies is not an easy task and often the resulting ontologies are not consistent or complete. Such ontologies, although often useful, lead to problems when used in semantically-enabled applications. Wrong conclusions may be derived or valid conclusions may be missed. RepOSE (*R*epair of *O*ntological *S*tructure *E*nvironment) tackles the problem of debugging the is-a structure of a fundamental kind of ontologies, i.e. taxonomies. It is a system that supports domain experts in detecting and repairing wrong and missing is-a relations. Using this system we have debugged the ontologies of the Anatomy track of OAEI 2010: MA (2744 concepts and 1807 asserted is-a relations) and NCI-A (3304 concepts and 3761 asserted is-a relations). The debugging took ca 5 hours and resulted in 107 new is-a relations added to MA and 64 to NCI-A, together with 3 asserted is-a relations removed from MA and 12 from NCI-A.

System. The input to RepOSE is an ontology network consisting of taxonomies (to be debugged) and correct mappings between the taxonomies. The debugging process consists of the phases of detecting and validating possible defects, and repairing wrong and missing is-a relations. The whole process is driven by the end user (domain expert). At any time during the process, the user can switch between different ontologies, start earlier phases, or switch between the repairing of wrong and missing is-a relations. The process ends when there are no more defects or defect suggestions to deal with. In the current version of RepOSE we have focused on detecting defects using the knowledge inherent in the network. (Other approaches are complementary and can also be used.) RepOSE suggests defects in the form of possibly missing is-a relations which are then validated by a domain expert. This gives us missing and wrong is-a relations. For these defects RepOSE computes repairing actions, i.e. is-a relations to add to and remove from the ontologies such that the missing is-a relations will be derivable from their host ontologies and the wrong is-a relations will not be derivable from the ontology network. To our knowledge, this is the first system that deals with both missing and wrong is-a relations in networked ontologies, and where the repairing of missing is-a relations is more advanced than just adding them to the ontologies.

Detection and validation. In RepOSE, the user loads the ontologies and mappings. Then the user can choose an ontology and click `Generate Candidate Missing is-a Relations` to compute the candidate missing is-a relations. These are is-a relations between concepts in an ontology which are logically derivable from the ontology network but not from the ontology alone. The result is shown as directed graphs in an interactive display. To reduce information overload while still giving the user information about interactions between the is-a relations, we show the candidate missing is-a relations in groups where for each member of the group at least one of the concepts subsumes or is subsumed by a concept of another member in the group. For instance,

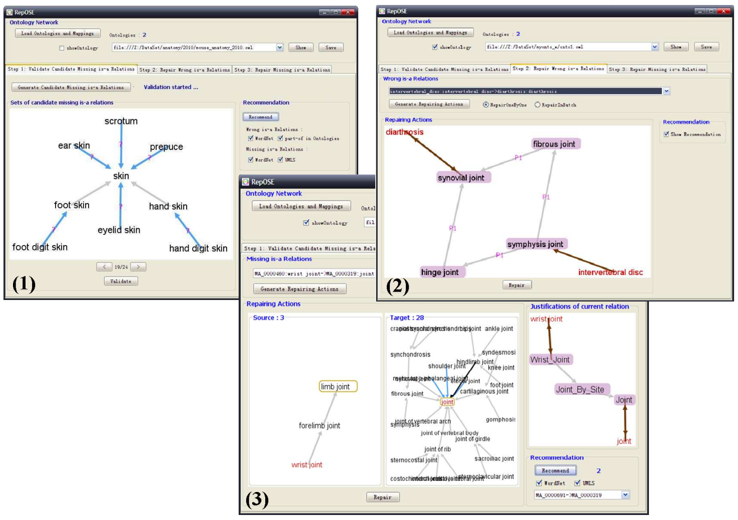

Fig. 1. User interface of RepOSE, including examples of (1) detecting and validating candidate missing is-a relations; (2) repairing wrong is-a relations; (3) repairing missing is-a relations.

