A Tableau-Based Explainer for DL Subsumption

Thorsten Liebig and Michael Halfmann

Dept. of AI, University of Ulm, D-89069 Ulm, Germany
liebig@informatik.uni-ulm.de
michael.halfmann@informatik.uni-ulm.de

Abstract. This paper describes the implementation of a tableau-based reasoning component which is capable of providing quasi natural language explanations for subsumptions within $\mathcal{ALEHF}_{R^+}$ TBoxes.

1 Motivation

W3C's recently recommended ontology language OWL is expected to be used even by non-sophisticated end users. However, the Description Logics (DLs) underlying OWL Lite ($\mathcal{SHIF}$) and OWL DL ($\mathcal{SHOIN}$) are quite expressive [1]. Authoring ontologies presumably is not possible without a basic understanding of the underlying reasoning services. Our explainer MEX[1] aims at supporting a deeper comprehension of subsumption, the core inference service, by providing an on-demand step by step quasi-natural language explanation.

MEX is capable of explaining subsumptions within a significant fraction of the DL underlying OWL Lite, namely definitorial $\mathcal{ALEHF}_{R^+}$ TBoxes with global domain and range restrictions. Such TBoxes require all axioms to be of the form $A \sqsubseteq D$ or $A \equiv D$, with A atomic and unique. The language $\mathcal{ALE}$ covers the top and bottom concept ($\top, \bot$), conjunction ($C \sqcap D$), qualified existential ($\exists r.C$) and universal quantification ($\forall r.C$), and atomic negation ($\neg A$). $\mathcal{ALEHF}_{R^+}$ extends $\mathcal{ALE}$ with role hierarchies ($p \sqsubseteq r$), a limited form of cardinality restrictions ($\leq n\ r$) with n either 0 or 1, and transitive as well as functional roles. Our system implements and extends the theoretical approach for explaining $\mathcal{ALC}$ subsumptions described in [2] which proposes to utilize a sequent proof derived from a tableau style algorithm. Although not explicitly present in our language, disjunction comes implicitly on rhs due to the refutation strategy of the tableau algorithm.

2 Implementation

Tableaux systems implement a refutation proof strategy. However, to prove a conclusion by showing its opposite to be contradictory doesn't seem to be intuitive. In our opinion, humans usually try to reduce a complex problem into sensible pieces which are more easily comprehensible. In consideration of the latter,

[1] Acronym for My EXplainer.

our system explains a subsumption by breaking it down into sub-subsumptions until those can be explained by simple statements. This course of action is triggered by a tableau-based approach, where tableaux rules are applied until a terminating clash occurs. However, for the task of explaining how a proof has been derived, the refutation strategy has to be hidden to the user. In order to achieve this we use a technique called tagging [2]. Tagging allows to reconstruct the original query for illustration of the derivation steps at any time of tableau processing. To be concrete, we distinguish between the right-hand side (rhs) and the left-hand side (lhs) of a subsumption query by tagging the rhs.

Our explainer MEX is implemented in Lisp. Its syntax for defining concepts and roles follows the KRSS standard [3], which easily allows to transfer existing ontologies from other systems, most notably the widely used RACER [4] reasoner. The internal data-structure is based on Lisp `structure` data-types. A node of a tableau proof tree is represented as a `structure` object, storing the subsumee (lhs) and subsumer (rhs) of the corresponding query, its role successor nodes, a list of disjunctive alternative nodes in case of a disjunction on rhs, a reference to its parent node, and some further parameters used for blocking and optimization. MEX also implements lazy unfolding, a well known optimization technique of DL tableaux algorithms. Lazy unfolding delays the unfolding of a concept to its given definition until it is required by the proof algorithm to proceed. This also helps to maximize performance in typical ontologies since only those axioms are taken into account which are actually proof relevant.

While successively creating the tableau tree MEX generates a corresponding proof explanation in parallel. Such an explanation is a list of atomic explanation steps. For each relevant tableau rule application or unfolding one or more explanation steps are added. An explanation step consists of its corresponding type, its depth in the tableau tree, a list of additional information (relevant concepts, nodes, etc.) and a textual explanation. The type and the additional parameters stored within each step should enable an external component to layout explanations with respect to the given application context (e. g. degree of detail, language, hierarchical structure vs. flat ordering).

As mentioned before, each tableau transformation will result in one or more explanation steps. A tableau transformation consists of unfolding steps as well as tableaux rules. For example, a role-successor node will be explained by introducing the new node and a description of the constraints which follow from the qualified quantifications on that role. Explanations will be given only to those nodes which are clash relevant. Since we extend previous work about explaining $\mathcal{ALC}$ we had to add explanations for cardinality restrictions, role hierarchies, merging of role-successors, and domain and range restrictions.

