Resolution-Based Approximate Reasoning for OWL DL*

Pascal Hitzler and Denny Vrandečić

AIFB, Universität Karlsruhe, Germany
{hitzler, vrandecic}@aifb.uni-karlsruhe.de

Abstract. We propose a new technique for approximate ABox reasoning with OWL DL ontologies. Essentially, we obtain substantially improved reasoning performance by disregarding non-Horn features of OWL DL. Our approach comes as a side-product of recent research results concerning a new transformation of OWL DL ontologies into negation-free disjunctive datalog [1, 2, 3, 4], and rests on the idea of performing standard resolution over disjunctive rules by treating them as if they were non-disjunctive ones. We analyse our reasoning approach by means of non-monotonic reasoning techniques, and present an implementation, called Screech.

1 Introduction

Knowledge representation and reasoning on the Semantic Web is done by means of ontologies. While the quest for suitable ontology languages is still ongoing, OWL [5] has been established as a 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 [6] 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 the 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. For the Semantic Web, TBox and RBox shall provide background vocabulary, while (annotated) webpages etc. constitute ABoxes which are interlinked with intensional knowledge. The Semantic Web thus envisions a distributed knowledge source, built from OWL ontologies and intertwining the knowledge like the World Wide Web interconnects websites.

* 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.

Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, pp. 383–397, 2005.

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* — thus 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 [7]. Methods are therefore being sought to cope with large ABoxes in an approximate manner.

The approach which we propose is based on the fact that data complexity is polynomial for non-disjunctive datalog. We utilise recent research results [1, 2, 3, 4] which allow the transformation of OWL DL ontologies into disjunctive datalog. Rather than doing (expensive) exact reasoning over the resulting disjunctive datalog knowledge base, we do approximate reasoning by treating disjunctive rules as if they were non-disjunctive ones. The resulting reasoning procedure is complete, but may be unsound in cases. Its data complexity is polynomial. We are also able to give a characterization of the resulting approximate inference by means of standard methods from logic programming semantics.

This paper is structured as follows. In Section 2, we first discuss the general rationale behind approximate reasoning, and how it relates to other reasoning frameworks. We then recall formal terminology and notation for OWL DL, and shortly review datalog and SLD-resolution. Then, in Section 4, we explain how OWL DL ontologies can be transformed into disjunctive datalog. In Section 5 we introduce the new approximate SLD-resolution procedure which we propose. The presentation of our implementation Screech in Section 6 is followed by an Example in Section 7, and an experimental evaluation in Section 8. We conclude and discuss future work in Section 9.

2 Non-classical Reasoning — Common Grounds

The sophisticated reasoning tasks required when dealing with expressive knowledge representation languages like those based on description logics are known to be of high computational complexity. In the face of ever increasing data quantities to be processed, new methods are needed to obtain usable systems. As the high computational complexity of the reasoning tasks is unavoidable, the method of choice for obtaining scalable systems is to use *approximate reasoning* techniques. In a nutshell, approximate reasoning rests on the idea of decreasing the complexity of a problem by imposing controlled changes on either the language used or the inference operation used for the deduction. The resulting lower complexity and consequent speed-up thus comes at the price of unsoundness or incompleteness (or both), but in a controlled and well-understood manner which allows to assess the quality of the deduction made by the approximate reasoner. So-called *anytime algorithms* develop the idea a bit further and guarantee convergence to exact answers given enough time, while providing approximate results during the reasoning process.

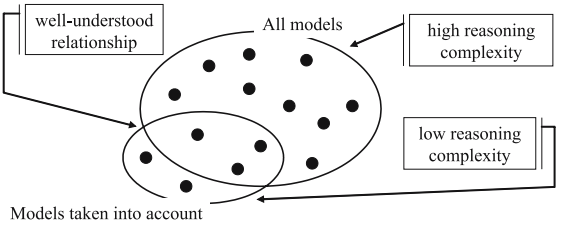

Fig. 1. Semantic view on approximate reasoning

A semantic perspective on approximate reasoning is depicted in Figure 1. When a theory is being considered, classical reasoning may be of high computational complexity and thus be unsuitable for time-critical tasks. By taking different models into account than the classical ones, the complexity of reasoning can be reduced. The resulting approximate inference may be incomplete or unsound with respect to classical inference, but in a controlled and well-understood manner, which makes the inferences suitable for further use.

