Approximate OWL Instance Retrieval with SCREECH*

Pascal Hitzler[2], Markus Krötzsch[2], Sebastian Rudolph[2], and Tuvshintur Tserendorj[1]

[1] FZI Karlsruhe, Germany
[2] AIFB, University of Karlsruhe, Germany

Abstract. With the increasing interest in expressive ontologies for the Semantic Web, it is critical to develop scalable and efficient ontology reasoning techniques that can properly cope with very high data volumes. For certain application domains, approximate reasoning solutions, which trade soundness or completeness for increased reasoning speed, will help to deal with the high computational complexities which state of the art ontology reasoning tools have to face. In this paper, we present a comprehensive overview of the SCREECH approach to approximate instance retrieval with OWL ontologies, which is based on the KAON2 algorithms, facilitating a compilation of OWL DL TBoxes into Datalog, which is tractable in terms of data complexity. We present three different instantiations of the Screech approach, and report on experiments which show that the gain in efficiency outweighs the number of introduced mistakes in the reasoning process.

1 Introduction

Scalability of reasoning remains one of the major obstacles in leveraging the full power of the Web Ontology Language OWL [14] for practical applications. Indeed, large-scale applications normally use only a fragment of OWL which is very shallow in logical terms, and thus cannot employ the more sophisticated reasoning mechanisms for accessing knowledge which is implicit in knowledge bases. While the use of such shallow techniques already has added value, it would be preferable if the more complex logical constructors in the language could also be used. Consequently, scalability of OWL reasoning needs to be investigated on a broad front in order to advance the state of the art by several orders of magnitude.

Among the many possible approaches to address scalability, one of them concerns the use of logic programming for this purpose. This can be traced back to the work on Description Logic Programs (DLP) [4, 23], which are a naive Horn fragment of OWL DL.[3] Along the same lines lies the OWL DL-fragment Horn-$\mathcal{SHIQ}$ [7, 16, 11], which is based on the transformation algorithms implemented in the KAON2-system[4] [16, 15]. Horn-$\mathcal{SHIQ}$ is strictly more expressive than DLP and allows, for example, the free use of existential role restrictions.

* Research reported in this paper was supported by the EU in the IST project NeOn (IST-2006-027595, http://www.neon-project.org/)), by the Deutsche Forschungsgemeinschaft (DFG) under the ReaSem project, and by the the German Federal Ministry of Education and Research (BMBF) under the Theseus project, http://theseus-programm.de.

[3] Some recent developments can be found in [10, 12, 13].

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

Dagstuhl Seminar Proceedings 08091
Logic and Probability for Scene Interpretation
http://drops.dagstuhl.de/opus/volltexte/2008/1615

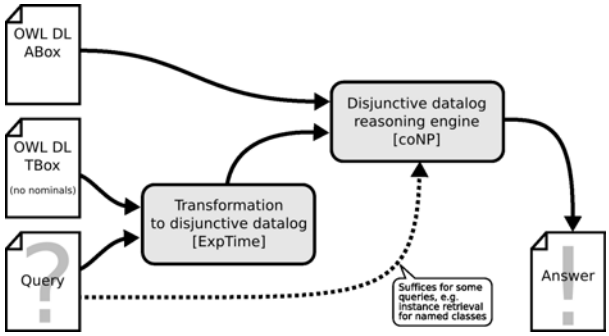

Fig. 1. KAON2 approach to reasoning

At the same time, a different effort to leveraging Horn logic for OWL reasoning rests on the idea of approximate reasoning, which presupposes an application scenario where speed is so important that it becomes reasonable to allow some incorrect inferences in order to speed up the reasoning. Our implementation is called SCREECH [5], and it is based on the idea of approximating an OWL DL knowledge base by Horn clauses. Initial experiments reported in [5] – and briefly in [18] – have shown that SCREECH indeed improves runtime in some cases, but further evaluations had been missing so far.

