# Orel: Database-Driven Reasoning for OWL 2 Profiles

Markus Krötzsch, Anees ul Mehdi, and Sebastian Rudolph

Institute AIFB, Karlsruhe Institute of Technology, DE
{mak,ame,sru}@aifb.uni-karlsruhe.de

**Abstract.** We describe Orel, a reasoning system for an ontology language which subsumes both the EL and the RL profile of the recently standardised web ontology language OWL 2. Orel performs consequence-driven reasoning on the database level which is always sound. It is guaranteed to be complete if the ontology is contained in one of the two profiles. We present the underlying calculus, the core algorithm, and initial evaluation results.

## 1 Introduction

With the standardisation of the Web Ontology Language OWL 2 in 2009 [1], the development of theoretically well-studied and practically deployable expressive ontology languages for the Semantic Web has reached a new level of maturity. Among various other improvements, the new version of OWL is the first that adequately addresses the trade-off between logical expressivity and scalability that is inherent to formal knowledge representation by specifying additional light-weight language profiles. The three OWL 2 profiles EL, RL, and QL constitute sublanguages which – while still sufficiently expressive for many applications – exhibit a polynomial time complexity for standard reasoning tasks, and are therefore particularly suitable for working with large ontological descriptions [2].

The Orel software that is introduced in this system description provides storage and reasoning services for both OWL EL and RL. The specific features that set it apart from existing implementations are twofold. First, its implementation is tailored toward materialisation of entailments in a persistent storage backend such as a relational database management system (DBMS). Second, it realises a rule-based approach for implementing both OWL RL and OWL EL inferencing in a single polytime algorithm.

Orel's approach to reasoning is to express inference tasks for OWL 2 in terms of inference tasks for the simple rule language datalog [3]. The basis of this method is an entailment-preserving translation of description logics to datalog that has been introduced in [4]. The latter approach has been presented for a hybrid ontology-rule language that includes features which cannot be expressed in OWL 2. This provides an interesting path for extending Orel to also cover some of the expressivity of rule languages like SWRL [5] or RIF-BLD [6], but the present paper focusses on the supported OWL 2 features only.

In Section 2, we discuss Orel's inferencing calculus, and present some optimisations for data-centric processing. Thereafter, in Section 3, we briefly highlight our basic approach for extending this inferencing mechanism to efficient schema inferencing, and

$$PhD \sqsubseteq AcademicDegree \qquad PostDoc \sqsubseteq \exists has.PhD \qquad Graduate \equiv \exists has.AcademicDegree$$

$$\begin{array}{rr}
PhD(x) \to AcademicDegree(x) & Graduate(x) \to has(x, d_{\exists has.AD}) \\
PostDoc(x) \to has(x, d_{\exists has.PhD}) & Graduate(x) \to AcademicDegree(d_{\exists has.AD}) \\
PostDoc(x) \to PhD(d_{\exists has.PhD}) & has(x, y) \wedge AcademicDegree(y) \to Graduate(x)
\end{array}$$

**Fig. 1.** Example translation to datalog

in Section 4, we recall the general techniques for adapting a rule-based calculus for execution in a relational DBMS. Section 5 provides further details on the implementation and initial evaluation results. We discuss related work in Section 6 and give an outlook to the future development of Orel in Section 7. Orel is free software that can be obtained at `http://code.google.com/p/orel/`.

## 2 A Data-Driven Approach for Translating OWL into Datalog

The algorithms in [4] extend to a first-order knowledge representation language dubbed ELP that combines features of the description logic $\mathcal{EL}^{++}$ [7], Description Logic Rules [8], and DL-safe Rules [9]. Yet, the expressivity of ELP has been restricted sufficiently to allow for polynomial-time reasoning. Instead of repeating the formal details that can readily be found in [4], we summarise the underlying approach by means of a brief example, and provide more detailed descriptions of the algorithms that are actually implemented in Orel. Throughout this work, we use description logic syntax for concisely expressing the semantics of OWL 2 axioms.

As an example, consider the set of OWL 2 axioms in Fig. 1 (top). Following a strategy as in [4], this knowledge base would be translated into the rule set in Fig. 1 (bottom). These rules are intended to be read as first-order implications based on a standard predicate logic semantics.[1] Note that the translation is faithful regarding the signature: OWL classes are translated into unary predicates, and OWL properties into binary predicates. Thus it is not hard to see how axioms from the original ontology relate to implications in the translated datalog program.