Figure 1(1) shows a group of 6 candidate missing is-a relations (shown using blue arrows - (*ear skin*, *skin*), (*scrotum*, *skin*), (*prepuce*, *skin*), (*eyelid skin*, *skin*), (*hand digit skin*, *hand skin*) and (*foot digit skin*, *foot skin*)), together with 2 existing is-a relations (shown using grey arrows - (*hand skin*, *skin*) and (*foot skin*, *skin*)). The candidate missing is-a relations should be validated by a domain expert as being missing is-a relations or wrong is-a relations. Initially, they are shown using arrows labeled by '?' which the user can toggle to 'W' for wrong relations and 'M' for missing relations. Further, we implemented a recommendation algorithm for validation. As is-a and part-of are often confused, the user can ask for a recommendation based on existing part-of relations in the ontology or in external domain knowledge (WordNet). If a part-of relation exists between the concepts of a candidate missing is-a relation, it is likely a wrong is-a relation (the '?' label is replaced by a 'W?' label). Similarly, recommendations for missing is-a relations (the '?' label is replaced by a 'M?' label) can be generated based on the existence of is-a relations in external domain knowledge (WordNet and UMLS). When a user decides to finalize the validation of a group of candidate missing is-a relations, RepOSE checks for contradictions in the current validation as well as with previous decisions and if contradictions are found, the current validation will not be allowed and a message window is shown to the user. We note that a user can validate all or some of the candidate missing is-a relations as well as switch to another ontology.

Repairing wrong is-a relations. Figure 1(2) shows the RepOSE tab for repairing wrong is-a relations. Clicking on the `Generate Repairing Actions` button, results in the computation of repairing actions for each wrong is-a relation of the ontology under repair. The wrong is-a relations are then ranked in ascending order according to the number of possible repairing actions and shown in a drop-down list. Then, the user can select a wrong is-a relation and repair it using an interactive display. The display shows a directed graph representing the justifications [1]. The nodes represent concepts while the edges represent is-a relations in the justifications. These is-a relations may be existing asserted is-a relations (shown in grey) which essentially are the possible repairing actions, mappings (brown), unrepaired missing is-a relations (blue) and the added repairing actions for the repaired missing is-a relations (black). The concepts in the wrong is-a relation are displayed in red. Concepts in other ontologies are marked with background in different colors. For instance, Figure 1(2) shows the display for the wrong is-a relation *(intervertebral disc, diarthrosis)*, which contains 4 existing asserted is-a relations, two mappings, and 4 concepts, i.e. *synovial joint*, *hinge joint*, *fibrous joint* and *symphysis joint*, from another ontology in the network. The user can choose to repair all wrong is-a relations together or one by one. For the wrong is-a relations under repair, the user can choose, by clicking, multiple existing asserted is-a relations on the display as repairing actions and click the `Repair` button. RepOSE ensures that only existing asserted is-a relations are selectable, and when the user finalizes the repair decision, RepOSE ensures that the wrong is-a relations under repair and every selected is-a relation will not be derivable from the ontology network after the repairing.

During the repairing, the user can choose to use the recommendation feature by enabling the `Show Recommendation` check box. The recommendation algorithm will then compute hitting sets [3] for all the justifications of the wrong is-a relations under repair. Each hitting set contains a minimal set of is-a relations to be removed to repair the wrong is-a relations. The recommendation algorithm then assigns a priority to each possible repairing action based on how often it occurs in the hitting sets. For instance, in the case of Figure 1(2), the hitting sets are $\{$(*symphysis joint*, *fibrous joint*), (*symphysis joint*, *hinge joint*)$\}$, $\{$(*symphysis joint*, *fibrous joint*), (*hinge joint*, *synovial joint*)$\}$, $\{$(*fibrous joint*, *synovial joint*), (*symphysis joint*, *hinge joint*)$\}$ and $\{$(*fibrous joint*, *synovial joint*), (*hinge joint*, *synovial joint*)$\}$. Each of the is-a relations appears twice in the hitting sets and thus are recommended with equal priority (indicated by the pink labels marked 'Pn', where n reflects the priority ranking). Upon the selection of a repairing action, the recommendations are recalculated and the labels are updated. As long as there are labels, more repairing actions need to be chosen. When the repairing is executed, a number of updates need to be done. New candidate missing is-a relations may appear. Some other wrong is-a relations may also have been repaired by the current repairing. Some repaired missing is-a relations may become missing again. In other cases the possible repairing actions for wrong and missing is-a relations may change. RepOSE computes these consequences of the repair and performs the necessary updates.

