Inline Evaluation of Hybrid Knowledge Bases *
PhD Description

Guohui Xiao, Thomas Eiter

Institute of Information Systems 184/3
Vienna University of Technology
Favoritenstraße 9–11, A–1040 Vienna, Austria
{xiao, eiter}@kr.tuwien.ac.at

Abstract. The deployment of knowledge representation formalisms to the Web has created the need for hybrid formalisms that combine heterogeneous knowledge bases. The aim of this research is to improve the reasoning efficiency over hybrid knowledge bases (KBs). The traditional way of reasoning over hybrid KBs is to use different underlying reasoners to access the different data sources, which causes overhead. To remedy this, we propose a new strategy, called inline evaluation, which compiles the whole hybrid KB into a new KB using only one single formalism. Hence we can use a single reasoner to do the reasoning tasks, and improve the efficiency of hybrid reasoning.

keywords: hybrid KBs, logic programming, description logic, dl-programs

1 Introduction

The deployment of KR formalisms to the Web has created the need for hybrid formalisms that combine heterogeneous knowledge bases (KBs). The combination of logical rules with Description Logics (DLs) now is central to the Semantic Web architecture. Many approaches for defining hybrid KBs have been proposed, cf. [2].

In this research, we focus on dl-programs [4], which are a loose coupling of an ontology and a rule set. The traditional ways of reasoning over dl-programs use a native DL reasoner to reason about the ontology part, and use a native rule reasoner to deal with the rule part. Due to the interaction between the two parts, many calls to the DL and rule reasoner generally can not be avoided, which causes overhead. To remedy this, we propose a new strategy, called inline evaluation, which rewrites the whole dl-program KB into a new KB using only a rule formalism. Hence we can use a single rule reasoner to do the reasoning tasks, and improve the efficiency of reasoning.

The remainder of this paper is structured as follows: In Section 2, we recall the semantics and reasoning approaches of hybrid KBs, mainly dl-programs. Section 3 presents the new general framework of the inline evaluation for hybrid KBs. In Section 4, we discuss the building blocks in this framework, i.e. the ways dealing with dl-programs over different DL fragments. Section 5 concludes with summary.

* This work is supported by the European Commission under the project OntoRule (IST-2009-231875)

2 Hybrid Knowledge Bases

Informally, a hybrid KB is a pair $KB = (\Sigma, P)$, where Σ is a DL based ontology and P is a set of logical rules. The approaches of defining hybrid KBs fall into three categories, following the representational paradigms of the respective approaches: loose coupling, tight coupling, and the embedding approaches [2]. The loose coupling approaches, like *dl-programs* [4] and *F-Logic# KBs* [10], define the interface between the two formalisms based on the exchange of the entailment. The tight coupling approaches, like *SWRL* [12], *r-hybrid KBs* [20] and *ELP* [15], define the interface based on common models. The embedding approaches, like *MKNF KBs* [18], *G-hybrid KBs* [11] and *Open Answer Set Programming* [8], define the interface based on embeddings of both the ontology and the rules in a single unifying non-monotonic formalism.

In this work, we are mainly interested in the loose coupling approach and use dl-programs as a prominent example. DL-programs [4] support a loosely-coupled integration of rules and ontologies, and provide an expressive combination framework based on the interaction of rules with a DL KB via so-called *dl-atoms*. Such dl-atoms query the DL KB by checking for entailment of ground atoms or axioms w.r.t. the KB; as knowledge deduced by the rules can be streamed up to the DL KB in turn, a bidirectional flow of information is possible. As an example, $DL[Student \uplus phd\text{-}student; Person](X)$ is a dl-atom, which intuitively means that the DL predicate $Student$ will be extended by the LP predicate $phd\text{-}student$, and the query result of $Person$ from the extended ontology will be sent to the rule part.

DLVHEX[1] [5] is a prototype for computing the answer set of so-called HEX-programs – an extension of dl-programs for reasoning with external sources (not necessarily DL KBs) under the answer set semantics. By using the *Description Logic Plugin*[2], which interfaces to OWL ontologies via a DL reasoner (currently RacerPro[3]), DLVHEX can reason from dl-programs under the answer set semantics.

To consider a concrete hybrid KB, let $\mathcal{KB} = (\Sigma, P)$ be a dl-program, where $\Sigma = \{\ C \sqsubseteq D\ \}$ and $P = \{p(a) \leftarrow ; \quad s(a) \leftarrow ; \quad s(b) \leftarrow ;$
$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b) \quad (*)\}$. The rule part P of $\mathcal{KB}$ is simple. However, because of the two different dl-atoms $\Lambda = \{\lambda_1 \stackrel{\Delta}{=} DL[C \uplus s; D](a), \lambda_2 \stackrel{\Delta}{=} DL[C \uplus p; D](b)\}$ in rule (*), dlvhex has to call RacerPro at least twice.