While this translation is straightforward in many cases, a special approach is needed to cover existential expressions as in `ObjectSomeValuesFrom`. Since datalog does not allow existential entailments, auxiliary constants are introduced to represent additional "anonymous" individuals the existence of which is required by the ontology. Please note that only a single constant is introduced for affected class expressions during the translations. This limits the amount of additional individuals that need to be considered, and it is vital to retain polytime complexity.

While the above translation is rather intuitive for the most part, the presented encoding has several practical drawbacks that come to the fore when attempting an actual implementation. In particular, the created rule set may become rather large; it grows linearly with the size of the knowledge base. However, typical LP engines exhibit far

[1] Since we are only interested in positive entailments, assuming a non-monotonic semantics for datalog would not lead to different inference results. See [3] for details.

| | | | | | |
|---|---|---|---|---|---|
| $A(n)$ | $\mapsto$ | $\texttt{inst}(n, \hat{A})$ | $A \sqsubseteq C$ | $\mapsto$ | $\texttt{subClass}(\hat{A}, \hat{C})$ |
| $R(n, m)$ | $\mapsto$ | $\texttt{triple}(n, R, m)$ | $A \sqcap B \sqsubseteq C$ | $\mapsto$ | $\texttt{subIntersect}(\hat{A}, \hat{B}, \hat{C})$ |
| $\exists R.\mathsf{Self}(n)$ | $\mapsto$ | $\texttt{self}(n, R)$ | | | |
| $\exists R.A \sqsubseteq C$ | $\mapsto$ | $\texttt{subSomeValues}(R, \hat{A}, \hat{C})$ | $\exists R.\mathsf{Self} \sqsubseteq C$ | $\mapsto$ | $\texttt{selfImplies}(R, \hat{C})$ |
| $A \sqsubseteq \exists R.B$ | $\mapsto$ | $\texttt{someValues}(\hat{A}, R, \hat{B}, d_{\exists R.B})$ | $A \sqsubseteq \exists R.\mathsf{Self}$ | $\mapsto$ | $\texttt{impliesSelf}(\hat{A}, R)$ |
| $A \sqsubseteq \forall R.B$ | $\mapsto$ | $\texttt{allValuesFrom}(\hat{A}, R, \hat{B})$ | $A \sqsubseteq \leqslant 1\, R.B$ | $\mapsto$ | $\texttt{atMostOne}(\hat{A}, R, \hat{B})$ |
| $R \sqsubseteq T$ | $\mapsto$ | $\texttt{subProperty}(R, T)$ | $\mathsf{Disj}(R, S)$ | $\mapsto$ | $\texttt{disjoint}(R, S)$ |
| $R \circ S \sqsubseteq T$ | $\mapsto$ | $\texttt{subPropertyChain}(R, S, T)$ | $R^{-} \sqsubseteq S$ | $\mapsto$ | $\texttt{subInverseOf}(R, S)$ |
| For each individual name $n$ in the ontology, add the fact $\texttt{nom}(n)$ to the transformation. | | | | | |
| For each class name or nominal $A$ in the ontology, add the fact $\texttt{subClass}(A, \top)$. | | | | | |

**Fig. 2.** Creating an initial fact base from DL axioms in Orel; for a class $C$ define $\hat{C} := n$ if $C = \{n\}$ is a nominal class, and $\hat{C} := C$ if $C$ is a class name, $\top$, or $\bot$

better performance when more facts and less rules are given. Similarly, DBMS can handle large amounts of data while implications in the above formulation work on this data and would thus correspond to database operations. The next section therefore introduces a modified approach that is taken in Orel. This observation calls for a different encoding strategy, where ontological information (such as subclass relationships) is stored as facts, while logical ramifications are governed by "meta-rules" that resemble the rules of a deduction calculus. Thereby, classes and properties have to be treated as datalog individuals. The above example might then be encoded by the following facts:

$$\begin{aligned}
\texttt{subClass}(\mathit{PhD}, \mathit{AcademicDegree})\\
\texttt{someValues}(\mathit{PostDoc}, \mathit{Has}, \mathit{PhD}, d_{\exists \mathit{has.PhD}})\\
\texttt{someValues}(\mathit{Graduate}, \mathit{Has}, \mathit{AcademicDegree}, d_{\exists \mathit{has.AD}})\\
\texttt{subSomeValues}(\mathit{Has}, \mathit{AcademicDegree}, \mathit{Graduate})
\end{aligned}$$