In this paper, we introduce two new variants of the SCREECH approach (in Section 3), resulting in three related algorithms, which can be used in combination for approximate OWL reasoning. We will then report on experiments (in Section 4) which we performed for all approaches. They show that all three variants of SCREECH indeed result in significant speed-up under only a very small number of introduced mistakes.

This paper is an extended abstract of [22].

2 Preliminaries: The KAON2-Transformation

Reasoning with KAON2 is based on special-purpose algorithms which have been designed for dealing with large ABoxes. They are detailed in [16] and we present a birds' eyes perspective here, which suffices for our purposes. The underlying rationale of the algorithms is that algorithms for deductive databases have proven to be efficient in dealing with large numbers of facts. The KAON2 approach utilises this by transforming OWL DL ontologies to disjunctive datalog, and by the subsequent application of the mentioned and established algorithms for dealing with disjunctive datalog.

A birds' eyes perspective on the KAON2 approach is depicted in Figure 1. KAON2 can handle $\mathcal{SHIQ}(\mathbf{D})$ ontologies, which corresponds roughly to OWL DL without nominals. The TBox, together with a query are processed by the sophisticated KAON2-transformation algorithm which returns a disjunctive datalog program. This, together with an ABox, is then fed into a disjunctive datalog reasoner which eventually returns an answer to the query.

In some cases, e.g. when querying for instances of named classes, the query does not need to be fed into the transformation algorithm but instead needs to be taken into

account only by the datalog reaoner. This allows to compute the disjunctive datalog program offline, such that only the disjunctive datalog engine needs to be invoked for answering the query. All experiments we report on have been performed this way, i.e. they assume an offline transformation of the TBox prior to the experiments.

The program returned by the transformation algorithm is in general not logically equivalent to the input TBox. The exact relationship is given below in Theorem 1 due to [16]. Note that statement (b) suffices for our purposes. It also shows that the KAON2 datalog reasoning engine can in principle be replaced by other (sound and complete) reasoning engines without changing the results of the inference process.

Theorem 1. *Let K be a $\mathcal{SHIQ}(\mathbf{D})$ TBox and $D(K)$ be the datalog output of the KAON2 transformation algorithm on input K. Then the following claims hold.*

(a) K is unsatisfiable if and only if $D(K)$ is unsatisfiable.
(b) $K \models \alpha$ if and only if $D(K) \models \alpha$, where α is of the form $A(a)$ or $R(a, b)$, for A a named concept and R a simple role.
(c) $K \models C(a)$ for a nonatomic concept C if and only if, for Q a new atomic concept, $D(K \cup \{C \sqsubseteq Q\}) \models Q(a)$.

Convenient access to the KAON2 transformation algorithm is given by means of the KAON2 OWL Tool[5] dlpconvert,[6] which can also produce F-Logic [8] serialisations which can be used with F-Logic engines like OntoBroker.

3 Variants of Screech

Due to the inherent high complexity of reasoning with ontologies, it is to be expected that some application settings will defy even the smartest approaches for achieving sound and complete scalable algorithms. The method of choice for dealing with such situations is to use approximate reasoning, which trades correctness for time, but in a controlled and well-understood way [2]. Approximate Reasoning is indeed recently receiving rising attention from Semantic Web researchers, due to the obvious suitable use cases in this application domain. For some recent work, see e.g. [21, 6, 3, 24, 17, 20].

The SCREECH approach for instance retrieval is based on the fact that data complexity is polynomial for non-disjunctive datalog, while for OWL DL it is coNP complete even in the absence of nominals [7]. SCREECH utilises the KAON2 algorithms, but rather than doing (expensive) exact reasoning over the resulting disjunctive datalog knowledge base, it does approximate reasoning by treating disjunctive rules as if they were non-disjunctive ones, i.e. the disjunctive rules are approximated by Horn rules.

