COBRA, a Demo

Rafael Peñaloza[1,2⋆] and Aparna Saisree Thuluva[1⋆⋆]

[1] Institute for Theoretical Computer Science,
Technische Universität Dresden
[2] Center for Advancing Electronics Dresden
penaloza@tcs.inf.tu-dresden.de,aparna_saisree.thuluva@tu-dresden.de

COBRA is a tool for performing context-based reasoning in OWL ontologies, with an emphasis on its applications. COBRA is built on top of standard OWL reasoning tools, and uses the OWL-API, which makes it easy to use, maintain, and keep up-to-date with reasoning technologies.

To understand the behaviour of COBRA, we must first explain the idea of context-based reasoning. We consider as input for the system an *annotated* OWL ontology. Intuitively, this annotated ontology is a compact representation of a class of ontologies, which we call *contexts*. More precisely, every axiom in an OWL ontology is annonated, using the owl:annotation tag, with a list of labels that express the contexts that this axiom belongs to. For example, if we have two contexts $\mathcal{C}_1$ and $\mathcal{C}_2$ containing axioms α_1 and α_2; and α_2 and α_3, respectively, then the annotated input ontology $\mathcal{O}$ will contain three axioms α_1, α_2, and α_3 annotated with $\{c_1\}$, $\{c_1, c_2\}$, and $\{c_2\}$, respectively. One of the main goals of context-based reasoning is to identify the class of contexts that entail a given logical consequence.

An obvious method for solving this reasoning problem would be to first extract all the different contexts from the annotated ontology, and perform standard reasoning in all of them. Given the speed of state-of-the-art OWL reasoners, this approach, in the following called the *naïve approach* produces a very efficient reasoning procedure for realistic OWL ontologies, assuming that the number of contexts is small. However, as the number of contexts grows, so does the total execution time of the naïve method, by a linear factor.

To handle cases where the number of contexts is high (e.g., above a thousand) more effective methods based on axiom-pinpointing, for detecting large clusters of contexts entailing the consequence, through only one call to the reasoner, and a variant of Reiter's Hitting Set Tree (HST) algorithm [3] for ensuring that all possible contexts have been detected through a systematic search. Specifically, we use the HST approach for computing the so-called boundary of a consequence described in [1]. Notice that in this case, the HST approach can be further optimized, since the background lattice required in [1] corresponds to the class of all sets of labels, with union and intersection as operators, which are easier to compute.

COBRA implements the naïve and the HST methods via a sequence of black-box calls to standard OWL reasoners. For the HST method, it must call a rea-

⋆ Partially supported by DFG within the Cluster of Excellence 'cfAED'
⋆⋆ Supported by the International MSc Program in Computational Logic (MCL)

soner capable of explanation services. Specifically, the reasoner must not only answer whether a consequence holds from an ontology or not, but in the affirmative case, additionally provide a justification; that is, a minimal sub-ontology still entailing this consequence. Additionally, since the naïve and the HST algorithms both require reasoning over several, slightly different, ontologies, incremental reasoning tools have a strong positive impact in the performance of the overall system. The current version of COBRA uses the explanation capabilities of HermiT [4] and the very efficient incremental reasoner ELK [5] whenever the input ontology adheres to the OWL 2 EL profile of OWL 2.

A typical application scenario for context-based reasoning is in the area of error-tolerant reasoning, where the different contexts correspond to the repairs of an unwanted consequence (see [2] for details). For that reason, COBRA provides also an ontology-annotation service based on the computation of all repairs. In a nutshell, the system receives as input a classical OWL ontology $\mathcal{O}$, and an unwanted consequence of $\mathcal{O}$. It then computes all the repairs w.r.t. this unwanted consequence, and annotates $\mathcal{O}$ considering each repair as a context. Notice that applying this tool to Snomed CT can lead to situations with over ten million contexts.

The demo will showcase the behaviour of the different components of the tool and their use for practical applications, with special emphasis on error-tolerant reasoning.

References

1. F. Baader, M. Knechtel, and R. Peñaloza. Context-dependent views to axioms and consequences of semantic web ontologies. *J. Web Sem.*, 12:22–40, 2012.
2. M. Ludwig and R. Peñaloza. Error-tolerant reasoning in the description logic EL. In *Proc. of JELIA'14*, volume 8761 of *LNCS*, pages 107–121. Springer, 2014.
3. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
4. R. Shearer, B. Motik, and I. Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In *Proc. of OWLED 2008 EU*, Karlsruhe, Germany, 2008.
5. F. S. Yevgeny Kazakov, Markus Krötzsch. The incredible elk - from polynomial procedures to efficient reasoning with el ontologies. *J. Autom. Reas.*, 53(1):1–61, 2014.