Similar situations occur in the context of other sophisticated reasoning techniques. For non-monotonic reasoning, for example, a subset of the classical models is usually considered, which is selected by means of e.g. additional syntax constructs or by redefining the semantics of existing ones. Non-monotonic reasoning thus allows to arrive at conclusions which cannot be derived using classical reasoning: It is complete, but unsound, and can be described as *supraclassical* [8]. The rationale in this case is to model aspects of human commonsense reasoning like *jumping to conclusions*, again in a controlled and well-understood manner. Complexity considerations are often treated as secondary in this context.

Paraconsistent reasoning — or reasoning with inconsistency — can be approached from a similar perspective. While inconsistent knowledge bases have no classical models, paraconsistent reasoning strives to identify suitable models to be assigned to the knowledge base nevertheless, in order to allow the inference of meaningful consequences. As such, paraconsistent reasoning is sound, but incomplete with respect to classical logic, and can thus be termed *subclassical*.

Table 1 summarizes our discussion. While the table can certainly be extended further taking other forms of reasoning into account, we restrict ourselves to the mentioned examples, as the main goal of this paper is to present an approximate reasoning method for OWL DL, and not a comparative theory of reasoning

Table 1. Comparision of non-classical reasoning approaches

reasoning approach	focus	models taken into account	typical complexity
classical		all classical models	high
non-monotonic	commonsense	some classical models	very high
paraconsistent	inconsistency	more than the classical models	high
approximate	performance	variable	low

approaches. We have included this discussion because it explains the general rationale behind our approximate reasoning method, and will help us in analyzing it. Indeed, in all reasoning paradigms mentioned, it is important to obtain a clear understanding of the inference relation computed. This can be done by semantic analyses, i.e. by characterizations of the models taken into account. From the general perspective described in this section, it will later come as no surprise to the reader that we will analyze our approximate reasoning methods by means of standard techniques from non-monotonic reasoning. Indeed, in our particular case the models taken into account for approximate reasoning will turn out to be a subset of the classical models, as in non-monotonic reasoning.

3 Preliminaries

3.1 OWL DL Syntax and Semantics

OWL DL is a syntactic variant of the $\mathcal{SHOIN}(\mathbf{D})$ description logic [9]. Hence, although several XML and RDF syntaxes for OWL DL exist, it will be convenient to use the traditional description logic notation since it is more compact, and we recall the notation below. For the correspondence between this notation and various OWL DL syntaxes, see [9].

We indeed assume that the reader is familiar with OWL and thus with $\mathcal{SHOIN}(\mathbf{D})$, as space restrictions forbid to reintroduce them, but recall that $\mathcal{SHOIN}(\mathbf{D})$ supports reasoning with concrete datatypes, such as strings or integers [10]. Recall also that the description logic syntax for concepts in $\mathcal{SHOIN}(\mathbf{D})$ is defined as follows, where A is an atomic concept, R is an abstract role, S is an abstract simple role, $T_{(i)}$ are concrete roles, d is a concrete domain predicate, a_i and c_i are abstract and concrete individuals, respectively, and n is a non-negative integer:

$$\begin{aligned} C &\to A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \geq n\, S \mid \leq n\, S \mid \{a_1, \ldots, a_n\} \mid \\ &\quad \mid \geq n\, T \mid \leq n\, T \mid \exists T_1, \ldots, T_n.D \mid \forall T_1, \ldots, T_n.D \\ D &\to d \mid \{c_1, \ldots, c_n\} \end{aligned}$$

The $\mathcal{SHIQ}(\mathbf{D})$ description logic is obtained from $\mathcal{SHOIN}(\mathbf{D})$ by disallowing nominal concepts of the form $\{a_1, \ldots, a_n\}$ and $\{c_1, \ldots, c_n\}$, and by allowing qualified number restrictions of the form $\geq n\, S.C$ and $\leq n\, S.C$, for C a $\mathcal{SHIQ}(\mathbf{D})$ concept and S a simple role.

As description logics, $\mathcal{SHOIN}(\mathbf{D})$, i.e. OWL DL, and $\mathcal{SHIQ}(\mathbf{D})$ inherit their semantics from first-order logic by the standard translations known e.g. from [11], which we do not repeat here.