It is even worse in the real application when a fixpoint algorithm is often used. In such cases, calls to RacerPro will be performed again and again. While some optimizations, e.g. DL caching [14], are proposed, several calls are unavoidable in general.

3 The Framework of the Inline Evaluation of Hybrid KBs

The aim of this work is to improve the efficiency of hybrid reasoning. To use hybrid KBs in real application, we need efficient reasoners for reasoning tasks. While some reasoning prototypes for hybrid KBs exist, the performance is not satisfactory in general. There are mainly two reasons: (i) to combine the different data sources, the cost

[1] http://www.kr.tuwien.ac.at/research/systems/dlvhex
[2] http://www.kr.tuwien.ac.at/research/systems/dlvhex/dlplugin.html
[3] http://www.racer-systems.com/

of calling of external reasoners is significant; (ii) the inter-leverage is costly and comes with many other issues.

To avoid the overhead of calling external reasoners, we propose a method compiling the hybrid KB into a new KB in one formalism; hence we can only use one reasoner for the compiled KB to do the reasoning tasks. We call such reasoning strategy the *inline evaluation* over hybrid KBs.

Let us continue to consider the dl-program $\mathcal{KB}$ above. Note that each dl-atom sends up a different input/hypothesis to Σ and that entailments for each different input might be different. To this purpose, we copy Σ to new disjoint equivalent versions for each dl-atom, i.e., for each distinct dl-atom λ, we define a new knowledge base Σ_λ that results from replacing all concept and role names by a λ-subscripted version. Thus, for the set $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$ of dl-atoms, we have $\Sigma_{\lambda_i} = \{\ C_{\lambda_i} \sqsubseteq D_{\lambda_i}\ \}, i = 1, 2$.

We translate these disjoint ontologies to a Datalog program, resulting in the rules $\Phi(\Sigma_{\lambda_i}) = \{D_{\lambda_i}(X) \leftarrow C_{\lambda_i}(X)\ \}, i = 1, 2$.

The inputs in the dl-atoms Λ_P can then be encoded as rules $\rho(\Lambda_P)$:

$$\{C_{\lambda_1}(X) \leftarrow s(X); \quad C_{\lambda_2}(X) \leftarrow p(X)\}.$$

It remains to replace the original dl-rules with rules not containing dl-atoms: P^{ord} results from replacing each dl-atom $DL[\lambda; Q](\mathbf{t})$ in P with a new atom $Q_\lambda(\mathbf{t})$, such that $P^o$is the Datalog$^\neg$ program

$$P^o \triangleq \{p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow; \quad q \leftarrow D_{\lambda_1}(a), not\ D_{\lambda_2}(b)\}.$$

One can see that indeed $\mathcal{KB} \models q$ and $\Phi(\Sigma_{\lambda_1}) \cup \Phi(\Sigma_{\lambda_2}) \cup P^o \cup \rho(\Lambda_P) \models q$, effectively reducing reasoning w.r.t. the dl-program to a Datalog$^\neg$ program.

We generalize the above idea to a general framework. Intuitively, one reasoning task over dl-programs can be reduced to another reasoning task over a Datalog program by carefully rewriting each components, and such rewriting should be modular. Then the ontology part can be inline evaluated in the resulting Datalog program. Formally, an *inline evaluation for dl-programs* is a tuple $(\Phi_{DL}, \Phi_{Int}, \Phi_{LP}, \Phi_Q)$, where each component rewrites the ontology (resp. dl-atoms, rule, query) to some Datalog program or query, s.t. for every dl-program $KB = (\Sigma, P)$ and query q, we have $KB \models q$ iff $\bigcup_{\lambda \in \Lambda} \Phi_{DL}(\Sigma, \lambda) \cup \Phi_{Int}(\Lambda) \cup \Phi_{LP}(P) \models \Phi_Q(q)$, where Λ is the set of dl-atoms occurring in KB.

Regarding the reduction/rewriting of DL to Datalog, one can relax this by taking auxiliary relations into account that might depend on the data (more precisely, on the universe of the data). Examples of such auxiliary relations are orderings, or successor relations etc. The important for such relations is that they are "uniform" in the sense that changes to facts (ABox) in the ontology do not affect them. Note that the rewriting of KB is query-independent; optimal query dependent rewriting can also be meaningful, e.g. magic sets [6].