The predicate names used here hint at the intended interpretation but are not formally related to the OWL 2 vocabulary. Note that the auxiliary constants $d_{\exists \mathit{has.PhD}}$ and $d_{\exists \mathit{has.AD}}$ are already included in the above facts. Since we are interested in a rule set without function symbols (datalog), all required constant symbols must be explicitly created beforehand. We now can encode the intended semantics in derivation rules such as the following:

$$\begin{aligned}
\texttt{subClass}(a, b) \wedge \texttt{inst}(x, a) &\rightarrow \texttt{inst}(x, b)\\
\texttt{someValues}(a, r, b, d) \wedge \texttt{inst}(x, a) &\rightarrow \texttt{triple}(x, r, d)\\
\texttt{someValues}(a, r, b, d) \wedge \texttt{inst}(x, a) &\rightarrow \texttt{inst}(d, b)\\
\texttt{subSomeValues}(r, a, b) \wedge \texttt{triple}(x, r, y) \wedge \texttt{inst}(y, a) &\rightarrow \texttt{inst}(x, b),
\end{aligned}$$

Here we encode assertions about instances in the obvious way with the additional meta-predicates `inst` for class instances, and `triple` for role assertions. All terms in the above rules are variables, but here and below we use different letters for capturing the underlying intuition: $a$, $b$, $c$ for class names, $r$, $s$, $t$ for role names, $x$, $y$, $z$ for individual names, and $d$ for auxiliary constants.

As in the above example, most features of OWL EL and RL can be supported by suitable meta-rules based on the datalog translation in [4]. For the most prominent features of the two profiles, the translation of axioms to meta-facts is specified in Fig. 2, and the according materialisation rules are presented in Fig. 3. The translation assumes

$$\mathtt{nom}(x) \to \mathtt{inst}(x, x) \tag{1}$$
$$\mathtt{nom}(x) \wedge \mathtt{triple}(x, r, x) \to \mathtt{self}(x, r) \tag{2}$$
$$\mathtt{subClass}(a, b) \wedge \mathtt{inst}(x, a) \to \mathtt{inst}(x, b) \tag{3}$$
$$\mathtt{subIntersect}(a, b, c) \wedge \mathtt{inst}(x, a) \wedge \mathtt{inst}(x, b) \to \mathtt{inst}(x, c) \tag{4}$$
$$\mathtt{subSomeValues}(r, a, c) \wedge \mathtt{triple}(x, r, y) \wedge \mathtt{inst}(y, a) \to \mathtt{inst}(x, c) \tag{5}$$
$$\mathtt{someValues}(a, p, b, d) \wedge \mathtt{inst}(x, a) \to \mathtt{triple}(x, p, d) \tag{6}$$
$$\mathtt{someValues}(a, p, b, d) \wedge \mathtt{inst}(x, a) \to \mathtt{inst}(d, b) \tag{7}$$
$$\mathtt{selfImplies}(r, c) \wedge \mathtt{self}(x, r) \to \mathtt{inst}(x, c) \tag{8}$$
$$\mathtt{impliesSelf}(a, r) \wedge \mathtt{inst}(x, a) \to \mathtt{self}(x, r) \tag{9}$$
$$\mathtt{impliesSelf}(a, r) \wedge \mathtt{inst}(x, a) \to \mathtt{triple}(x, r, x) \tag{10}$$
$$\mathtt{subProperty}(r, t) \wedge \mathtt{triple}(x, r, y) \to \mathtt{triple}(x, t, y) \tag{11}$$
$$\mathtt{subProperty}(r, t) \wedge \mathtt{self}(x, r) \to \mathtt{self}(x, t) \tag{12}$$
$$\mathtt{subPropertyChain}(r, s, t) \wedge \mathtt{triple}(x, r, y) \wedge \mathtt{triple}(y, s, z) \to \mathtt{triple}(x, t, z) \tag{13}$$
$$\mathtt{disjoint}(r, s) \wedge \mathtt{triple}(x, r, y) \wedge \mathtt{triple}(x, s, y) \to \mathtt{inst}(x, \bot) \tag{14}$$
$$\mathtt{inst}(x, y) \wedge \mathtt{nom}(y) \to \mathtt{inst}(y, x) \tag{15}$$
$$\mathtt{inst}(x, y) \wedge \mathtt{nom}(y) \to \mathtt{nom}(x) \tag{16}$$
$$\mathtt{triple}(x1, r, y) \wedge \mathtt{inst}(x2, y) \wedge \mathtt{nom}(y) \to \mathtt{triple}(x1, r, x2) \tag{17}$$
$$\begin{aligned}&\mathtt{allValuesFrom}(a, r, b) \wedge \mathtt{nom}(x) \wedge \mathtt{nom}(y) \wedge \\ &\quad \mathtt{triple}(x, r, y) \wedge \mathtt{inst}(x, a) \to \mathtt{inst}(y, b)\end{aligned} \tag{18}$$
$$\begin{aligned}&\mathtt{atMostOne}(a, r, b) \wedge \mathtt{nom}(x) \wedge \mathtt{nom}(y_1) \wedge \mathtt{nom}(y_2) \wedge \mathtt{inst}(x, a) \wedge \\ &\quad \mathtt{triple}(x, r, y_1) \wedge \mathtt{inst}(y_1, b) \wedge \mathtt{triple}(x, r, y_2) \wedge \mathtt{inst}(y_2, b) \to \mathtt{inst}(y1, y2)\end{aligned} \tag{19}$$
$$\mathtt{subInverseOf}(r, s) \wedge \mathtt{nom}(x) \wedge \mathtt{nom}(y) \wedge \mathtt{triple}(x, r, y) \to \mathtt{triple}(y, s, x) \tag{20}$$