3.2 Datalog and SLD-Resolution

A (*definite* or *negation-free*) *disjunctive logic program* P consists of a finite set of *clauses* or *rules* of the form

$$\forall x_1 \ldots \forall x_n.(H_1 \vee \cdots \vee H_m \leftarrow A_1 \wedge \cdots \wedge A_k),$$

commonly written as

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

where $x_1, \ldots, x_n$ are exactly all variables occuring in $H_1 \vee \cdots \vee H_m \leftarrow A_1 \wedge \cdots \wedge A_k$, and all H_i and A_j are atoms over some given first-order language Σ. The disjunction $H_1 \vee \cdots \vee H_m$ is called the *rule head*, and the conjunction $A_1 \wedge \cdots \wedge A_k$ is called the *rule body*. The set of all ground instances of atoms defined over Σ is called the *Herbrand base* of P and is denoted by B_P. The set of all ground instances of rules in P is denoted by $\mathsf{ground}(P)$. A rule is said to be *non-disjunctive* if $m = 1$. It is called a *fact* if $k = 0$. We abstract from the order of the atoms in the heads respectively bodies; it is not important for our results. A disjunctive logic program is called a *(disjunctive) datalog* program if it does not contain function symbols.

Note that we do not consider logic programs to come with one specific semantics. Some people for example associate datalog with the minimal model semantics only. For our treatment, datalog and logic programs are defined via syntax only. We do not specify a specific semantics because in the following we will discuss *different* semantics for logic programs in their relation to proof procedures. One of the semantics we will consider is the semantics coming from interpreting logic programs as a set of first order formulas, and in this case we use $\models$ to denote entailment in classical first-order predicate logic.

SLD-resolution (see e.g. [12]) is an efficient top-down query-answering technique for programs consisting of non-disjunctive rules, and has been implemented and successfully applied in standard Prolog systems.[1] In this framework, a ground atom can be derived from a program if and only if it is true in the least (and thus in all) Herbrand models of the program.

In the following, we mean by a *conjunctive query* simply a conjunction $B_1 \wedge \cdots \wedge B_n$ of atoms. The query is called *ground* if it does not contain any variables.

Given a conjunctive query $B_1 \wedge \cdots \wedge B_n$, an *SLD-resolution step* on the atom B_i with a non-disjunctive rule $H \leftarrow A_1, \ldots, A_k$ produces a conjunctive query

$$B_1\theta \wedge \cdots \wedge B_{i-1}\theta \wedge A_1\theta \wedge \cdots \wedge A_k\theta \wedge B_{i+1}\theta \wedge \cdots \wedge B_n\theta$$

where θ is the most general unifier of B_i and H. An *SLD-refutation* of a conjunctive query $B_1 \wedge \cdots \wedge B_n$ in a non-disjunctive program P is a finite sequence of conjunctive queries $Q_0, \ldots, Q_n$, where (i) $Q_0 = B_1 \wedge \cdots \wedge B_n$, (ii) each Q_i with $i > 0$ is obtained from Q_{i-1} by an SLD-resolution step with some rule from P on some literal B_i, and (iii) $Q_n = \Box$, i.e. the conjunctive query Q_n does not contain any literals. If an SLD-refutation of $B_1 \wedge \cdots \wedge B_n$ in P exists, we write $P \vdash B_1 \wedge \cdots \wedge B_n$.

One of the fundamental results in logic programming states that $A \in B_P$ can be proven by SLD-resolution if and only if A is a logical consequence of P, i.e. if and only if A is true in the least Herbrand model of P:

[1] Like SWI or XSB Prolog, http://www.swi-prolog.org, http://xsb.sourceforge.net.

Theorem 1 ([12]). *For a ground conjunctive query $B_1 \wedge \cdots \wedge B_n$ and a non-disjunctive program P, $P \vdash B_1 \wedge \cdots \wedge B_n$ if and only if $P \models B_1 \wedge \cdots \wedge B_n$. In other words, entailment of ground conjunctive queries under SLD-resolution is entailment in predicate logic.*

SLD-resolution also allows to derive answers to non-ground queries: For a conjunctive (and not necessarily ground) query Q there exist an SLD-refutation if and only if $P \models \exists x_1 \ldots \exists x_n.Q$, where $x_1, \ldots, x_n$ are the variables occuring in Q. By keeping track of the most general unifiers used in the process, it is also possible to obtain bindings for (some of) the x_i in the form of (answer) substitutions θ, such that $P \models \exists y_1 \ldots \exists y_k(Q\theta)$, where the y_i are exactly those variables occurring in $Q\theta$. In order to keep our exhibition focused, we will only deal with ground queries.