We will first describe the basic variant of SCREECH, which was introduced in [5], and which we call SCREECH-ALL here. SCREECH-ALL is complete, but may be unsound in cases. Its data complexity is polynomial. Two other variants of SCREECH, SCHREECH-NONE and SCREECH-ONE, will be described subsequently.

[5] `http://owltools.ontoware.org/`
[6] `http://logic.aifb.uni-karlsruhe.de/wiki/Dlpconvert`

SCREECH-ALL uses a modified notion of *split program* [19] in order to deal with the disjunctive datalog. Given a rule

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

as an output of the KAON2 transformation algorithm, the *derived split rules* are defined as:

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

For a given disjunctive program P its *split program* P' is defined as the collection of all split rules derived from rules in P. It can be easily shown that for instance retrieval tasks, the result obtained by using the split program instead of the original one is complete but may be unsound. This is even the case if all integrity constraints, i.e. rules of the form

$$\leftarrow B_1, \ldots, B_n$$

are removed from the split program.

So SCREECH-ALL utilises the following subsequent steps for approximate ABox reasoning for $\mathcal{SHIQ}$.

1. Apply transformations as in the KAON2 system in order to obtain a negation-free disjunctive datalog program.
2. Obtain the split program as described above.
3. Do reasoning with the split program, e.g. using the KAON2 datalog engine.

Given a TBox K, the split program obtained from K by steps 1 and 2 is called the *screeched version* of K. 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 step 3. The resulting approach has the following theoretical features:

- It is complete with respect to OWL DL semantics.
- Data complexity is polynomial.

A prototype implementation of the approach is available as the SCREECH-ALL OWL approximate reasoner.[7] It is part of the KAON2 OWL Tools. We can convert a $\mathcal{SHIQ}$ ontology into a disjunctive datalog program, e.g. by using the KAON2 OWL Tool `dlpconvert` with the `-x` switch. SCREECH-ALL then accesses the results of the translation through the KAON2 API, creates the corresponding split programs and serializes them as Horn logic programs in Edinburgh Prolog syntax or in F-Logic [9, 1] syntax. We need to mention, however, that in general support for concrete domains and other features like integrity constraints is not necessarily implemented in off-the-shelf logic programming systems. In these cases, concrete domains etc. cannot be used. The KAON2 OWL Tool `ded`, for example, performs a language weakening step by removing all concrete domains, and may come in handy in such situations.

We will now introduce two other variants of SCREECH, besides SCREECH-ALL introduced above. These other variants are called SCREECH-NONE and SCREECH-ONE.

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

Table 1. SCREECH variants and their basic properties

variant	description	sound	complete
SCREECH-ALL	use *all* of the split rules	no	yes
SCREECH-NONE	use *none* of the split rules	yes	no
SCREECH-ONE	use *one* of the split rules	no	no

SCREECH-NONE is defined by simply removing all disjunctive rules (and all integrity constraints) after the transformation by the KAON2-algorithm. The resulting reasoning procedure is sound, but incomplete, on $\mathcal{SHIQ}$ knowledge bases.

SCREECH-ONE is defined by replacing each disjuntive rules by *exactly one* of the split rules. This selection can be done randomly, but will be most useful if the system has some knowledge – probably of statistical nature – on the size of the extensions of the named classes.[8] We also remove all integrity constraints after the translation. The resulting reasoning proceedure is neither sound nor complete.

Properties of SCREECH are summarised in Table 1.

4 Experimental Evaluation

An approximate reasoning procedure needs to be evaluated on real data from practical applications. Handcrafted examples are of only limited use as the applicability of approximate methods depends on the structure inherent in the experimental data.

So we evaluated some popular publicly available ontologies. In some cases we had to cautiously modify them in order to enable KAON2 to perform reasoning tasks on them, but the general approach was to first use KAON2 for transforming the TBoxes to disjunctive datalog. Also offline, a screeched version of the TBox was produced. We then invoked the KAON2 disjunctive datalog engine on both the resulting disjunctive datalog program and on the screeched version, to obtain a comparison of performance.

