The SCREECH OWL reasoner – Scalable approximate ABox reasoning with OWL*

Pascal Hitzler and **Denny Vrandečić**
AIFB, Universität Karlsruhe, Germany

Abstract

We present a preliminary version of the approximate OWL reasoning system SCREECH. It builds on the KAON2 system and performs OWL ABox reasoning in an approximate manner. It trades soundness of reasoning for efficiency, with resulting polynomial worst-case data complexity. It has been developed for use in time-critical applications where quick response time is more important than a full guarantee of correctness of answers. The theoretical background for the system is explained in [Hitzler and Vrandečić, 2005] and is being presented at the conference.

1 The OWL scalability problem

Knowledge representation and reasoning on the Semantic Web is done by means of ontologies. The W3C established the Web Ontology Language OWL [W3C, 2004] as core standard. It comes in three flavours, as OWL Full, OWL DL and OWL Lite, where OWL Full contains OWL DL, which in turn contains OWL Lite. The latter two coincide semantically with certain description logics and can thus be considered fragments of first-order predicate logic.

OWL ontologies can be understood to consist of two parts, one intensional, the other extensional. In description logics terminology, the intensional part consists of a TBox and an RBox, and contains knowledge about concepts (called *classes*) and complex relations between them (called *roles*). The extensional part consists of an ABox, and contains knowledge about entities and how they relate to the classes and roles from the intensional part. The Semantic Web envisions a distributed knowledge source, built from OWL ontologies and intertwining the knowledge like the Web interconnects websites today.

With an estimated 25 million active websites today and correspondingly more webpages, it is apparent that reasoning on the Semantic Web will have to deal with very large ABoxes. Complexity of ABox reasoning — also called *data complexity* — measures complexity in terms of ABox size only, while considering the intensional part of the ontology to be of constant size. For the different OWL variants, data complexity is at least NP-hard, which indicates that it will not scale well in general. Therefore, methods are sought to cope with large ABoxes in an approximate manner. The idea is to use quick heuristic reasoning when time constraints are more important than the correctness of the answers. A typical use case is online question answering, where it is usually more important to quickly offer the user a set of possible answers instead of letting him wait for a long time in order to receive precise responses. The system shown here can be part of a multi-tier anytime system.

*The authors acknowledge support by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project, and by the European Commission under contract IST-2003-506826 SEKT and under the KnowledgeWeb Network of Excellence. The expressed content is the view of the authors but not necessarily the view of any of the projects as a whole.

2 From OWL to datalog

The approach which we propose is based on the fact that data complexity is polynomial for non-disjunctive datalog. We utilise recent research results about the transformation of OWL DL ontologies into disjunctive datalog, and perform heuristic approximate reasoning by transforming the disjunctive database into a non-disjunctive one.

The transformation is based on the fact that OWL DL is a subset of first-order logic. OWL axioms can thus be translated directly into logical formulas and transformed into clausal form using any of the standard algorithms. The resulting clauses can be represented as disjunctive datalog rules which do not contain negation.

Note, however, that due to possible skolemization steps in the clausal form translation, the resulting datalog rules may contain function symbols. In general, datalog with function symbols is undecidable, but since we obtain the datalog program by a translation from OWL DL, which is decidable, inferencing over the resulting program must be decidable. Standard datalog engines, however, do in general not terminate in the presence of function symbols. To cope whith this problem, a sophisticated method has been presented in [Hustadt *et al.*, 2004] which allows to get rid of the function symbols without loosing ABox consequences. As a result, we obtain a function- and negation-free disjunctive datalog program, which can be dealt with using standard techniques.

There is one other catch: The approach presented in [Hustadt *et al.*, 2004] does not allow to deal with nominals, i.e. it

supports only $\mathcal{SHIQ}(\mathbf{D})$ instead of $\mathcal{SHOIN}(\mathbf{D})$ (the latter is the description logic coinciding with OWL DL). We remark that to date — and to the best of our knowledge — no efficient reasoning algorithms for $\mathcal{SHOIN}(\mathbf{D})$ have been implemented. We will return to a possible treatment of nominals in our approach later.