4 Reducing OWL DL Knowledge Bases to Disjunctive Datalog Programs

We utilise recent research results about the transformation of OWL DL ontologies into disjunctive datalog, and perform 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 with this problem, a sophisticated method has been presented in [2, 3] 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 [2, 3] does not yet 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 reasoning algorithms for $\mathcal{SHOIN}(\mathbf{D})$ have been implemented. We will return to a possible treatment of nominals in our approach later.

The translation algorithm is schematically depicted in Figure 2. It transforms a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB into a disjunctive datalog program $\mathsf{DD}(KB)$. The steps of the algorithm are as follows. (1) Transitivity axioms are removed by adding axioms of a form similar to $\forall S.C \sqsubseteq \forall S.(\forall S.C)$ for transitive roles S. (2) The knowledge base is translated into clausal form by standard transformations based on first-order predicate logic. This introduces function symbols due

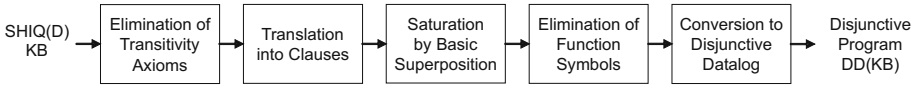

Fig. 2. Algorithm for Reducing $\mathcal{SHIQ}(\mathbf{D})$ to Datalog Programs

to necessary skolemization steps. (3) The TBox of the knowledge base is partially saturated by adding logical consequences. This is the crucial step of the algorithm. (4) The saturation from step (3) now allows to remove all function symbols which were introduced in step (2). Some additional axioms are added to ensure that the algorithm remains sound and complete. (5) The knowledge base is translated into disjunctive datalog clauses; this step is now straightforward.

It shall be noted that the details of the crucial step (3) are very sophisticated. They guarantee that the removal of function symbols in step (4) is at all possible. Step (3) is of exponential complexity, however for the ABox reasoning task which we focus on in this paper, Step (3) can in principle be performed offline, as this step is independent of the ABox – but note that this offline computation may still be difficult if the TBox is large, which is a seperate issue and deserves further in-depth studies which are outside the scope of this paper. A full presentation of the translation with correctness proofs is technically involved and lengthy, and space restrictions forbid to go into further detail; we refer the interested reader to [2, 3]. In [1] full proofs are given which show amongst other things that KB is unsatisfiable if and only if $\mathsf{DD}(KB)$ is unsatisfiable. This suffices for reasoning over KB as reasoning tasks can be transformed into unsatisfiability checks.

5 Approximate Resolution

While approximate reasoning methods for propositional and first-order logic have been proposed (see e.g. [13, 14, 15, 16, 17, 18]), they have hardly been applied in the context of Semantic Web technologies. The few exceptions are reported e.g. in [19, 20, 21] — to the best of our knowledge, this list is exhaustive. The success of the approaches is mixed. [21] reports on an analysis indicating that straightforward adaptations of methods proposed by [14] do not suffice. [20] reports good results but is not an approximate reasoning method in the more narrow sense as the reasoning performed is exact, and thus does not address the complexity problems underlying OWL DL reasoning. [19] deals with approximating queries, while we focus on ABox reasoning. We will now present a novel approach based on the translation of OWL DL to disjunctive datalog, as presented earlier.