For all our experiments, we used a T60p IBM Thinkpad with 1.9GB of RAM, with the Java 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03).

We performed comprehensive experiments with GALEN, WINE, DOLCE, and SEMINTEC. The following is a summary of the results.

- SCREECH-ALL shows an average speedup in the range between 8 and 67%, depending on the queried class and the ontology under consideration, while 38 to 100% of the computed answers are correct. Most interestingly, a higher speedup usually seemed to correlate with less errors.
- SCREECH-ONE compared to SCREECH-ALL has overall *less* errors. In most cases, all correct class members are retrieved. Runtime is similar to SCREECH-ALL.
- SCREECH-NONE compared to SCREECH-ALL shows similar run-time. In most cases, the extensions are computed *correctly* – with the exception of WINE, for which we get 2% missing answers.

[8] This was suggested by Michael Sintek.

Table 2. Overview of SCREECH evaluations. Note that due to the completeness of SCREECH-ALL, the recall values are always 100% as well as the precision values for SCREECH-NONE due to its soundness. Moreover, the three SCREECH variants coincide in the case of the SEMINTEC ontology.

	SCREECH-ALL			SCREECH-ONE			SCREECH-NONE		
ontology	time saved	precision	recall	time saved	precision	recall	time saved	precision	recall
GALEN	74.6%	91.7%	100%	72.5%	97.4%	99.9%	76.5%	100%	99.8%
DOLCE	29.1%	62.1%	100%	17.5%	87.8%	100%	23.0%	100%	100%
WINE	34.5%	95.8%	100%	30.1%	98.0%	100%	28.6%	100%	97.7%
SEMINTEC	69.9%	100%	100%	66.0%	100%	100%	66.2%	100%	100%
VICODI	55.1%	100%	100%	54.1%	100%	100%	53.4%	100%	100%

– Running SCREECH-ALL and SCREECH-NONE in parallel and comparing the results, allows the following: If the computed extensions are of the same size, then we know that all (and only correct) class members have been found. This is the case for more than 76% of all classes we computed.

Overall, the obtained results are promising: the performance improvement is stable over all ontologies which we included in our experiments. The performance gain varied between 29.1 and 76.5%, while the amount of correctly retrieved classes was above 76% for all but one of the ontologies – see Table 2. It is encouraging to see that the approach appears to be feasible even for the sophisticated WINE ontology, and also for the SEMINTEC ontology, although in the latter case we only remove integrity constraints. Concerning the comparatively bad results on DOLCE, we note that the results are quite counterintuitive. One would naively expect that performance improvements go hand-in-hand with loss of precision. However, for DOLCE we measured both the least runtime improvement and the worst performance in terms of correctness. Concerning correctness, we suspect that the comparatively large number of incorrect answers is caused by the fact that DOLCE uses a large number of complete partitions of the form $A \equiv A_1 \sqcup \cdots \sqcup A_n$, where all the A_i are also specified to be mutually disjoint. It is intuitively clear that this kind of axioms introduces disjunctive (non-Horn-style and therefore harder to approximate) information on the one hand and integrity constraints (those being neglected in our approximation) on the other. However, this does not in itself explain why we did not observe a higher speedup. This indicates that the properties of ontologies which lead to performance improvement through screeching must be less straightforward than initially expected. For a clarification, more evaluations taking into account a wider range of ontologies with differing characteristics w.r.t. expressivity, used language features, or statistical measures like degree of population will lead to substantial hypotheses.

5 Conclusions

Motivated by the obvious need for techniques enhancing the scalability of reasoning related tasks, we have investigated three variants of the SCREECH approach to approx-

imate reasoning in OWL ontologies, and showed that we obtain a favourable trade-off between time saved and number of errors introduced.