**Fig. 3.** Inference rules for deriving entailments in Orel

that all axioms have first been decomposed into a simplified normal form that does not use more than one concept operator per concept expression. To simplify the presentation, we use the names of classes, roles, and individuals, as well as $\top$ and $\bot$ as constant symbols in the database instead of assigning numerical identifiers to such names as done in the actual implementation.

Regarding the rules of Fig. 3, we can observe that the rules only derive new facts for the predicates `inst`, `triple`, and `self` that correspond to assertional axioms, as well as for the auxiliary predicate `nom`. To see the purpose of the latter, first note that a special simplification of the rule set is achieved by using the same identifiers for individual names and for nominal classes containing only this individual. Constants that can be considered as nominal classes are marked with `nom`, so that the rule (1) of Fig. 3 generates tautological assertions of the form $\{n\}(n)$. It is not hard to see that all equality statements that can be derived in OWL EL must involve at least one individual name, and can thus be expressed by a class assertion axiom for a nominal class. These observations allow us to simplify the equality theory of [4] to rules (15)–(17) of Fig. 3.

All rules that relate to features that are specific to OWL RL are restricted to individuals in `nom`. This corresponds to the restriction of DL-safety that has been also used in [4]. As noted there, the relevant entailments of an OWL RL ontology can be obtained when restricting reasoning to *named* individuals. *Anonymous* individuals, in contrast, cannot be inferred to exist in OWL RL and are only relevant for the EL part of a knowledge base. As discussed in [4], the DL-safe combination of EL and RL features not only captures all entailments that would be expected from either language in isolation, but also allows some semantic interactions between the two languages. In this case, however, the above inferencing algorithm is not guaranteed to produce all entailments – indeed, a polynomial time algorithm cannot achieve this goal.

Features that are missing in Fig. 2 and 3 are only OWL EL's restricted form of property ranges, the universal role, and concrete domains (data ranges). Orel interprets all property ranges as OWL RL axioms of the form $\top \sqsubseteq \forall R.C$, and does not currently support the universal role. Concrete domains, however, are supported and the according rules are omitted here for reasons of space. Various other features, such as assymmetry of roles, that have been omitted above can readily be expressed in terms of the given features.

Finally, it should be observed that the given inference rules do not materialise facts that can be concluded if the knowledge base is inconsistent. However, it is ensured that inconsistencies lead to derivations of the form $\mathtt{inst}(n, \bot)$ for some constant $n$. Orel checks for this condition for being able to return correct answers without explicitly materialising all possible inferences in the database.