The expansion of a tableau branch terminates as soon as a clash is found. For each type of clash we have formulated a quasi-natural language statement, explaining the subsumption relationship of the original query. For example, due to the fact that we have to take the origin of the clash relevant constructs into account we have to distinguish between four types of cardinality clashes. Other language constructs like transitive roles and domain as well as range restrictions

do not add extra clash types but require additional steps explaining the source of the expressions they append to existing tableaux nodes. The elements of a disjunction (which only can occur on rhs) within a node are stored in an or-branch slot of the `structure` object. They all have to clash in order to close the node they occur in. From the viewpoint of explaining they correspond to a conjunction and will therefore be explained with help of an enumeration of sub-explanations. For the lack of space we refer to [5] for a more detailed description about explaining $\mathcal{ALEHF}_{R^+}$ TBoxes.

When processing a conjunction of different expansions (e. g. role successors) the first one in which a clash is found will be explained by MEX. A future optimization could select the most intuitive explanation from the set of alternatives.

To generate concise and simple explanations we have implemented several optimizations which condense the explanation in specific situations. The two most important ones, filtering and mode switching, are described in the following.

Filtering is a simple method for pruning disjunction on rhs with help of a structural comparison. Consider the subsumption $A \sqcap B \sqcap C \sqsubseteq A \sqcap C$. The standard approach would split up the explanation into two sub-subsumptions according to the internal disjunction on rhs. Very likely this is unnecessary because this subsumption obviously holds, since the subsumee is a direct specialization of the subsumer. Therefore MEX structurally compares the lhs and rhs after lazy unfolding and prior to any further processing. If the rhs is a syntactical subset of the lhs an obvious subsumption is found and explained accordingly. A distantly related technique, called normalizing and naming is found in the DL literature.

Another optimization method is called mode switching and applies to situations where at some stage of tableau processing either the rhs or lhs is unsatisfiable on its own (i. e. independently from each other). This corresponds to either the subsumee being equivalent to $\bot$ or the subsumer being equivalent to $\top$ on explanation side. In such a case our explainer will switch to either unsatisfiability or tautology explaining while disregarding the other side. Since our explanations are generated on-the-fly while building the tableau tree this requires to check each side concerning unsatisfiability in advance. We use the RACER reasoner as an external component for this test. An additional benefit of the optional RACER connection is the increased performance when using the dual-reasoner architecture as proposed in [6]. This optimization feature uses RACER to determine the effectively clash relevant expressions within each side. This will prune the explainer tableau by leaving out irrelevant node expansions.

3 OntoTrack Integration

Explaining an inference service can improve the authoring process of an ontology to a large degree. Consequently, explanations are most powerful when combined with an ontology editing tool. Therefore we made MEX accessible to our interactive ontology authoring tool ONTOTRACK [7] via the offered plug-in interface. Figure 1 displays the most important system components with respect to this integration. As an on-demand functionality ONTOTRACK converts the current

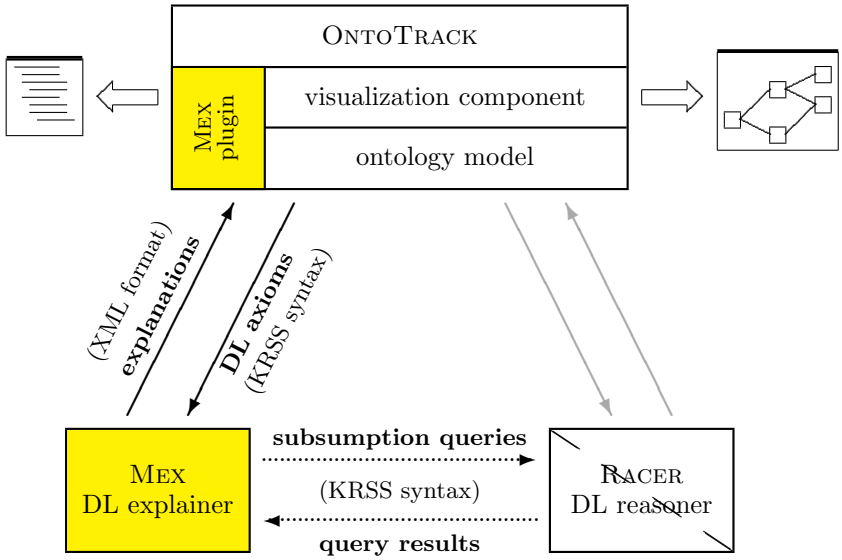

Fig. 1. Architecture of Mex as an OntoTrack plugin with optional Racer link-up.

internal ontology model into KRSS [3] syntax and sends it together with an explanation request to Mex via TCP. An explanation is generated and transmitted as an XML serialization back to OntoTrack. This enables the OntoTrack plug-in to display the explanation in a graphical way as an expandable tree list.

The Racer link-up is optional, but leads to optimized explanations (with mode switching) and an increased performance. OntoTrack itself also uses Racer for instant reasoning feedback to user interactions. As indicated in Figure 1, OntoTrack and Mex currently use two different Racer TBox instances because of lack of a standardized mapping of ontology axioms into Racer TBoxes. Future work will be concerned with a standardized Racer connection, so that Mex is able to access the OntoTrack model serialization directly.