4 Inline Evaluation of DL-Programs over Different DLs

Due to the different representations of the components, we propose to consider several ways of inline evaluations. For each proposal, there are mainly three issues:

– developing an inline evaluation algorithm;
– implementing a prototype reasoner;
– developing some experiments and evaluating the prototype reasoner.

4.1 Tractable Reasoning for DL-Programs over Datalog-rewritable DLs

So far, we have worked on the tractable reasoning for dl-programs over polynomial Datalog-rewritable DLs [9, 23].

We defined a class of Datalog-rewritable DLs, and investigate how reasoning with dl-programs over such DLs under well-founded semantics can be reduced to Datalog$^{\neg}$ (Datalog with negation) by means of an efficient transformation. Noticeably, for dl-programs without negation, the result should be a standard Datalog program; moreover, the transformation preserves stratified negation.

We introduced a particular Datalog-rewritable DL, called $\mathcal{LDL}^{+}$. This DL has no negation and distinguishes between expressions on the left- and right-hand side of axioms. It offers expressive concept- and role expressions on the left-hand side of axioms. $\mathcal{LDL}^{+}$ is tractable under both data and combined complexity; more precisely, we showed that it is PTIME-complete in both settings.

We reviewed the different OWL 2 Profiles and related them to $\mathcal{LDL}^{+}$. While $\mathcal{LDL}^{+}$ misses some constructs, e.g., the *exists restriction* on axiom right-hand sides as in $\mathcal{EL}^{++}$ and $DL\text{-}Lite$, or negation as in the $DL\text{-}Lite$ families, it adds others, e.g., expressive role constructs and *transitive closure* (which is not expressible in first-order logic). Furthermore, we show that $\mathcal{LDL}^{+}$ encompasses Description Logic Programs [7] without a complexity increase.

We developed a prototype reasoner DReW[4], which rewrites $\mathcal{LDL}^{+}$ontologies (dl-programs over $\mathcal{LDL}^{+}$ ontologies) to Datalog (Datalog$^{\neg}$) programs, and calls an underlying rule-based reasoner to perform the actual reasoning. For $\mathcal{LDL}^{+}$ ontologies, it handles instance checking as well as answering of conjunctive queries (CQs). For dl-programs over $\mathcal{LDL}^{+}$ ontologies, it computes the well-founded model [3].

While the basic steps of inline evaluation have exhibited good results, more research is needed.

4.2 Inline Evaluation of DL-Programs over OWL 2 Fragments

OWL 2 RL, OWL 2 EL, and OWL 2 QL are three tractable fragments of OWL 2 [17]. Although they are less expressive than the whole OWL 2 language, they are very scalable and can still capture some useful fragments. We consider how to inline evaluate dl-programs over them.

The central task here is the rewritings to Datalog. For OWL 2 RL, while $\mathcal{LDL}^{+}$already covers OWL 2 RL, another encoding is also available in [17]. For OWL 2 EL, the completion rules for $\mathcal{EL}^{++}$ [1] are essentially Datalog rules. For OWL 2 QL, the techniques of rewriting of Conjunctive Queries over DL-Lite [21] and rewriting of dl-programs over DL-Lite ontology to SQL [22] can be employed.

[4] http://www.kr.tuwien.ac.at/research/systems/drew

4.3 Inline Evaluation of DL-Programs over Horn Fragments of DLs

Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ are Horn fragments of OWL 1 and OWL 2 respectively [19]. They are both ExpTime hard; more precisely, KB satisfiability is ExpTime-complete for Horn-$\mathcal{SHOIQ}$, and is 2-ExpTime-complete for Horn-$\mathcal{SROIQ}$. The main difficulty for inlining stems from, in clause terms, the existential quantifier in the head and that the Herbrand universe is insufficient for evaluation. However, despite their high expressiveness, both DLs above have polynomial data complexity and can be translated to Datalog as shown in [19].

The datalog encoding in [19] is used for the proof of the complexity result. If one directly implements it, the predicate arities depend on the overall number of concept names and roles in the KB. Thus even "small" GCIs may translate into rules with high predicate arities; most of the current Datalog reasoners can not handle them efficiently.

Alternatively, we plan to optimize the rewriting rules in [19], and implement some of the derivations prescribed by the rules inside our reasoner, like [13], then transform the intermediate result to a datalog. We expect this approach will combine the advantage of the consequence driven reasoning, which is efficient for TBox reasoning [13], and the advantage of deductive database, which can handle large size of ABox [16].