5.1 Approximate SLD-Resolution

Having obtained the translated knowledge base in the form of a disjunctive datalog program, ABox reasoning remains NP-hard, and thus untractable. If the datalog program is non-disjunctive, though, reasoning is polynomial in the size of

the ABox. We therefore propose the following approximate reasoning technique in order to facilitate this insight. Given a conjunctive query $B_1 \wedge \cdots \wedge B_n$, an *approximate SLD-resolution step* on the atom B_i with a disjunctive rule $H_1 \vee \cdots \vee H_m \leftarrow A_1, \ldots, A_k$ is a conjunctive query

$$B_1\theta \wedge \cdots \wedge B_{i-1}\theta \wedge A_1\theta \wedge \cdots \wedge A_k\theta \wedge B_{i+1}\theta \wedge \cdots \wedge B_n\theta$$

such that θ is the most general unifier of B_i and some H_j. *Approximate SLD-refutation* is defined analogously to SLD-refutation, where approximate SLD-resolution steps are used instead of (usual) SLD-resolution steps.

It is necessary to pursue the question what notion of entailment underlies the approximate reasoning technique we propose. Following the spirit of the obervations from Section 2, we want to identify the set of models which underly the inference relation provided by approximate SLD-resolution. For this purpose, we need the following notion, which is derived from standard notions in non-monotonic reasoning over logic programs.

Definition 1 (cf. [22, 23, 24]). *A model M of a disjunctive program P is called* well-supported *if there exists a function $l : B_P \to \mathbb{N}$ such that for each $A \in M$ there exists a rule $A \vee H_1 \vee \cdots \vee H_m \leftarrow A_1, \ldots, A_k$ in $\mathsf{ground}(P)$ with $M \models A_i$ and $l(A) > l(A_i)$ for all i and k.*

Definition 1 is a straightforward adaptation of the notion of well-supported model for non-disjunctive programs, as given in [23]. For non-disjunctive (and negation-free) programs, the well-supported models are exactly the minimal ones, but this is not in general the case for disjunctive programs: Just consider the program consisting of the single rule $p \vee q \leftarrow$. Then $\{p, q\}$ is a well-supported model, but is not minimal.

Lifted appropriately to (non-disjunctive) programs with negation, the well-supported models coincide with the well-known stable models. This was shown in [23] and studied in-depth in [24, 25]. Stable models [26] provide the base for the most popular non-monotonic reasoning paradigm called *Answer Set Programming*, of which the two most prominent implementations are DLV and SMODELS [27, 28]. Our results thus stand within this well-established tradition.

It is apparent that $A \in B_P$ is entailed by a (disjunctive) program P by approximate SLD-resolution if and only if it is true in at least one well-supported model of P. This is called *brave reasoning with well-supported models*. A formal proof of the following proposition is omitted for space restrictions.

Proposition 1. *Entailment of ground conjunctive queries under approximate SLD-resolution is brave reasoning with well-supported models.*

As an example, consider the (propositional) program consisting of the two rules $p \vee q \leftarrow$ and $r \leftarrow p \wedge q$. Its minimal models are $\{q\}$ and $\{p\}$, so r is not bravely entailed by reasoning with minimal models. However all of $\{q\}$, $\{p\}$, $\{p, q\}$ and $\{p, q, r\}$ are well-supported models, so r is bravely entailed by reasoning with well-supported models.

There is an alternative way of formalizing approximate SLD-resolution using a modified notion of *split program* [29]. Given a rule

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

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. Approximate SLD-resolution on P is obviously identical to SLD-resolution over P'.

Minimal models are well-supported, as can be seen from the following result which was obtained along the lines of research laid out in [24, 25].

Theorem 2 ([30]). *Let P be a disjunctive program. Then a model M of P is a minimal model of P if and only if there exists a function $l : B_P \to \mathbb{N}$ such that for each A which is true in M there exists a rule $A \vee H_1 \vee \cdots \vee H_m \leftarrow A_1, \ldots, A_k$ in $\mathsf{ground}(P)$ with $M \models A_i$, $M \not\models H_k$ and $l(A) > l(A_i)$ for all i and k.*

We hence have the following result, noting that $P \models Q$ for any ground conjunctive query Q and program P if and only if Q is true in all minimal models of P.

Proposition 2. *Let P be a (possibly disjunctive) program and Q be a ground conjunctive query with $P \models Q$. Then there exists an approximate SLD-refutation for Q.*

We remark that for negation-free disjunctive programs minimal models again coincide with *answer sets* [26], as in the currently evolving *Answer Set Programming Systems*, as already mentioned.

5.2 Approximate Resolution for OWL DL

Our proposal is based on the idea of converting a given OWL DL knowledge base into a function-free definite disjunctive logic program, and then to apply approximate resolution for ABox reasoning.