Figure 2 shows a screen-shot of OntoTrack displaying a sample ontology and an explanation for A subsuming B provided by Mex. To invoke an explanation a user only has to select "Explain" within the mouse enabled context menu on edges representing a subsumption relationship.

References

1. Horrocks, I., Patel-Schneider, P.: Reducing OWL entailment to description logic satisfiability. Journal of Web Semantics **1** (2004) 345–357
2. Borgida, A., Franconi, E., Horrocks, I., McGuinness, D., Patel-Schneider, P.: Explaining $\mathcal{ALC}$ subsumption. In: Proc. of the Int. Workshop on Description Logics (DL99), Linköping, Sweden (1999) 37–40
3. Patel-Schneider, P., Swartout, B.: Description-Logic Knowledge Representation System Specification (1993) Working Version (Draft).

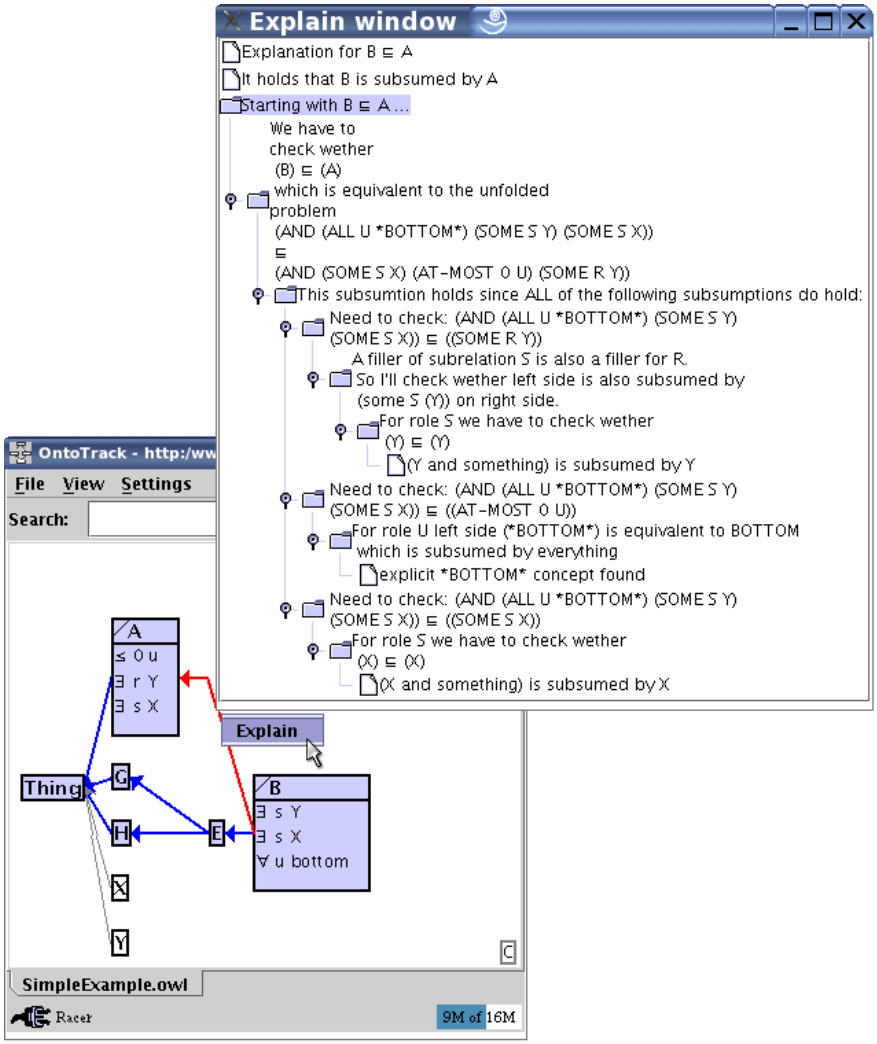

Fig. 2. OntoTrack with an explanation

4. Haarslev, V., Möller, R.: Description of the Racer System and its Applications. In: Proc. ot the Int. Workshop on Description Logics (DL01), Stanford, CA, USA (2001)
5. Liebig, T., Halfmann, M.: Explaining Subsumtion in $\mathcal{ALEHF}_{R^+}$ TBoxes. In: Int. Workshop on Description Logics (DL05). (2005) to appear.
6. Kwong, F.: Explaining Description Logic Reasoning. In: Proc. of the 2004 Int. Workshop on Description Logics (DL04), Whistler, BC, Canada (2004) 210
7. Liebig, T., Noppens, O.: OntoTrack: Combining Browsing and Editing with Reasoning and Explaining for OWL Lite Ontologies. In: Proc. of the Int. Semantic Web Conference (ISWC 2004), Hiroshima, Japan (2004) 244–258