3 Approximate SLD-Resolution

Having obtained datalog rules of the form

$$H_1 \vee \cdots \vee H_m \leftarrow A_1, \ldots, A_k,$$

ABox reasoning is still NP-hard. For our approximate reasoning approach, we utilize the fact that when all rules are non-disjunctive, i.e. when $m = 1$, then standard resolution methods can be used which render the reasoning to be polynomial with regard to the number of facts. Hence, we use a modified notion of *split programs* [Sakama and Inoue, 1994]. Given the above rule, the *derived split rules* are defined as:

$$H_1 \leftarrow A_1, \ldots, A_k \quad \ldots \quad H_m \leftarrow A_1, \ldots, A_k.$$

For a given disjunctive program P, its *split program* P' is defined as the collection of all split rules derived from rules in P. Polynomial ABox reasoning can now be performed using the split program and classic resolution techniques, e.g. SLD-resolution as used in standard Prolog systems [Lloyd, 1988]. The combined reasoning method, which we call *approximate SLD-resolution*, is obviously complete but unsound, and hence it is necessary to pursue the question of exactly what notion of entailment underlies the approximate reasoning technique we propose. Space restrictions forbid us to go into detail, so it shall suffice to say that approximate SLD-resolution boils down to brave reasoning with well-supported models, where the latter notion is a straightforward adaptation of the notion of well-supported model from [Fages, 1994] to the disjunctive case.

In order to be able to deal with all of OWL DL, we need to add a well-known preprocessing step to get rid of nominals. We can do this by *Language Weakening* as follows: For every occurrence of $\{o_1, \ldots, o_n\}$, where $n \in \mathbb{N}$ and the o_i are abstract or concrete individuals, replace $\{o_1, \ldots, o_n\}$ by some new concept name D, and add ABox assertions $D(o_1), \ldots, D(o_n)$ to the knowledge base. Note that the transformation just given does in general not yield a logically equivalent knowledge base, because some information is lost in the process.

Putting all the pieces together, the following steps describe our approximate ABox reasoning for OWL DL.

1. Apply Language Weakening as just mentioned in order to obtain a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base.
2. Apply transformations as in Section 2 in order to obtain a negation-free disjunctive datalog program.
3. Use approximate SLD-resolution for query-answering.

The first two steps can be considered to be preprocessing steps for setting up the intensional part of the database. ABox reasoning is then done in the last step. From our discussions, we can conclude the following properties of approximate ABox reasoning for $\mathcal{SHIQ}(\mathbf{D})$.

- It is complete with respect to first-order predicate logic semantics.
- It is sound and complete with respect to brave reasoning with well-supported models.
- Data complexity of our approach is polynomial.

4 SCREECH OWL

We have implemented the proposed approach as SCREECH[1], based on KAON2[2]. It utilizes KAON2's sophisticated translation algorithms from OWL DL into datalog, and returns the corresponding split program which can be fed into any standard Prolog interpreter for ABox reasoning.

5 Conclusions

Our approach provides ABox reasoning with polynomial time complexity. It is complete, but it is also unsound with respect to first-order logic. However, the inference underlying our approach can be characterized using standard methods from the area of non-monotonic reasoning.

The checking whether a conjunctive query is a predicate logic consequence of a (negation-free) disjunctive logic program P amounts to checking whether the query is valid in *all* minimal models of P, i.e. corresponds to *cautious* reasoning with minimal models. Along this insight, we foresee the possibility to develop an algorithm which would first find a brave answer of a query, and then substantiate this answer by subsequent calculations. This and other refinements of our approach are in development.

References

[Fages, 1994] François Fages. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.

[Hitzler and Vrandečić, 2005] Pascal Hitzler and Denny Vrandečić. Resolution-based approximate reasoning for OWL DL. In Y. Gil et al., editors, *Proceedings of ISWC05*, volume 3729 of *Lecture Notes in Computer Science*, pages 383–397. Springer, Berlin, 2005.

[Hustadt *et al.*, 2004] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics with a concrete domain in the framework of resolution. In R. López de Mántaras and L. Saitta, editors, *Proceedings of ECAI'2004*, pages 353–357. IOS Press, 2004.

[Lloyd, 1988] John W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 1988.

[Sakama and Inoue, 1994] C. Sakama and K. Inoue. An alternative approach to the semantics of disjunctive logic programs and deductive databases. *Journal of Automated Reasoning*, 13:145–172, 1994.

[W3C, 2004] W3C. Web Ontology Language (OWL). http://www.w3.org/2004/OWL/, 2004.

[1] http://logic.aifb.uni-karlsruhe.de/screech

[2] http://kaon2.semanticweb.org

Demo

In this appendix we describe the implementation and the demo of the approach covered in the main part of this poster description.

Implementation

`Screech` is implemented and distributed as part of the KAON2 OWL tools[3]. These are a set of tools exposing the power of the KAON2[4] ontology management infrastructure to the command line. The KAON2 OWL tools consist of over a dozen of small tools, like `count`, that counts the occurrences of a certain ontology element within an ontology, or `filter`, that filters out ontology elements of a specified kind.