In order to be able to deal with all of OWL DL, we need to add a preprocessing step to get rid of nominals, i.e. we need to compile $\mathcal{SHOIN}(\mathbf{D})$ ontologies to $\mathcal{SHIQ}(\mathbf{D})$. 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, so some information is lost in the process. Putting all the pieces together, we propose the following subsequent steps for 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 4 in order to obtain a negation-free disjunctive datalog program.
3. Apply 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 wrt. brave reasoning with well-supported models.
- Data complexity of our approach is polynomial.

6 Screech OWL

A preliminary implementation of our approach is available as the Screech OWL approximate reasoner.[2] It is part of the KAON2 OWL tools.[3]

KAON2[4] is the KArlsruhe ONtology framework, which includes a fast OWL reasoner based on the transformation algorithms mentioned in Section 4, and also includes many other features helpful to work with ontologies. Among the KAON2 OWL tools, `deo` performs the language weakening step described in Section 5.2 in order to obtain a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base. As KAON2 implements the sophisticated translation algorithms described in Section 4, we can convert an OWL ontology into a disjunctive datalog program, e.g. by using the `dlpconvert` KAON2 OWL tool with the `-x` switch.

Screech then accesses the results of the translation through the KAON2 API, creates the corresponding split programs and serialises them as Horn logic programs in Edinburgh Prolog syntax. The result can be fed to any Prolog interpreter — or other logic programming engine —, which in turn can be used to perform ABox reasoning and inferencing over the knowledge base.

For completeness, we need to mention 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`,[3] for example, performs a language weakening step by removing all concrete domains, and may come in handy in such situations.

7 An Example

We demonstrate our approach by means of a simple OWL DL ontology. It contains only a class hierarchy and an ABox, and no roles, but this will suffice to display the main issues.

[2] http://logic.aifb.uni-karlsruhe.de/screech
[3] http://www.aifb.uni-karlsruhe.de/WBS/dvr/owltools
[4] http://kaon2.semanticweb.org

$$\begin{aligned}
\text{serbian} \sqcup \text{croatian} &\sqsubseteq \text{european} \\
\text{eucitizen} &\sqsubseteq \text{european} \\
\text{german} \sqcup \text{french} \sqcup \text{beneluxian} &\sqsubseteq \text{eucitizen} \\
\text{beneluxian} &\equiv \text{luxembourgian} \sqcup \text{dutch} \sqcup \text{belgian}
\end{aligned}$$

serbian(ljiljana)	serbian(nenad)	german(pascal)	french(julien)
croatian(boris)	german(markus)	german(stephan)	croatian(denny)
indian(sudhir)	belgian(saartje)	german(rudi)	german(york)

Fig. 3. Example ontology

The ontology is shown in Figure 3, and its intended meaning is self-explanatory. Note that the fourth line,

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

translates into the four clauses

$$\begin{aligned}
\text{luxembourgian}(x) \vee \text{dutch}(x) \vee \text{belgian}(x) &\leftarrow \text{beneluxian}(x), \qquad (1) \\
\text{beneluxian}(x) &\leftarrow \text{luxembourgian}(x), \\
\text{beneluxian}(x) &\leftarrow \text{dutch}, \\
\text{and} \qquad \text{beneluxian}(x) &\leftarrow \text{belgian}(x).
\end{aligned}$$

Thus, our approach changes the ontology by treating the disjunctions in line (1) as conjunctions. This change affects the soundness of the reasoning procedure. However, most of the ABox consequences which can be derived by approximate SLD-resolution are still correct. Indeed, there are only two derivable facts which do not follow from the knowledge base by classical reasoning, namely

$$\text{dutch(saartje)} \qquad \text{and} \qquad \text{luxemburgian(saartje)}.$$

All other derivable facts are correct.

Screech translates the ontology from Figure 3 into the Prolog program listed in Figure 4. As standard implementations of SLD-resolution do not use fair selection functions and also use depth-first search for higher efficiency, they may sometimes fail to produce answers because they run into infinite branches of the search tree. This occurs, for example, when using SWI-Prolog[5]. A reordering of the clauses may improve the results, but does not solve the problem entirely. More satisfactory performance can be obtained by using SLD-resolution with tabling, as implemented e.g. in the XSB Prolog system[6]. In this case, all desired consequences can be derived.