## 3 Schema Reasoning with Orel

The calculus that has been introduced above is able to derive assertional axioms such as the instances of an atomic concept. For complex concept expressions, it might be required to first extend the knowledge base with auxiliary axioms and to (re)complete the materialisation process thereafter. Such auxiliary axioms, however, are hardly affecting the semantics of the knowledge base since they conservatively extend it, and hence many such checks can safely be performed without resetting the database.

The matter is different when checking for the entailment of schema axioms such as concept subsumption. Indeed, there are practically important ontologies such as SNOMED CT which do not contain any individual names, and for which concept subsumption is the chief inferencing problem. It is well known that this problem can be reduced to instance retrieval: for checking if an axiom $A \sqsubseteq B$ is entailed, a new "test" individual $c$ is introduced into the knowledge base together with the assertion $A(c)$. If this implies $B(c)$ then the subsumption is concluded.

Unfortunately, this approach to testing does not preserve the semantics of the knowledge base. Indeed, asserting $A(c)$ may even lead to a global inconsistency (in which case $B(c)$ and thus $A \sqsubseteq B$ is also entailed). Thus, test assertions disallow the naive parallel execution of many queries that could be considered typical for a database system, and they require possibly expensive deletion operations after the test is finished. While it is of course possible to execute each test on a separate copy of the database – possibly realised by marking facts in the database as belonging to a particular test instead of separating databases on the DBMS layer – this approach multiplies the data that has to be stored at each time, and reduces the performance gains due to the re-use of persistently stored previous computations.

The problem is less severe when restricting to smaller languages than OWL EL. For example, the algorithm described in [10] computes all concept subsumptions of an $\mathcal{ELH}$ knowledge base in parallel without executing separate tests for each. While $\mathcal{ELH}$ allows for this mode of reasoning, it is not clear how to establish such an algorithm for

$\mathcal{EL}^{++}$. In particular, the original algorithm as proposed in [7] is incomplete. The glitch can be fixed, but only at the price of specifying the subsumption axiom the entailment of which is to be tested before running the algorithm, thus requiring many runs instead of one. We conjecture that this is unavoidable.

Due to these difficulties, Orel uses a mixed approach for finding concept subsumptions. The calculus uses the simple rules that have been introduced above when this is guaranteed to yield correct results, but it creates additional copies of axioms when the computation results in derivations that cannot be handled in this manner. The goal of our approach is to avoid the significant overhead that is required in the general case whenever possible, but tuning the calculus for this purpose is subject of ongoing work. Currently, Orel's schema inferencing is most efficient when ontologies do not contain nominal classes (in certain problematic contexts), and it decreases in performance when combinations of nominals, existential quantifiers, and OWL RL features occur.

## 4 Applying Derivation Rules on RDBMS

Relational database management systems (RDBMS) are tailored toward the processing of large amounts of data, and the efficient manipulation of such data. As such they appear to be well-suited for implementing materialisation on a persistent storage system. However, inferencing operations are often still rather atypical for RDBMS since they involve large inner joins over all entries in a table. Moreover, RDBMS provide elaborate functions such as transaction management that are not required by typical inferencing scenarios but that can significantly slow down operations. For this reason, various optimisations are needed for using RDBMS as a basis for implementing the outlined inferencing procedure.

It is well-known that datalog rules are closely related to operations in relational algebra [3]. The correspondence is achieved by storing the extension of each datalog predicate in a database table, the columns of which correspond to the arguments taken by the predicate. Rule (3) of Fig. 3 could therefore be realised by the following SQL operation:

```
INSERT INTO inst (x,y) SELECT t1.x AS x, t2.y AS y
FROM subClass AS t1 INNER JOIN inst AS t2 ON t1.x=t2.y
```

Executing this SQL statement extends the `inst` table with all facts that can be derived in one application of rule (3) of Fig. 3. We provide this statement for illustrating the mapping to SQL commands – using it iteratively in an implementation would lead to prohibitively large amounts of unnecessary computations. Indeed, the operation derives the same conclusions in each iteration, just like the original rule does when processed operationally.