4.4 Putting Everything Together

We will implement a prototype systemsupporting different rewriting strategies, which can inline evaluate dl-programs over different DLs. One open issue is how to setting up benchmark to test our system. We will consider test data from different data sources.

5 Summary

The aim of this research is to improve the efficiency of hybrid reasoning. To avoid the overhead of calling external reasoners for hybrid KBs, we propose a new strategy, inline evaluation, which compiles the whole hybrid KBs to a single KB in one formalism. As we can use only one reasoner for the complied KB, the efficiency can be improved.

We expect that the efficiency of hybrid reasoning can be improved for a large part of hybrid KBs. And the complexity of developing reasoners for hybrid KBs can be reduced to encoding to a existing reasoner.

However inline evaluation is not always feasible: (i) Not all the DLs can be efficiently encoded to Datalog ($^\neg$). For example, it is hard to deal with the full OWL 2 DL. (ii) When the source of the KB can not reached, we can not compile it. For example, if some source can only be accessed via some query interface.

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. IJCAI*, pages 364–369. Morgan-Kaufmann Publishers, 2005.
2. J. de Bruijn, P. Bonnard, H. Citeau, S. Dehors, S. Heymans, J. Pührer, and T. Eiter. Combinations of rules and ontologies: State-of-the-art survey of issues. Technical Report Ontorule D3.1, Ontorule Project Consortium, June 2009. http://ontorule-project.eu/.

3. T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer. Well-founded semantics for description logic programs in the Semantic Web. *ACM Trans. Comput. Log.*, 12(2):11, 2011.
4. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.
5. T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *ESWC*, volume 4011 of *LNCS*, pages 273–287. Springer, 2006.
6. W. Faber, G. Greco, and N. Leone. Magic sets for data integration. In D. Fox and C.P. Gomes, editors, *AAAI*, pages 1528–1531. AAAI Press, 2008.
7. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. WWW 2003*, pages 48–57. ACM, 2003.
8. S. Heymans. *Decidable Open Answer Set Programming*. Phd thesis, Theoretical Computer Science Lab (TINF), CS Dept, Vrije Universiteit Brussel, February 2006.
9. S. Heymans, T. Eiter, and G. Xiao. Tractable reasoning with dl-programs over datalog-rewritable description logics. In *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 35–40. IOS Press, 2010.
10. S. Heymans, R. Korf, M. Erdmann, J. Pührer, and T. Eiter. Loosely coupling f-logic rules and ontologies. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI 2010)*, pages 248–255. IEEE Computer Society, 2010.
11. S. Heymans, L. Prediou, C. Feier, J. de Bruijn, and D. van Nieuwenborgh. G-hybrid knowledge bases. In *Proc. of ICLP'06 Workshop on Applications of Logic Programming in the Semantic Web and Semantic Web Services (ALPSWS 2006)*, 2006.
12. I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission, World Wide Web Consortium, 2004.
13. Y. Kazakov. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In Craig Boutilier, editor, *IJCAI*, pages 2040–2045, 2009.
14. T. Krennwallner. Integration of Conjunctive Queries over Description Logics into HEX-Programs. Master's thesis, Vienna University of Technology, October 2007.
15. M. Krötzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In *Proc. ISWC 2008*, pages 649–664, 2008.
16. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, University of Karlsruhe, Karlsruhe, Germany, January 2006.
17. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, editors. *OWL 2 Web Ontology Profiles*. W3C, 2008. W3C Rec. 27 Oct. 2009, http://www.w3.org/TR/owl2-profiles/.
18. B. Motik and R. Rosati. Reconciling Description Logics and Rules. *Journal of the ACM*, 57(5):1–62, 2010.
19. M. Ortiz, S. Rudolph, and M. Simkus. Worst-case optimal reasoning for the Horn-dl fragments of OWL 1 and 2. In *KR*, pages 269–279. AAAI Press, May 2010.
20. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
21. R. Rosati and A. Almatelli. Improving query answering over dl-lite ontologies. In *KR*. AAAI Press, 2010.
22. P. Schneider. Evaluation of description logic programs using an RDBMS. Master's thesis, Vienna University of Technology, December 2010.
23. G. Xiao, S. Heymans, and T. Eiter. DReW: a reasoner for datalog-rewritable description logics and dl-programs. In *Informal Proc. 1st Int'l Workshop on Business Models, Business Rules and Ontologies (BuRO 2010), Sept. 21, 2010, Bressanone/Italy*, 2010. http://ontorule-project.eu/attachments/075_buro2010-proceedings.pdf.