[5] http://www.swi-prolog.org/
[6] http://xsb.sourceforge.net

```
serbian(ljiljana).  serbian(nenad).    german(pascal).   french(julien).
croatian(boris).    german(markus).    german(stephan).  croatian(denny).
indian(sudhir).     belgian(saartje).  german(rudi).     german(york).
european(X)         :- serbian(X).
european(X)         :- croatian(X).
european(X)         :- eucitizen(X).
eucitizen(X)        :- german(X).
eucitizen(X)        :- french(X).
eucitizen(X)        :- beneluxian(X).
beneluxian(X)       :- luxembourgian(X).
beneluxian(X)       :- dutch(X).
beneluxian(X)       :- belgian(X).
dutch(X)            :- beneluxian(X).
luxembourgian(X)    :- beneluxian(X).
belgian(X)          :- beneluxian(X).
```

Fig. 4. Example SCREECH output

8 Experiments and 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.

For our evaluation we have performed experiments with the OWL DL version of the GALEN Upper Ontology,[7] 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.[8] 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.[9] 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 2, 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.

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

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

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

Table 2. Performance comparison for instance retrieval using disjunctive datalog (DD) vs. the corresponding split program (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 class in the last row is Biological_object_that_has_left_right_symmetry.

Time (DD)	Time (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_part_of_heart
11036 ms	7711 ms	4/4	Human_red_blood_cell_mature
11055 ms	5949 ms	24/58	Biological_object_that...

The times were obtained with initial Java VM memory set to 256 MByte. Under memory restrictions, the speed-up is more significant, which is probably caused by the necessity to allocate additional memory for the DD reasoning task. Corresponding figures are given in Table 3. Our experiments also indicate that SCREECH may be useful when hardware is limited, for example in portable devices.

9 Conclusions and Further Work

In a nutshell, our proposed procedure approximates reasoning by disregarding non-Horn features of OWL DL ontologies. We argue that this is a reasonable approach to approximate reasoning with OWL DL in particular because many of the currently existing ontologies rarely use language constructs that do not fall into the Horn fragment of OWL DL [31]. So it can be projected that even in the future these constructs will play a minor role and thus should be the first to be tempered with in order to gain tractable reasoning.

Our approach provides ABox reasoning with polynomial time complexity. While it is complete, it is also unsound with respect to first-order logic. We have shown, however, that the inference underlying our approach can be characterized using standard methods from the area of non-monotonic reasoning. We have also presented our implementation SCREECH, and verified the usefulness of our approach by means of experiments.

Table 3. Performance comparison as in Table 2, but with 128 MByte intial memory

Time (DD)	Time (SPLIT)	Instances	Class Name
32997 ms	4817 ms	154/154	Biological_object
33028 ms	4947 ms	9/9	Specified_set
32927 ms	4987 ms	9/13	Multiple
32977 ms	4957 ms	16/16	Probe_structural_part_of_heart
32987 ms	7350 ms	4/4	Human_red_blood_cell_mature
32947 ms	4796 ms	24/58	Biological_object_that...

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. Theorem 2 suggests how an anytime algorithm for this might be obtained: After performing approximate SLD-resolution, it remains to be checked whether there is any (ground instance of a) rule used in the refutation of the query, which has an atom A in its head besides the one used in the refutation and such that A is (cautiously) entailed by the program. Such an algorithm might then first find a brave proof of a query, and then substantiate this proof by subsequent calculations. Our approach may also be useful for the quick derivation of *possible* answers to a query, which may then be used for efficient guidance of the search within a sound and complete OWL reasoner. These and other issues are currently under investigation.

Acknowledgement. We are grateful for discussions with Boris Motik about his and our work.