Various optimisations have been proposed and thoroughly investigated to overcome this problem [3]. One way to increase efficiency is to keep track of the iteration step in which a fact was derived, and to make sure that rule applications require new facts to be involved in the derivation. This leads to the so-called *semi-naive* bottom-up evaluation which is largely used in Orel. Writing $\texttt{inst}^i$ for the predicate that corresponds to the extension of `inst` as derived in step $i$, this strategy boils down to evaluating the following rule:

$\mathtt{subClass}(x, y) \wedge \mathtt{inst}^i(y, z) \rightarrow \mathtt{inst}^{i+1}(x, z)$

Unfortunately, semi-naive evaluation can still derive large numbers of redundant facts during inferencing. More efficient general purpose optimisations like *magic sets* are available when only certain entailments are of interest (typically at query time) but are not useful for full materialisation. But more efficient forward chaining algorithms do exist as well, and have been studied in the area of databases, and in particular in relation with transitive closure computations [11]. Since these approaches often assume very simple rule sets, they can not be directly adopted to the inference rules of Orel, and part of the ongoing development effort around the tool is to suitably adapt techniques from this area.

## 5 Implementation and Initial Results

Orel is implemented in Java, using the OWL API [12] for accessing OWL documents. The current default RDBMS that is used in Orel is MySQL although only minor adjustments would be needed to move to another RDBMS. Orel is free software and can be obtained (including its source code) from `http://code.google.com/p/orel/`.

The current implementation of Orel is still not fully optimised in various respects. On the one hand, we are exploring heuristics for improving the inferencing control flow. On the other hand, optimising database queries for a particular RDBMS is a tedious process with many dependencies on the technical infrastructure used in testing. We have found that different server setups and machine configurations can lead to a 50% reduction in ontology loading times while incurring a slow-down of several orders of magnitude for materialisation. Thus, while we cannot give reproducible evaluation figures, we can provide some first insights into general runtime behaviour.

The OWL 2 test cases[2] have been used to test the correctness of the implementation. For performance testing, we specifically focussed on the well-known SNOMED CT ontology, a medical terminology of about 425,000 axioms with a strong focus on subclass subsumptions. We considered loading and inference materialisation for this ontology. Load times have shown to be rather similar across very systems of diverse performance, typically ranging between 9min and 20min. These times reflect the slow inserting behaviour of relational databases – the given times are already based on an optimised loading phase that controls transaction management and indexing, and that exploits client-side caching and rewritten bulk updates. Yet, the writing speed is a strong limiting factor (computing the data for writing takes but a few seconds). Application areas for DBMS-based systems of course assume axioms to change at a slow rate, thus reducing the relevance of initial loading times.

Loading does not involve reasoning, i.e. materialisation. At the current stage of implementation, Orel is able to successfully classify SNOMED CT but it cannot compete with highly optimised in-memory systems like Condor [13]: almost 2 hours are needed on a fast database server. This reflects some of the limitations of using an off-the-shelf RDBMS, and we expect significant potential for speed-up by using alternative backends. Similar results have been reported for the SAOR inference engine for OWL Horst

[2] `http://owl.semanticweb.org/`

[14], and we are not aware of any system that uses MySQL as a reasoning backend. In spite of the comparatively low performance of the current implementation, we were still able accomplish major speed improvements for the classification by improving control structure and inference rules. Most of these optimisations are directly applicable to other backends as well.

## 6 Related Work

The objective or Orel is to provide a stable framework for OWL ontology management and inferencing based on persistent storage. Approaches of rule-based bottom-up materialisation of consequences have a long history, and Orel therefore can build on a significant amount of prior work, both practical and theoretical in nature.

On the theoretical side, there is a large body of well-established research to be found in the area of (relational) databases, especially related to the optimisation of recursive queries [3] and the construction of materialised views [15]. We have discussed herein only briefly the basic use of a semi-naive evaluation strategy, but other approaches are applicable in a similar fashion when optimising for further use cases. Typical examples for such techniques are magic sets (used for optimising complex bottom-up computations needed at query time) and incremental materialisation (used for efficiently recreating inferences when new data is added).

More recently, much work has been conducted on “no-SQL” approaches to persistent storage, leading to a number of database-like systems that are tailored toward improved efficiency for non-relational data schemes such as sets of RDF triples, JSON documents, or simple key value pairs. These developments can be beneficial for selecting more suitable storage backends for Orel in the future, but they are not directly related to the work on the current system. Indeed, Orel’s architecture abstracts all storage operations so that inference and control structures do not refer to SQL or any other concrete DBMS feature in any way.