Repairing missing is-a relations. Figure 1(3) shows the RepOSE tab for repairing missing is-a relations. Clicking on the `Generate Repairing Actions` button, results in the computation of repairing actions for the missing is-a relations of the ontology under repair. For a missing is-a relation (a, b) we compute $Source(a, b)$ as the set of more general concepts of a and $Target(a, b)$ as the set of more specific concepts of b. To not introduce equivalence relations where in the original ontology there are only is-a relations, we remove the super-concepts of b from $Source(a, b)$, and the sub-concepts of a from $Target(a, b)$. Adding an element from $Source(a, b) \times Target(a, b)$ to the ontology makes missing is-a relation (a, b) derivable. Once the Source and Target sets are computed, the missing is-a relations are ranked with respect to the number of possible repairing actions. The first missing is-a relation in the list has the fewest possible repairing actions, and may therefore be a good starting point. When the user chooses a missing is-a relation, its Source and Target sets are displayed on the left and right, respectively, within the `Repairing Actions` panel (Figure 1(3)). Both have zoom control and can be opened in a separate window. Concepts in the missing is-a relations are highlighted in red, existing asserted is-a relations are shown in grey, unrepaired missing is-a relations in blue and added repairing actions for the missing is-a relations in black. For instance, Figure 1(3) shows the Source and Target sets for the missing is-a relation *(wrist joint, joint)*, which contain 3 and 26 concepts, respectively. The Target panel shows 3 unrepaired missing is-a relations, i.e. *(elbow joint, joint)*, *(shoulder joint, joint)* and *(metacarpo-phalangeal joint, joint)*, as well as a previously added repairing action *(hinderlimb joint, joint)*. The `Justifications of current relation` panel is a read-only panel that displays the justifications of the current missing is-a relation as an extra aid.

For the selected missing is-a relation, the user can also ask for recommended repairing actions by clicking the `Recommend` button. The recommendation algorithm computes for missing is-a relation (a, b) the most informative [2] repairing actions from $Source(a, b) \times Target(a, b)$ that are supported by domain knowledge. In general, the system presents a list of recommendations. By selecting an element in the list, the concepts in the recommended repairing action are identified by round boxes in the panels. For instance, for the case in Figure 1(3), the recommendation algorithm proposes to add *(limb joint, joint)* to repair *(wrist joint, joint)*.

The user can repair the missing is-a relation by selecting a concept in the Source panel and a concept in the Target panel and clicking on the `Repair` button. Sometimes, the selected repairing action may contradict with already known wrong is-a relations. In such cases, the repairing will not be allowed and a message window is shown to the user. When the selected repairing action is allowed, the repairing action is executed, and a number of updates need to be done. New candidate missing is-a relations may appear. Some other missing is-a relations may also have been repaired by the current repairing. Some repaired wrong is-a relations may also become derivable again. In other cases the possible repairing actions for wrong and missing is-a relations may change. RepOSE computes these consequences of the repair and performs the necessary updates.

Demonstration. In the demonstration we guide the visitors through a debugging session using (parts of the) ontologies from the Anatomy track of OAEI. Further, we explain the algorithms for the detection and validation of defects, as well as the generation, recommendation and execution of repairing actions.

References

1. E Jimenez-Ruiz, B Cuenca Grau, I Horrocks, and R Berlanga. Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences. In *6th European Semantic Web Conference*, pages 173–187, 2009.
2. P Lambrix, Q Liu, and H Tan. Repairing the Missing is-a Structure of Ontologies. In *4th Asian Semantic Web Conference*, pages 76–90, 2009.
3. R Reiter. A theory of diagnosis from first principles. In *Artificial Intelligence*, volume 32, pages 57–95, 1987.