References

1. Hustadt, U., Motik, B., Sattler, U.: Reasoning for Description Logics around $\mathcal{SHIQ}$ in a Resolution Framework. Technical Report 3-8-04/04, FZI, Karlsruhe, Germany (2004) http://www.fzi.de/wim/publikationen.php?id=1172.
2. Hustadt, U., Motik, B., Sattler, U.: Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In: Proc. of the 9th Conference on Knowledge Representation and Reasoning (KR2004), AAAI Press (2004)
3. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics with a concrete domain in the framework of resolution. In de Mántaras, R.L., Saitta, L., eds.: Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004, IOS Press (2004) 353–357
4. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. In: Proceedings of the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, November 2004. (2004) To appear.
5. W3C: Web ontology language (OWL). www.w3.org/2004/OWL/ (2004)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)
7. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In Kaelbling, L.P., Saffiotti, A., eds.: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland. (2005) 466–471
8. Makinson, D.: Bridges from Classical to Nonmonotonic Logic. Volume 5 of Texts in Computing. King's College Publications, London (2005)
9. Horrocks, I., Patel-Schneider, P.F.: A Proposal for an OWL Rules Language. In: Proc. of the Thirteenth Int'l World Wide Web Conf.(WWW 2004), ACM (2004)
10. Lutz, C.: Description Logics with Concrete Domains — A Survey. In: Advances in Modal Logics. Volume 4., King's College Publications (2003)
11. Horrocks, I., Sattler, U., Tobies, S.: Practical Reasoning for Very Expressive Description Logics. Logic Journal of the IGPL **8** (2000) 239–263
12. Lloyd, J.W.: Foundations of Logic Programming. Springer, Berlin (1988)

13. Selman, B., Kautz, H.A.: Knowledge compilation using Horn approximations. In: Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91). (1991) 904–909
14. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artificial Intelligence **74** (1995) 249–310
15. Dalal, M.: Anytime clausal reasoning. Annals of Mathematics and Artificial Intelligence **22** (1998) 297–318
16. Cadoli, M., Scarcello, F.: Semantical and computational aspects of Horn approximations. Artificial Intelligence **119** (2000)
17. van Harmelen, F., ten Teije, A.: Describing problem solving methods using anytime performance profiles. In: Proceedings of ECAI'00, Berlin (2000) 181–186
18. Groot, P., ten Teije, A., van Harmelen, F.: Towards a structured analysis of approximate problem solving: a case study in classification. In: Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR'04), Whistler, Colorado (2004)
19. Stuckenschmidt, H., van Harmelen, F.: Approximating terminological queries. In Larsen, H., et al, eds.: Proc. of the 4th International Conference on Flexible Query Answering Systems (FQAS)'02). Advances in Soft Computing, Springer (2002)
20. Horrocks, I., Li, L., Turi, D., Bechhofer, S.: The Instance Store: DL reasoning with large numbers of individuals. In: Proceedings of the International Workshop on Description Logics, DL2004, Whistler, Canada. (2004) 31–40
21. Groot, P., Stuckenschmidt, H., Wache, H.: Approximating description logic classification for semantic web reasoning. In Gómez-Pérez, A., Euzenat, J., eds.: 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., Springer (2005) 318–332
22. Apt, K.R., Blair, H.A., Walker, A.: Towards a theory of declarative knowledge. In Minker, J., ed.: Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann, Los Altos, CA (1988) 89–148
23. Fages, F.: Consistency of Clark's completion and existence of stable models. Journal of Methods of Logic in Computer Science **1** (1994) 51–60
24. Hitzler, P., Wendt, M.: A uniform approach to logic programming semantics. Theory and Practice of Logic Programming **5** (2005) 123–159
25. Hitzler, P.: Towards a systematic account of different semantics for logic programs. Journal of Logic and Computation **15** (2005) 391–404
26. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9** (1991) 365–385
27. Eiter, T., Leone, N., Mateis, C., Pfeifer, G., Scarcello, F.: A deductive system for nonmonotonic reasoning. In Dix, J., et al, eds.: Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97). Volume 1265 of Lecture Notes in Artificial Intelligence., Springer, Berlin (1997)
28. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138** (2002) 181–234
29. Sakama, C., Inoue, K.: An alternative approach to the semantics of disjunctive logic programs and deductive databases. Journal of Automated Reasoning **13** (1994) 145–172
30. Knorr, M.: Level mapping characterizations for quantitative and disjunctive logic programs. Bachelor's Thesis, Department of Computer Science, Technische Universität Dresden, Germany (2003)
31. Volz, R.: Web Ontology Reasoning with Logic Databases. PhD thesis, AIFB, University of Karlsruhe (2004)