On the more practical side, there are a number of past and current systems that support rule-based inferencing on relational databases. We are not aware of any tool that supports more than a single OWL 2 profile based on such an approach, making Orel’s multi-profile integration novel. Also, the overall architectures of systems differ significantly, even if rule-based inferences are used at the core. The main relationship to Orel therefore is in the actual reasoning module that saturates a knowledge base for a given set of rules, whereas functions such as checking ontology entailment are often highly specific to a given tool. In fact, we do not know of any freely available database-driven reasoner that can check ontology entailment for any OWL profile, the implementation of which was not a minor part of the current Orel system.

The system whose inferencing is most closely related to Orel is the DB reasoner for $\mathcal{ELH}$ [10]. This system supports only a small fragment of the OWL EL profile, but the rules applied for this part are closely related to those used in Orel for the respective features. The only inference problem that DB supports is classification, but it shows some very good performance characteristics for this task, especially regarding memory usage.

The only other database-driven inference engine for $\mathcal{EL}$ that we are aware of is a prototype system that was presented in [16]. In this case, the focus is on conjunctive query answering, with the main contribution being to show that such queries can be answered rather directly on databases with a certain state of materialisation. Loading performance and memory consumption have not been optimised in this work, and are not as good as for the DB reasoner, but outstanding query performance could be obtained. The existence of prototype systems like the above add to our motivation for developing a stable, free platform that can be used to integrate and refine the underlying approaches and algorithms.

Most other database systems that support OWL reasoning focus on OWL RL or on a subset thereof. The most current such implementation that was reported is the OWL reasoner of Oracle 11g.[3] Many systems focus on DLP [17] or pD$^*$ [18], thus providing only incomplete coverage of OWL RL inferencing. Prominent examples include OWLIM [19], DLDB2 [20], and Minerva [21].

Further systems provide yet more restricted amounts of OWL or RDFS inferences mostly for augmenting RDF-based instance data. An interesting example is SAOR for which a non-standard storage implementation has lead to significant performance increases as compared to MySQL [14]. Even though SAOR does not support many OWL features yet, this hints at the potential that non-SQL databases may have for improving the efficiency of systems like Orel.

Finally, rule-based inferencing on top of RDF data has been supported by some tools, the most prominent among which is probably Jena which features a proprietary inference rule implementation.[4] In this case, rules are rather understood in the sense of production rule systems where they form a configurable part of an application that is used to perform relevant computations.

A rather different class of database-driven ontology reasoners are systems that allow for OWL QL querying, such as the QuOnto system.[5] The nature of the problems involved here are somewhat different, and query rewriting often plays a central role. However, recent works in this field have also suggested the use of partial materialisation for improved query performance [22].

## 7 Conclusion and Future Work

We have presented the new ontology inference and management engine Orel, and its underlying approach based on rule-based bottom-up materialisation of consequences in a database. Similar materialisation approaches have been explored for (sometimes incomplete) OWL Full/RDF(S) inferencing, most notably in SAOR [14] and OWLIM [19]. Conversely, there are also a number of fast in-memory implementations available for handling (parts of) OWL EL. Orel is different from both classes of systems as it provides RDBMS based inferencing for OWL EL, and is using a new algorithmic basis that allows for a unified treatment of OWL EL and RL.

[3] `http://www.oracle.com/technology/tech/semantic_technologies/`

[4] `http://jena.sourceforge.net/inference/`

[5] `http://www.dis.uniroma1.it/~quonto/`

The ongoing work on Orel pursues a number of independent goals. Of course, performance is considered as a core challenge, and both the deduction calculus and the storage backend can be improved to address it. For improving the calculus, we develop rule sets that avoid redundant conclusions, and experiment with optimisation methods for efficiently computing closures of datalog programs. Regarding the storage backend, we consider other database paradigms related to recent no-SQL approaches. Another vital feature for a database-driven system are efficient update methods for adding and deleting axioms without recomputing all derivations. Methods for maintenance of materialised views are well known [15], but strongly depend on the details of the implemented calculus.