KAON2 and the KAON2 OWL tools are implemented in Java 5 and thus require the Java run time environment of the appropriate version.

The approximate reasoning approach described here makes use of two of the KAON2 OWL tools, namely `deo` and `Screech` itself. `deo` applies the preprocessing step of language weakening on OWL ontologies as described in Section 3, thus removing all nominals.

After that, `Screech` is applied to convert the ontology into a semantically equivalent disjunctive datalog program (a step that can also be done seperately using the `dlpconvert` OWL tool). The resulting program is then analysed and all rules with disjunctive heads are replaced by the appropriate new rules. The resulting split program is then serialised in datalog or F-Logic syntax.

Example

We demonstrate our approach by translating an ontology fragment. This is only to exemplify the main issues of the `Screech` approach. Consider the following axiom:

$$\text{beneluxian} \equiv \text{luxembourgian} \sqcup \text{dutch} \sqcup \text{belgian}$$

Translating this into disjunctive datalog we would get the following statements:

```
beneluxian(X) :- belgian(X).
beneluxian(X) :- dutch(X).
beneluxian(X) :- luxembourgian(X).
belgian(X), dutch(X), luxembourgian(X)
              :- beneluxian(X).
```

`Screech` instead replaces the last rule by the following set of rules:

```
belgian(X)       :- beneluxian(X).
dutch(X)         :- beneluxian(X).
luxembourgian(X) :- beneluxian(X).
```

Thus, our approach changes the ontology by treating the disjunctions in the last line as conjunctions. This change affects the soundness of the reasoning procedure. However, usually most of the ABox consequences which can be derived by approximate SLD-resolution are still correct as can be seen in the following evaluation.

[3] http://owltools.ontoware.org

[4] http://kaon2.semanticweb.org

t_{DD}	t_{split}	Instances	Class Name
11036 ms	6489 ms	154/154	Biological_object
11026 ms	5959 ms	9/9	Specified_set
11006 ms	6219 ms	9/13	Multiple
11015 ms	5898 ms	16/16	Probe_structural...
11036 ms	7711 ms	4/4	Human_red...
11055 ms	5949 ms	24/58	Biological_object...

Table 1: Performance comparison for instance retrieval using disjunctive datalog (t_{DD}) vs. the corresponding split program (t_{split}), on the KAON2 datalog engine. *Instances* indicates the number of instances retrieved using DD versus SPLIT, e.g. class *Multiple* contained 9 individuals, while the split program allowed to retrieve 13 (i.e. the 9 correct individuals plus 4 incorrect ones). The full name of the classes in the last three rows are Probe_structural_part_of_heart, Human_red_blood_cell_mature and Biological_object_that_has_left_right_symmetry.

Evaluation

For our evaluation we have performed experiments with the OWL DL version of the GALEN Upper Ontology,[5] as it appears to be sufficiently natural and realistic. As it is a TBox ontology only, we populated GALEN's 175 classes randomly with 500 individuals.[6] GALEN does not contain nominals or concrete domains. GALEN has 673 axioms (the population added another 500). The TBox translation to disjunctive datalog took about 2300 ms, after which we obtained 2687 disjunctive datalog rules containing 267 disjunctions within 133 rules. Among these were 152 integrity constraints (i.e. rules with empty head), which we removed for our experiment as they led to inconsistency of the database.[7] After splitting disjunctive rules, we arrived at 2802 Horn rules.

We then randomly selected classes and queried for their extension using the KAON2 datalog engine, both for processing the disjunctive datalog program and for the split program. Some of the typical results are listed in Table 1, which indicates a significant speed-up of more than 40% on average, while the vast majority of the retrieved answers is correct. Note that we obtain significant speed-up although the KAON2 datalog engine is not optimized for Horn programs, but rather tuned to efficient performance on definite disjunctive datalog.

Future work

For the future we plan to improve the system by enhanced approximate reasoning algorithms. We also plan to do further evaluation of our approach, and develop a method to evaluate ontologies beforehand in order to estimate the benefit in terms of speed-up as well as the costs in term of possible incorrect answers of applying `Screech` on a specific ontology or even query.

[5] http://www.cs.man.ac.uk/∼rector/ontologies/simple-top-bio/

[6] Using the `pop` KAON2 OWL tool.

[7] This is an expected effect. Removal of the integrity constraints does not destroy completeness of the approximate reasoning procedure.