References

1. Jürgen Angele and Georg Lausen. Ontologies in F-logic. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 29–50. Springer, 2004.
2. Dieter Fensel and Frank Van Harmelen. Unifying reasoning and search to web scale. *IEEE Internet Computing*, 11(2):94–96, March/April 2007.
3. Perry Groot, Heiner Stuckenschmidt, and Holger Wache. Approximating description logic classification for semantic web reasoning. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, volume 3532 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2005.
4. Benjamin Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logics. In *Proc. of WWW 2003, Budapest, Hungary, May 2003*, pages 48–57. ACM, 2003.
5. Pascal Hitzler and Denny Vrandecic. Resolution-based approximate reasoning for OWL DL. In Y. Gil et al., editors, *Proceedings of the 4th International Semantic Web Conference, Galway, Ireland, November 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 383–397. Springer, Berlin, 2005.
6. Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The Instance Store: DL reasoning with large numbers of individuals. In *Proceedings of the International Workshop on Description Logics, DL2004, Whistler, Canada*, pages 31–40, 2004.
7. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland*, pages 466–471, 2005.
8. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, July 1995.
9. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.
10. Markus Krötzsch, Pascal Hitzler, Denny Vrandecic, and Michael Sintek. How to reason with OWL in a logic programming system. In Thomas Eiter, Enrico Franconi, Ralph Hodgson, and Susie Stephens, editors, *Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web, RuleML2006*, pages 17–26, Athens, Georgia, 2006. IEEE Computer Society.
11. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity of Horn description logics. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence, AAAI-07, Vancouver, British Columbia, Canada, July 2007*, pages 452–457, 2007.
12. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Description Logic Rules. In *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI-08, Patras, Greece, July 2008*, pages 80–84. IOS Press, 2008.
13. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. ELP: Tractable rules for OWL 2. In *Proceedings of the 7th International Semantic Web Conference (ISWC2008)*, 2008. To appear.
14. Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview, February 2004. `http://www.w3.org/TR/owl-features/`.

15. B. Motik and U. Sattler. A comparison of reasoning techniques for querying large description logic ABoxes. In *Proc. of the 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006)*, Phnom Penh, Cambodia, November 2006.
16. Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe, 2006.
17. Jeff Z. Pan and Edward Thomas. Approximating OWL-DL ontologies. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1434–1439. AAAI Press, 2007.
18. Sebastian Rudolph, Markus Krötzsch, Pascal Hitzler, Michael Sintek, and Denny Vrandecic. Efficient OWL reasoning with logic programs – evaluations. In Massimo Marchiori, Jeff Z. Pan, and Christian de Sainte Marie, editors, *Proceedings of The First International Conference on Web Reasoning and Rule Systems 2007 (RR2007)*, volume 4524 of *Springer Lecture Notes in Computer Science*, pages 370–373. Springer, 2007.
19. 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.
20. Heiner Stuckenschmidt. Partial matchmaking using approximate subsumption. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1459–1464. AAAI Press, 2007.
21. Heiner Stuckenschmidt and Frank van Harmelen. Approximating terminological queries. In H.L. Larsen and et al, editors, *Proc. of the 4th International Conference on Flexible Query Answering Systems (FQAS)'02)*, Advances in Soft Computing. Springer, 2002.
22. Tuvshintur Tserendorj, Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Approximate OWL reasoning with Screech. In *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems, RR2008, Karlsruhe, Germany, October 2008*, 2008. To appear.
23. R. Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, Institute AIFB, University of Karlsruhe, 2004.
24. Holger Wache, Perry Groot, and Heiner Stuckenschmidt. Scalable instance retrieval for the semantic web by approximation. In Mike Dean, Yuanbo Guo, Woochun Jun, Roland Kaschek, Shonali Krishnaswamy, Zhengxiang Pan, and Quan Z. Sheng, editors, *Web Information Systems Engineering - WISE 2005 Workshops, WISE 2005 International Workshops, New York, NY, USA, November 20-22, 2005, Proceedings*, volume 3807 of *Lecture Notes in Computer Science*, pages 245–254. Springer, 2005.