Besides these obvious goals, there are a number of interesting directions to further develop the core system. Relevant additional features include (conjunctive) query answering, explanation, and extensions with non-standard expressive features such as nonmonotonic inferencing. Other important fields of research concern distribution and parallelisation. At the same time, we seek concrete application scenarios that can be used to explore the practical utility of a robust and scalable OWL inferencing system.

*Acknowledgements.* The work reported herein has been supported by the EU in project ACTIVE (IST-2007-215040) and by the German Research Foundation under the ExpresST project.

## References

1. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S., eds.: OWL 2 Web Ontology Language: Primer. W3C Recommendation (27 October 2009) Available at `http://www.w3.org/TR/owl2-primer/`.
2. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., eds.: OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009) Available at `http://www.w3.org/TR/owl2-profiles/`.
3. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)
4. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. In Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K., eds.: Proc. 7th Int. Semantic Web Conf. (ISWC'08). Volume 5318 of LNCS., Springer (2008) 649–664
5. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B.N., Dean, M.: SWRL: A Semantic Web Rule Language. W3C Member Submission (21 May 2004) Available at `http://www.w3.org/Submission/SWRL/`.
6. Boley, H., Kifer, M., eds.: RIF Basic Logic Dialect. W3C Candidate Recommendation (1 October 2009) Available at `http://www.w3.org/TR/rif-bld/`.
7. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In Kaelbling, L., Saffiotti, A., eds.: Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05), Professional Book Center (2005) 364–369
8. Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N., eds.: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08), IOS Press (2008) 80–84
9. Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. Journal of Web Semantics **3**(1) (2005) 41–60

10. Delaitre, V., Kazakov, Y.: Classifying $\mathcal{ELH}$ ontologies in SQL databases. In Patel-Schneider, P.F., Hoekstra, R., eds.: Proceedings of the OWLED 2009 Workshop on OWL: Experiences and Directions. Volume 529 of CEUR Workshop Proceedings., CEUR-WS.org (2009)
11. Ioannidis, Y.E., Ramakrishnan, R.: Efficient transitive closure algorithms. In Bancilhon, F., DeWitt, D.J., eds.: Proceedings of the 14th International Conference on Very Large Data Bases (VLDB'88), Morgan Kaufmann (1988) 382–394
12. Horridge, M., Bechhofer, S., Noppens, O.: Igniting the OWL 1.1 touch paper: The OWL API. In Golbreich, C., Kalyanpur, A., Parsia, B., eds.: Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions. Volume 258 of CEUR Workshop Proceedings., CEUR-WS.org (2007)
13. Kazakov, Y.: Consequence-driven reasoning for horn $\mathcal{SHIQ}$ ontologies. [23] 2040–2045
14. Hogan, A., Harth, A., Polleres, A.: SAOR: authoritative reasoning for the web. International Journal on Semantic Web and Information Systems (IJSWIS) **2** (2009)
15. Gupta, A., Mumick, I.S., eds.: Materialized Views: Techniques, Implementations, and Applications. MIT Press (1999)
16. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. [23] 2070–2075
17. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proceedings of the 12th International Conference on World Wide Web (WWW'03), ACM (2003) 48–57
18. ter Horst, H.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Journal of Web Semantics **3** (2005)
19. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM – a pragmatic semantic repository for OWL. In: In Proc. Conf. on Web Information Systems Engineering (WISE) Workshops. (2005) 182–192
20. Pan, Z., Zhang, X., Heflin, J.: Dldb2: A scalable multi-perspective semantic web repository. In: Proc. 2008 IEEE/WIC/ACM Int. Conf. on Web Intelligence (WI'08), IEEE (2008) 489–495
21. Zhou, J., Ma, L., Liu, Q., Zhang, L., Yu, Y., Pan, Y.: Minerva: A scalable OWL ontology storage and inference system. In Mizoguchi, R., Shi, Z., Giunchiglia, F., eds.: Proc. 1st Asian Semantic Web Conf. (ASWC'08). Volume 4185 of LNCS., Springer (2006) 429–443
22. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: Combined fo rewritability for conjunctive query answering in dl-lite. In Cuenca Grau, B., Horrocks, I., Motik, B., Sattler, U., eds.: Proceedings of the 22nd International Workshop on Description Logics (DL'09). Volume 477 of CEUR Workshop Proceedings., CEUR-WS.org (2009)
23. Boutilier, C., ed.: Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI'09), IJCAI (2009)